Welcome

Welcome to the official SFML documentation. Here you will find a detailed view of all the SFML classes, as well as source files. If you are looking for tutorials, you can visit the official website at www.sfml-dev.org.
Short example

Here is a short example, to show you how simple it is to use SFML:

```cpp
#include <SFML/ LIChoice.hpp>
#include <SFML/Graphics.hpp>

int main()
{
    // Create the main window
    sf::RenderWindow App(sf::VideoMode(800, 600), "SFML window");

    // Load a sprite to display
    sf::Image Image;
    if (!Image.LoadFromFile("cute_image.jpg"))
        return EXIT_FAILURE;
    sf::Sprite Sprite(Image);

    // Create a graphical string to display
    sf::Font Arial;
    if (!Arial.LoadFromFile("arial.ttf"))
        return EXIT_FAILURE;
    sf::String Text("Hello SFML", Arial, 50);

    // Load a music to play
    sf::Music Music;
    if (!Music.OpenFromFile("nice_music.ogg"))
        return EXIT_FAILURE;

    // Play the music
    Music.Play();

    // Start the game loop
    while (App.IsOpened())
```
```cpp
{
    // Process events
    sf::Event Event;
    while (App.GetEvent(Event))
    {
        // Close window : exit
        if (Event.Type == sf::Event::Closed)
            App.Close();
    }

    // Clear screen
    App.Clear();

    // Draw the sprite
    App.Draw(Sprite);

    // Draw the string
    App.Draw(Text);

    // Update the window
    App.Display();
}

return EXIT_SUCCESS;
}
```
## Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sf::AudioResource</code></td>
<td>Abstract base class for every class that owns a device-dependant resource -- allow them to initialize / shutdown even when the audio context is not created</td>
</tr>
<tr>
<td><code>sf::Clock</code></td>
<td><strong>Clock</strong> is an utility class for manipulating time</td>
</tr>
<tr>
<td><code>sf::Color</code></td>
<td><strong>Color</strong> is an utility class for manipulating 32-bits RGBA colors</td>
</tr>
<tr>
<td><code>sf::Context</code></td>
<td>Class wrapping an OpenGL context</td>
</tr>
<tr>
<td><code>sf::Drawable</code></td>
<td>Abstract base class for every object that can be drawn into a render window</td>
</tr>
<tr>
<td><code>sf::Event</code></td>
<td><strong>Event</strong> defines a system event and its parameters</td>
</tr>
<tr>
<td><code>sf::Event::JoyButtonEvent</code></td>
<td>Joystick buttons events parameters</td>
</tr>
<tr>
<td><code>sf::Event::JoyMoveEvent</code></td>
<td>Joystick axis move event parameters</td>
</tr>
<tr>
<td><code>sf::Event::KeyEvent</code></td>
<td>Keyboard event parameters</td>
</tr>
<tr>
<td><code>sf::Event::MouseButtonEvent</code></td>
<td><strong>Mouse</strong> buttons events parameters</td>
</tr>
<tr>
<td><code>sf::Event::MouseMoveEvent</code></td>
<td><strong>Mouse</strong> move event parameters</td>
</tr>
<tr>
<td><code>sf::Event::MouseWheelEvent</code></td>
<td><strong>Mouse</strong> wheel events parameters</td>
</tr>
<tr>
<td><code>sf::Event::SizeEvent</code></td>
<td>Size events parameters</td>
</tr>
<tr>
<td><code>sf::Event::TextEvent</code></td>
<td>Text event parameters</td>
</tr>
<tr>
<td>Class</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>sf::Font</code></td>
<td>loading and manipulating character fonts</td>
</tr>
<tr>
<td><code>sf::Ftp</code></td>
<td>This class provides methods for manipulating the FTP protocol (described in RFC 959)</td>
</tr>
<tr>
<td><code>sf::Ftp::Ftp::DirectoryResponse</code></td>
<td>Specialization of FTP response returning a directory</td>
</tr>
<tr>
<td><code>sf::Ftp::Ftp::ListingResponse</code></td>
<td>Specialization of FTP response returning a filename listing</td>
</tr>
</tbody>
</table>
| `sf::Ftp::Ftp::Response` | This class wraps a FTP response, which is basically :
|                  | • a status code
|                  | • a message                                                                 |
| `sf::Glyph`    | Structure describing a glyph (a visual character)                          |
| `sf::Http`     | This class provides methods for manipulating the HTTP protocol (described in RFC 1945) |
| `sf::Http::Http::Request` | This class wraps an HTTP request, which is basically :
|                  | • a header with a method, a target URI, and a set of field/value pairs
|                  | • an optional body (for POST requests)                                      |
| `sf::Http::Http::Response` | This class wraps an HTTP response, which is basically :
|                  | • a header with a status code and a set of field/value pairs
|                  | • a body (the content of the requested resource)                            |
| `Image`        | Image is the low-level class for                                              |


<table>
<thead>
<tr>
<th>sf::Image</th>
<th>loading and manipulating images</th>
</tr>
</thead>
<tbody>
<tr>
<td>sf::Input</td>
<td>Input handles real-time input from keyboard and mouse</td>
</tr>
<tr>
<td>sf::IPAddress</td>
<td>IPAddress provides easy manipulation of IP v4 addresses</td>
</tr>
<tr>
<td>sf::Listener</td>
<td>Listener is a global interface for defining the audio listener properties; the audio listener is the point in the scene from where all the sounds are heard</td>
</tr>
<tr>
<td>sf::Lock</td>
<td>Lock is an exception-safe automatic wrapper for locking and unlocking mutexes</td>
</tr>
<tr>
<td>sf::Matrix3</td>
<td>Utility class to manipulate 3x3 matrices representing 2D transformations</td>
</tr>
<tr>
<td>sf::Music</td>
<td>Music defines a big sound played using streaming, so usually what we call a music :)</td>
</tr>
<tr>
<td>sf::Mutex</td>
<td>Mutex defines a mutex (MUTual EXclusion) object, that allows a thread to lock critical instructions to avoid simultaneous access with other threads</td>
</tr>
<tr>
<td>sf::NonCopyable</td>
<td>Utility base class to easily declare non-copyable classes</td>
</tr>
<tr>
<td>sf::Packet</td>
<td>Packet wraps data to send/to receive through the network</td>
</tr>
<tr>
<td>sf::PostFX</td>
<td>PostFX is used to apply a post effect to a window</td>
</tr>
<tr>
<td>sf::Randomizer</td>
<td>Randomizer is an utility class for generating pseudo-random numbers</td>
</tr>
<tr>
<td>sf::Rect&lt; T &gt;</td>
<td>Rect is an utility class for manipulating rectangles</td>
</tr>
<tr>
<td>sf::RenderTarget</td>
<td>Base class for all render targets (window, image, ...)</td>
</tr>
<tr>
<td>---------------------------</td>
<td>--------------------------------------------------------</td>
</tr>
<tr>
<td>sf::RenderWindow</td>
<td>Simple wrapper for <strong>sf::Window</strong> that allows easy 2D rendering</td>
</tr>
<tr>
<td>sf::Resource&lt; T &gt;</td>
<td>Base class for every resource that needs to notify dependent classes about its destruction</td>
</tr>
<tr>
<td>sf::ResourcePtr&lt; T &gt;</td>
<td>Safe pointer to a T resource (inheriting from **sf::Resource&lt;T&gt;*), its pointer is automatically reseted when the resource is destroyed</td>
</tr>
<tr>
<td>sf::Selector&lt; Type &gt;</td>
<td><strong>Selector</strong> allow reading from multiple sockets without blocking</td>
</tr>
<tr>
<td>sf::SelectorBase</td>
<td>Private base class for selectors</td>
</tr>
<tr>
<td>sf::Shape</td>
<td><strong>Shape</strong> defines a drawable convex shape ; it also defines helper functions to draw simple shapes like lines, rectangles, circles, etc</td>
</tr>
<tr>
<td>sf::SocketHelper</td>
<td>This class defines helper functions to do all the non-portable socket stuff</td>
</tr>
<tr>
<td>sf::SocketTCP</td>
<td><strong>SocketTCP</strong> wraps a socket using TCP protocol to send data safely (but a bit slower)</td>
</tr>
<tr>
<td>sf::SocketUDP</td>
<td><strong>SocketUDP</strong> wraps a socket using UDP protocol to send data fastly (but with less safety)</td>
</tr>
<tr>
<td>sf::Sound</td>
<td><strong>Sound</strong> defines the properties of a sound such as position, volume, pitch, etc</td>
</tr>
<tr>
<td>sf::SoundBuffer</td>
<td><strong>SoundBuffer</strong> is the low-level for loading and manipulating sound buffers</td>
</tr>
<tr>
<td>sf::SoundBufferRecorder</td>
<td>Specialized <strong>SoundRecorder</strong> which saves the captured audio</td>
</tr>
<tr>
<td>sf::SoundRecorder</td>
<td><strong>SoundRecorder</strong> is an interface for capturing sound data, it is meant to be used as a base class</td>
</tr>
<tr>
<td>sf::SoundStream</td>
<td><strong>SoundStream</strong> is a streamed sound, ie samples are acquired while the sound is playing</td>
</tr>
<tr>
<td>sf::SoundStream::SoundStream::Chunk</td>
<td>Structure defining a chunk of audio data to stream</td>
</tr>
<tr>
<td>sf::Sprite</td>
<td><strong>Sprite</strong> defines a sprite : texture, transformations, color, and draw on screen</td>
</tr>
<tr>
<td>sf::String</td>
<td><strong>String</strong> defines a graphical 2D text, that can be drawn on screen</td>
</tr>
<tr>
<td>sf::Thread</td>
<td><strong>Thread</strong> defines an easy way to manipulate a thread</td>
</tr>
<tr>
<td>sf::Unicode</td>
<td>Provides utility functions to convert from and to any unicode and ASCII encoding</td>
</tr>
<tr>
<td>sf::Unicode::Unicode::Text</td>
<td>This class is an abstract definition of a unicode text, it can be converted from and to any kind of string and encoding</td>
</tr>
<tr>
<td>sf::Vector2&lt; T &gt;</td>
<td><strong>Vector2</strong> is an utility class for manipulating 2 dimensional vectors</td>
</tr>
<tr>
<td>sf::Vector3&lt; T &gt;</td>
<td><strong>Vector3</strong> is an utility class for manipulating 3 dimensional vectors</td>
</tr>
<tr>
<td>sf::VideoMode</td>
<td><strong>VideoMode</strong> defines a video mode (width, height, bpp, frequency) and provides static functions for getting modes supported by the display device</td>
</tr>
<tr>
<td>sf::View</td>
<td>This class defines a view (position, size, etc</td>
</tr>
<tr>
<td>sf::Window</td>
<td><strong>Window</strong> is a rendering window; it can create a new window or connect to an existing one</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>sf::WindowListener</td>
<td><strong>Base class for classes that want to receive events from a window (for internal use only)</strong></td>
</tr>
<tr>
<td>sf::WindowSettings</td>
<td><strong>Structure defining the creation settings of windows</strong></td>
</tr>
</tbody>
</table>
sf::AudioResource
sf::AudioResource Class Reference

Abstract base class for every class that owns a device-depandant resource -- allow them to initialize / shutdown even when the audio context is not created. More...

#include <AudioResource.hpp>

Inheritance diagram for sf::AudioResource:

List of all members.
Protected Member Functions

<table>
<thead>
<tr>
<th>AudioResource ()</th>
<th>Default constructor.</th>
</tr>
</thead>
<tbody>
<tr>
<td>AudioResource (const AudioResource &amp;)</td>
<td>Copy constructor.</td>
</tr>
<tr>
<td>virtual ~AudioResource ()</td>
<td>Destructor.</td>
</tr>
</tbody>
</table>
Detailed Description

Abstract base class for every class that owns a device-dependant resource -- allow them to initialize / shutdown even when the audio context is not created.

Definition at line 41 of file AudioResource.hpp.
Constructor & Destructor Documentation

sf::AudioResource::AudioResource ( ) [protected]

Default constructor.

Definition at line 37 of file AudioResource.cpp.

sf::AudioResource::AudioResource ( const AudioResource & ) [protected]

Copy constructor.

Definition at line 46 of file AudioResource.cpp.

sf::AudioResource::~AudioResource ( ) [protected, virtual]

Destructor.

Definition at line 55 of file AudioResource.cpp.

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

- AudioResource.hpp
- AudioResource.cpp
sf::Clock
**sf::Clock Class Reference**

*Clock* is an utility class for manipulating time. [More...](#)

```cpp
#include <Clock.hpp>
```

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

<table>
<thead>
<tr>
<th></th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock</td>
<td>()</td>
<td>Default constructor.</td>
</tr>
<tr>
<td>float</td>
<td>GetElapsedTime () const</td>
<td>Get the time elapsed since last reset.</td>
</tr>
<tr>
<td>void</td>
<td>Reset ()</td>
<td>Restart the timer.</td>
</tr>
</tbody>
</table>
**Detailed Description**

**Clock** is an utility class for manipulating time.

Definition at line 39 of file **Clock.hpp**.
### Constructor & Destructor Documentation

<table>
<thead>
<tr>
<th>sf::Clock::Clock ( )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default constructor.</td>
</tr>
<tr>
<td>Definition at line 37 of file Clock.cpp.</td>
</tr>
</tbody>
</table>
## Member Function Documentation

### float sf::Clock::GetElapsedTime () const

Get the time elapsed since last reset.

**Returns:**

Time elapsed, in seconds

Definition at line 46 of file Clock.cpp.

### void sf::Clock::Reset ()

Restart the timer.

Definition at line 55 of file Clock.cpp.

---

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

- **Clock.hpp**
- **Clock.cpp**
<table>
<thead>
<tr>
<th>Main Page</th>
<th>Namespaces</th>
<th>Classes</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class List</td>
<td>Class Index</td>
<td>Class Hierarchy</td>
<td>Class Members</td>
</tr>
</tbody>
</table>

sf::Color
sf::Color Class Reference

Color is an utility class for manipulating 32-bits RGBA colors.

More...

#include <Color.hpp>

List of all members.
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Color ()</strong></td>
<td>Default constructor.</td>
</tr>
<tr>
<td><strong>Color (Uint8 R, Uint8 G, Uint8 B, Uint8 A=255)</strong></td>
<td>Construct the color from its 4 RGBA components.</td>
</tr>
<tr>
<td><strong>Color &amp; operator+= (const Color &amp;Other)</strong></td>
<td>Operator += overload to add a color.</td>
</tr>
<tr>
<td><em><em>Color &amp; operator</em>=(const Color &amp;Other)</em>*</td>
<td>Operator *= overload to modulate a color.</td>
</tr>
<tr>
<td><strong>bool operator==(const Color &amp;Other) const</strong></td>
<td>Compare two colors (for equality).</td>
</tr>
<tr>
<td><strong>bool operator!=(const Color &amp;Other) const</strong></td>
<td>Compare two colors (for difference).</td>
</tr>
</tbody>
</table>
## Public Attributes

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uint8</td>
<td>r</td>
<td>Red component.</td>
</tr>
<tr>
<td>Uint8</td>
<td>g</td>
<td>Green component.</td>
</tr>
<tr>
<td>Uint8</td>
<td>b</td>
<td>Blue component.</td>
</tr>
<tr>
<td>Uint8</td>
<td>a</td>
<td>Alpha (transparency) component.</td>
</tr>
</tbody>
</table>
### Static Public Attributes

<table>
<thead>
<tr>
<th>static const Color</th>
<th>Black</th>
<th>Black predefined color.</th>
</tr>
</thead>
<tbody>
<tr>
<td>static const Color</td>
<td>White</td>
<td>White predefined color.</td>
</tr>
<tr>
<td>static const Color</td>
<td>Red</td>
<td>Red predefined color.</td>
</tr>
<tr>
<td>static const Color</td>
<td>Green</td>
<td>Green predefined color.</td>
</tr>
<tr>
<td>static const Color</td>
<td>Blue</td>
<td>Blue predefined color.</td>
</tr>
<tr>
<td>static const Color</td>
<td>Yellow</td>
<td>Yellow predefined color.</td>
</tr>
<tr>
<td>static const Color</td>
<td>Magenta</td>
<td>Magenta predefined color.</td>
</tr>
<tr>
<td>static const Color</td>
<td>Cyan</td>
<td>Cyan predefined color.</td>
</tr>
</tbody>
</table>
Detailed Description

**Color** is an utility class for manipulating 32-bits RGBA colors.

Definition at line 40 of file **Color.hpp**.
Constructor & Destructor Documentation

<table>
<thead>
<tr>
<th>sf::Color::Color ( )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default constructor.</td>
</tr>
<tr>
<td>Definition at line 50 of file Color.cpp.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sf::Color::Color ( Uint8 R, Uint8 G, Uint8 B, Uint8 A = 255 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Construct the color from its 4 RGBA components.</td>
</tr>
</tbody>
</table>

**Parameters:**

- **R** : Red component (0 .. 255)
- **G** : Green component (0 .. 255)
- **B** : Blue component (0 .. 255)
- **A** : Alpha component (0 .. 255) (255 by default)

Definition at line 63 of file Color.cpp.
Member Function Documentation

**bool sf::Color::operator!= ( const Color & Other ) const**

Compare two colors (for difference).

**Parameters:**
- *Other* : *Color* to compare

**Returns:**
- True if colors are different

Definition at line 113 of file Color.cpp.

**Color & sf::Color::operator*= ( const Color & Other )**

Operator *\=* overload to modulate a color.

**Parameters:**
- *Other* : *Color* to modulate

**Returns:**
- Component-wise multiplication of the two colors

Definition at line 90 of file Color.cpp.

**Color & sf::Color::operator+= ( const Color & Other )**

Operator +\=* overload to add a color.

**Parameters:**
- *Other* : *Color* to add
**Returns:**
Component-wise saturated addition of the two colors

Definition at line 76 of file Color.cpp.

<table>
<thead>
<tr>
<th>bool sf::Color::operator== ( const Color &amp; Other ) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compare two colors (for equality).</td>
</tr>
</tbody>
</table>

**Parameters:**

*Other :* Color to compare

**Returns:**
True if colors are equal

Definition at line 104 of file Color.cpp.
## Member Data Documentation

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
<th>Definition Line</th>
<th>File</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uint8</td>
<td>sf::Color::a</td>
<td>Alpha (transparency) component.</td>
<td>119</td>
<td>Color.hpp</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Uint8</td>
<td>sf::Color::b</td>
<td>Blue component.</td>
<td>118</td>
<td>Color.hpp</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>const</td>
<td>Color sf::Color::Black [static]</td>
<td>Black predefined color.</td>
<td>104</td>
<td>Color.hpp</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>const</td>
<td>Color sf::Color::Blue [static]</td>
<td>Blue predefined color.</td>
<td>108</td>
<td>Color.hpp</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>const</td>
<td>Color sf::Color::Cyan [static]</td>
<td>Cyan predefined color.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Definition at line 111 of file Color.hpp.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------------------------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Uint8 sf::Color::g**

Green component.

Definition at line 117 of file Color.hpp.

<table>
<thead>
<tr>
<th>const Color sf::Color::Green [static]</th>
</tr>
</thead>
</table>

Green predefined color.

Definition at line 107 of file Color.hpp.

<table>
<thead>
<tr>
<th>const Color sf::Color::Magenta [static]</th>
</tr>
</thead>
</table>

Magenta predefined color.

Definition at line 110 of file Color.hpp.

<table>
<thead>
<tr>
<th>Uint8 sf::Color::r</th>
</tr>
</thead>
</table>

Red component.

Definition at line 116 of file Color.hpp.

<table>
<thead>
<tr>
<th>const Color sf::Color::Red [static]</th>
</tr>
</thead>
</table>

Red predefined color.

Definition at line 106 of file Color.hpp.
const Color sf::Color::White [static]

White predefined color.

Definition at line 105 of file Color.hpp.

const Color sf::Color::Yellow [static]

Yellow predefined color.

Definition at line 109 of file Color.hpp.

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

- Color.hpp
- Color.cpp

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
sf::Context
sf::Context Class Reference

Class wrapping an OpenGL context. More...

#include <Context.hpp>

Inheritance diagram for sf::Context:

List of all members.
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Context ()</strong></td>
<td>Default constructor, create the context.</td>
</tr>
<tr>
<td><strong>~Context ()</strong></td>
<td>Destructor, destroy the context.</td>
</tr>
<tr>
<td><strong>SetActive (bool Active)</strong></td>
<td>Activate or deactivate the context.</td>
</tr>
</tbody>
</table>
**Static Public Member Functions**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static bool</td>
<td><strong>IsContextActive</strong> ()</td>
</tr>
<tr>
<td></td>
<td><em>Check if there's a context bound to the current thread.</em></td>
</tr>
<tr>
<td>static Context &amp;</td>
<td><strong>GetGlobal</strong> ()</td>
</tr>
<tr>
<td></td>
<td><em>Get the global context.</em></td>
</tr>
</tbody>
</table>
Detailed Description

Class wrapping an OpenGL context.

All SFML windows already have their own context, so this class is more a helper for specific issues involving OpenGL and multi-threading. It's meant to be used internally.

Definition at line 49 of file Context.hpp.
## Constructor & Destructor Documentation

<table>
<thead>
<tr>
<th>sf::Context::Context ( )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default constructor, create the context.</td>
</tr>
<tr>
<td>Definition at line 44 of file Context.cpp.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sf::Context::~Context ( )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Destructor, destroy the context.</td>
</tr>
<tr>
<td>Definition at line 53 of file Context.cpp.</td>
</tr>
</tbody>
</table>
Member Function Documentation

**Context & sf::Context::GetGlobal ( ) [static]**

Get the global context.

**Returns:**
Reference to the global context

Definition at line 80 of file Context.cpp.

**bool sf::Context::IsContextActive ( ) [static]**

Check if there's a context bound to the current thread.

**Returns:**
True if there's a context bound to the current thread

Definition at line 71 of file Context.cpp.

**void sf::Context::SetActive ( bool Active )**

Activate or deactivate the context.

**Parameters:**
Active : True to activate the context, false to deactivate it

Definition at line 62 of file Context.cpp.

The documentation for this class was generated from the following files:
sf::Drawable
sf::Drawable Class Reference

Abstract base class for every object that can be drawn into a render window. More...

#include <Drawable.hpp>

Inheritance diagram for sf::Drawable:

List of all members.
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Drawable</strong> (const Vector2f &amp;Position=Vector2f(0, 0), const Vector2f &amp;Scale=Vector2f(1, 1), float Rotation=0.f, const Color &amp;Col=Color(255, 255, 255))</td>
<td>Default constructor.</td>
</tr>
<tr>
<td>virtual ~Drawable ()</td>
<td>Virtual destructor.</td>
</tr>
<tr>
<td>void SetPosition (float X, float Y)</td>
<td>Set the position of the object (take 2 values).</td>
</tr>
<tr>
<td>void SetPosition (const Vector2f &amp;Position)</td>
<td>Set the position of the object (take a 2D vector).</td>
</tr>
<tr>
<td>void SetX (float X)</td>
<td>Set the X position of the object.</td>
</tr>
<tr>
<td>void SetY (float Y)</td>
<td>Set the Y position of the object.</td>
</tr>
<tr>
<td>void SetScale (float ScaleX, float ScaleY)</td>
<td>Set the scale of the object (take 2 values).</td>
</tr>
<tr>
<td>void SetScale (const Vector2f &amp;Scale)</td>
<td>Set the scale of the object (take a 2D vector).</td>
</tr>
<tr>
<td>void SetScaleX (float FactorX)</td>
<td>Set the X scale factor of the object.</td>
</tr>
<tr>
<td>void SetScaleY (float FactorY)</td>
<td>Set the Y scale factor of the object.</td>
</tr>
<tr>
<td>void SetCenter (float CenterX, float CenterY)</td>
<td>Set the center of the object, in coordinates relative to the top-left of the object (take 2 values).</td>
</tr>
<tr>
<td>void SetCenter (const Vector2f &amp;Center)</td>
<td>Set the center of the object, in coordinates relative to the top-left of the object (take a 2D vector).</td>
</tr>
<tr>
<td>void SetRotation (float Rotation)</td>
<td>Set the orientation of the object.</td>
</tr>
<tr>
<td>void SetColor (const Color &amp;Col)</td>
<td>Set the color of the object.</td>
</tr>
<tr>
<td>void SetBlendMode (Blend::Mode Mode)</td>
<td>Set the blending mode for the object.</td>
</tr>
<tr>
<td>const Vector2f &amp; GetPosition () const</td>
<td>Get the position of the object.</td>
</tr>
<tr>
<td>const Vector2f &amp; GetScale () const</td>
<td>Get the current scale of the object.</td>
</tr>
<tr>
<td>const Vector2f &amp; GetCenter () const</td>
<td>Get the center of the object.</td>
</tr>
<tr>
<td>float GetRotation () const</td>
<td>Get the orientation of the object.</td>
</tr>
</tbody>
</table>
const Color & GetColor () const
Get the color of the object.

Blend::Mode GetBlendMode () const
Get the current blending mode.

void Move (float OffsetX, float OffsetY)
Move the object of a given offset (take 2 values).

void Move (const Vector2f &Offset)
Move the object of a given offset (take a 2D vector).

void Scale (float FactorX, float FactorY)
Scale the object (take 2 values).

void Scale (const Vector2f &Factor)
Scale the object (take a 2D vector).

void Rotate (float Angle)
Rotate the object.

sf::Vector2f TransformToLocal (const sf::Vector2f &Point) const
Transform a point from global coordinates into local coordinates (ie it applies the inverse of object's center, translation, rotation and scale to the point).

sf::Vector2f TransformToGlobal (const sf::Vector2f &Point) const
Transform a point from local coordinates into global coordinates (ie it applies the object's center, translation, rotation and scale to the point).
## Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>const Matrix3 &amp; GetMatrix () const</code></td>
<td>Get the transform matrix of the drawable.</td>
</tr>
<tr>
<td><code>const Matrix3 &amp; GetInverseMatrix () const</code></td>
<td>Get the inverse transform matrix of the drawable.</td>
</tr>
</tbody>
</table>
Friends

class   RenderTarget
Detailed Description

Abstract base class for every object that can be drawn into a render window.

Definition at line 58 of file Drawable.hpp.
## Constructor & Destructor Documentation

### sf::Drawable::Drawable

```cpp
sf::Drawable::Drawable ( const Vector2f & Position = Vector2f(0, 0),
                         const Vector2f & Scale = Vector2f(1, 1),
                         float Rotation = 0.f,
                         const Color & Col = Color(255, 255, 255, 255))
```

Default constructor.

**Parameters:**
- **Position**: Position of the object (0, 0 by default)
- **Scale**: Scale factor (1, 1 by default)
- **Rotation**: Orientation, in degrees (0 by default)
- **Col**: Color of the object (white by default)

Definition at line 39 of file `Drawable.cpp`.

### sf::Drawable::~Drawable

```cpp
sf::Drawable::~Drawable ( ) [virtual]
```

Virtual destructor.

Definition at line 56 of file `Drawable.cpp`. 

---
**Member Function Documentation**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Returns</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Blend::Mode</strong> sf::Drawable::GetBlendMode() const</td>
<td>Get the current blending mode.</td>
<td>Current blending mode</td>
<td>Definition at line 258 of file <strong>Drawable.cpp</strong>.</td>
</tr>
<tr>
<td><strong>const Vector2f &amp;</strong> sf::Drawable::GetCenter() const</td>
<td>Get the center of the object.</td>
<td>Current position of the center</td>
<td>Definition at line 231 of file <strong>Drawable.cpp</strong>.</td>
</tr>
<tr>
<td><strong>const Color &amp;</strong> sf::Drawable::GetColor() const</td>
<td>Get the color of the object.</td>
<td>Current color</td>
<td>Definition at line 249 of file <strong>Drawable.cpp</strong>.</td>
</tr>
<tr>
<td><strong>const Matrix3 &amp;</strong> sf::Drawable::GetInverseMatrix() const [protected]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Get the inverse transform matrix of the drawable.

**Returns:**
Inverse transform matrix

Definition at line 350 of file **Drawable.cpp**.

```cpp
const Matrix3 & sf::Drawable::GetMatrix() const [protected]
```

Get the transform matrix of the drawable.

**Returns:**
Transform matrix

Definition at line 334 of file **Drawable.cpp**.

```cpp
const Vector2f & sf::Drawable::GetPosition() const
```

Get the position of the object.

**Returns:**
Current position

Definition at line 213 of file **Drawable.cpp**.

```cpp
float sf::Drawable::GetRotation() const
```

Get the orientation of the object.

Rotation is always in the range [0, 360]

**Returns:**
**Current rotation, in degrees**

Definition at line 240 of file `Drawable.cpp`.

```cpp
const Vector2f & sf::Drawable::GetScale() const
```

Get the current scale of the object.

**Returns:**
Current scale factor (always positive)

Definition at line 222 of file `Drawable.cpp`.

```cpp
void sf::Drawable::Move(const Vector2f & Offset)
```

Move the object of a given offset (take a 2D vector).

**Parameters:**

- `Offset`: Amount of units to move the object of

Definition at line 278 of file `Drawable.cpp`.

```cpp
void sf::Drawable::Move(float OffsetX,
                        float OffsetY)
```

Move the object of a given offset (take 2 values).

**Parameters:**

- `OffsetX`: X offset
- `OffsetY`: Y offset
void sf::Drawable::Rotate ( float Angle )

Rotate the object.

**Parameters:**

*Angle*: Angle of rotation, in degrees

Definition at line 268 of file Drawable.cpp.

void sf::Drawable::Scale ( const Vector2f & Factor )

Scale the object (take a 2D vector).

**Parameters:**

*Factor*: Scaling factors (both values must be strictly positive)

Definition at line 306 of file Drawable.cpp.

void sf::Drawable::Scale ( float FactorX, float FactorY )

Scale the object (take 2 values).

**Parameters:**

*FactorX*: Scaling factor on X (must be strictly positive)
*FactorY*: Scaling factor on Y (must be strictly positive)

Definition at line 297 of file Drawable.cpp.
void sf::Drawable::SetBlendMode ( Blend::Mode Mode )

Set the blending mode for the object.

The default blend mode is Blend::Alpha

**Parameters:**

*Mode* : New blending mode

The default blend mode is Blend::Alpha

Definition at line 204 of file Drawable.cpp.

void sf::Drawable::SetCenter ( const Vector2f & Center )

Set the center of the object, in coordinates relative to the top-left of the object (take a 2D vector).

Set the center of the object, in coordinates relative to the top-left of the object (take a 2D vector).

The default center is (0, 0)

**Parameters:**

*Center* : New center

The default center is (0, 0)

Definition at line 171 of file Drawable.cpp.

void sf::Drawable::SetCenter ( float centerX, float centerY )
Set the center of the object, in coordinates relative to the top-left of the object (take 2 values).

Set the center of the object, in coordinates relative to the top-left of the object (take 2 values).

The default center is (0, 0)

**Parameters:**

- **CenterX**: X coordinate of the center
- **CenterY**: Y coordinate of the center

The default center is (0, 0)

Definition at line **157** of file **Drawable.cpp**.

```cpp
void sf::Drawable::SetColor ( const Color & Col )
```

Set the color of the object.

The default color is white

**Parameters:**

- **Col**: New color

The default color is white

Definition at line **194** of file **Drawable.cpp**.

```cpp
void sf::Drawable::setPosition ( const Vector2f & Position )
```

Set the position of the object (take a 2D vector).
Parameters:

- **Position**: New position

Definition at line 75 of file `Drawable.cpp`.

```cpp
void sf::Drawable::SetPosition ( float X, float Y )
```

Set the position of the object (take 2 values).

Parameters:

- **X**: New X coordinate
- **Y**: New Y coordinate

Definition at line 65 of file `Drawable.cpp`.

```cpp
void sf::Drawable::SetRotation ( float Rotation )
```

Set the orientation of the object.

Parameters:

- **Rotation**: Angle of rotation, in degrees

Definition at line 180 of file `Drawable.cpp`.

```cpp
void sf::Drawable::SetScale ( const Vector2f & Scale )
```

Set the scale of the object (take a 2D vector).

Parameters:

- **Scale**: New scale (both values must be strictly positive)
void sf::Drawable::setScale(float ScaleX, float ScaleY)

Set the scale of the object (take 2 values).

**Parameters:**
- `ScaleX` : New horizontal scale (must be strictly positive)
- `ScaleY` : New vertical scale (must be strictly positive)

void sf::Drawable::setScaleX(float FactorX)

Set the X scale factor of the object.

**Parameters:**
- `X` : New X scale factor

void sf::Drawable::setScaleY(float FactorY)

Set the Y scale factor of the object.

**Parameters:**
- `Y` : New Y scale factor
void sf::Drawable::SetX ( float \textit{X} )

Set the X position of the object.

\textbf{Parameters:}
\[
\textit{X} : \text{New X coordinate}
\]

Definition at line 85 of file \texttt{Drawable.cpp}.

void sf::Drawable::SetY ( float \textit{Y} )

Set the Y position of the object.

\textbf{Parameters:}
\[
\textit{Y} : \text{New Y coordinate}
\]

Definition at line 96 of file \texttt{Drawable.cpp}.

\texttt{sf::Vector2f sf::Drawable::TransformToGlobal ( const sf::Vector2f & \textit{Point} )}

Transform a point from local coordinates into global coordinates (ie it applies the object's center, translation, rotation and scale to the point).

Transform a point from local coordinates into global coordinates (ie it applies the object's center, translation, rotation and scale to the point).

\textbf{Parameters:}
\[
\textit{Point} : \text{Point to transform}
\]

\textbf{Returns:}
Transformed point
sf::Vector2f sf::Drawable::TransformToLocal ( const sf::Vector2f & Point ) const

Transform a point from global coordinates into local coordinates (ie it applies the inverse of object's center, translation, rotation and scale to the point).

Parameters:
  Point : Point to transform

Returns:
  Transformed point

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

- **Drawable.hpp**
- **Drawable.cpp**

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
  Documentation generated by doxygen 1.5.2 ::
sf::Event
sf::Event Class Reference

Event defines a system event and its parameters. More...

#include <Event.hpp>

List of all members.
## Classes

<table>
<thead>
<tr>
<th>struct</th>
<th>JoyButtonEvent</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Joystick buttons events parameters. <a href="#">More...</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>struct</th>
<th>JoyMoveEvent</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Joystick axis move event parameters. <a href="#">More...</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>struct</th>
<th>KeyEvent</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Keyboard event parameters. <a href="#">More...</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>struct</th>
<th>MouseButtonEvent</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mouse buttons events parameters. <a href="#">More...</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>struct</th>
<th>MouseMoveEvent</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mouse move event parameters. <a href="#">More...</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>struct</th>
<th>MouseWheelEvent</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mouse wheel events parameters. <a href="#">More...</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>struct</th>
<th>SizeEvent</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Size events parameters. <a href="#">More...</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>struct</th>
<th>TextEvent</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Text event parameters. <a href="#">More...</a></td>
</tr>
</tbody>
</table>
### Public Types

```c
enum EventType {
    Closed,
    Resized,
    LostFocus,
    GainedFocus,
    TextEntered,
    KeyPressed,
    KeyReleased,
    MouseWheelMoved,
    MouseButtonPressed,
    MouseButtonReleased,
    MouseMoved,
    MouseEntered,
    MouseLeft,
    JoyButtonPressed,
    JoyButtonReleased,
    JoyMoved,
    Count
}
```

*Enumeration of the different types of events.* [More...](#)
Public Attributes

<table>
<thead>
<tr>
<th>EventType</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>KeyEvent</td>
<td>Key</td>
</tr>
<tr>
<td>TextEvent</td>
<td>Text</td>
</tr>
<tr>
<td>MouseMoveEvent</td>
<td>MouseMove</td>
</tr>
<tr>
<td>MouseButtonEvent</td>
<td>MouseButton</td>
</tr>
<tr>
<td>MouseWheelEvent</td>
<td>MouseWheel</td>
</tr>
<tr>
<td>JoyMoveEvent</td>
<td>JoyMove</td>
</tr>
<tr>
<td>JoyButtonEvent</td>
<td>JoyButton</td>
</tr>
<tr>
<td>SizeEvent</td>
<td>Size</td>
</tr>
</tbody>
</table>

union {

};
Detailed Description

**Event** defines a system event and its parameters.

Definition at line 197 of file `Event.hpp`.
Member Enumeration Documentation

<table>
<thead>
<tr>
<th>enum sf::Event::EventType</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enumeration of the different types of events.</td>
</tr>
<tr>
<td>Definition at line 278 of file Event.hpp.</td>
</tr>
</tbody>
</table>
Member Data Documentation

<table>
<thead>
<tr>
<th>EventType sf::Event::Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type of the event.</td>
</tr>
<tr>
<td>Definition at line 303 of file Event.hpp.</td>
</tr>
</tbody>
</table>

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

- Event.hpp

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
  Documentation generated by doxygen 1.5.2 ::
sf::Event::JoyButtonEvent
sf::Event::JoyButtonEvent Struct Reference

Joystick buttons events parameters. More...

#include <Event.hpp>

List of all members.
Public Attributes

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>unsigned int</td>
<td>JoystickId</td>
</tr>
<tr>
<td>unsigned int</td>
<td>Button</td>
</tr>
</tbody>
</table>
Detailed Description

Joystick buttons events parameters.

Definition at line 260 of file Event.hpp.

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

- Event.hpp

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
sf::Event::JoyMoveEvent
sf::Event::Event::JoyMoveEvent Struct Reference

Joystick axis move event parameters. More...

#include <Event.hpp>

List of all members.
<table>
<thead>
<tr>
<th>Data Type</th>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unsigned int</td>
<td>JoystickId</td>
<td></td>
</tr>
<tr>
<td>Joy::Axis</td>
<td>Axis</td>
<td></td>
</tr>
<tr>
<td>float</td>
<td>Position</td>
<td></td>
</tr>
</tbody>
</table>
Detailed Description

Joystick axis move event parameters.

Definition at line 250 of file Event.hpp.

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

- Event.hpp

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
sf::Event::KeyEvent
sf::Event::Event::KeyEvent Struct Reference

Keyboard event parameters. More...

#include <Event.hpp>

List of all members.
## Public Attributes

<table>
<thead>
<tr>
<th>Key::Code</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool</td>
<td>Alt</td>
</tr>
<tr>
<td>bool</td>
<td>Control</td>
</tr>
<tr>
<td>bool</td>
<td>Shift</td>
</tr>
</tbody>
</table>
Detailed Description

Keyboard event parameters.

Definition at line 204 of file Event.hpp.

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

- Event.hpp
sf::Event::MouseButtonEvent
sf::Event::Event::MouseButtonEvent
Struct Reference

Mouse buttons events parameters. More...

#include <Event.hpp>

List of all members.
### Public Attributes

<table>
<thead>
<tr>
<th>Mouse::Button</th>
<th>Button</th>
<th>int</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>int</td>
<td>Y</td>
</tr>
</tbody>
</table>
Detailed Description

**Mouse** buttons events parameters.

Definition at line **232** of file **Event.hpp**.

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

- **Event.hpp**
sf::Event::MouseMoveEvent
sf::Event::Event::MouseMoveEvent

Struct Reference

**Mouse** move event parameters. [More...](#)

```cpp
#include <Event.hpp>
```

[List of all members.](#)
**Public Attributes**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>int X</strong></td>
<td></td>
</tr>
<tr>
<td><strong>int Y</strong></td>
<td></td>
</tr>
</tbody>
</table>
Detailed Description

Mouse move event parameters.

Definition at line 223 of file Event.hpp.

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

- Event.hpp

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
<table>
<thead>
<tr>
<th>Main Page</th>
<th>Namespaces</th>
<th>Classes</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Class List</td>
<td>Class Index</td>
<td>Class Hierarchy</td>
<td>Class Members</td>
</tr>
</tbody>
</table>

**sf::Event::MouseWheelEvent**
sf::Event::Event::MouseWheelEvent

Struct Reference

Mouse wheel events parameters. More...

#include <Event.hpp>

List of all members.
### Public Attributes

<table>
<thead>
<tr>
<th></th>
<th>Delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>Delta</td>
</tr>
</tbody>
</table>
Detailed Description

Mouse wheel events parameters.

Definition at line 242 of file Event.hpp.

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

- Event.hpp

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
sf::Event::SizeEvent
sf::Event::Event::SizeEvent Struct Reference

Size events parameters. More...

#include <Event.hpp>

List of all members.
## Public Attributes

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>unsigned int</td>
<td>Width</td>
</tr>
<tr>
<td>unsigned int</td>
<td>Height</td>
</tr>
</tbody>
</table>
Detailed Description

Size events parameters.

Definition at line 269 of file Event.hpp.

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

- Event.hpp

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
   Documentation generated by doxygen 1.5.2 ::
sf::Event::TextEvent
sf::Event::Event::TextEvent Struct Reference

Text event parameters. More...

#include <Event.hpp>

List of all members.
**Public Attributes**

<table>
<thead>
<tr>
<th>Uint32</th>
<th>Unicode</th>
</tr>
</thead>
</table>


Detailed Description

Text event parameters.

Definition at line 215 of file Event.hpp.

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

- Event.hpp
sf::Font Class Reference

Font is the low-level class for loading and manipulating character fonts. More...

#include <Font.hpp>

Inheritance diagram for sf::Font:

List of all members.
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Font</strong> ()</td>
<td>Default constructor.</td>
</tr>
<tr>
<td><strong>bool LoadFromFile</strong> (const std::string &amp;Filename, unsigned int CharSize=30, const Unicode::Text &amp;Charset=ourDefaultCharset)</td>
<td>Load the font from a file.</td>
</tr>
<tr>
<td><strong>bool LoadFromMemory</strong> (const char *Data, std::size_t SizeInBytes, unsigned int CharSize=30, const Unicode::Text &amp;Charset=ourDefaultCharset)</td>
<td>Load the font from a file in memory.</td>
</tr>
<tr>
<td><strong>unsigned int GetCharacterSize () const</strong></td>
<td>Get the base size of characters in the font; All glyphs dimensions are based on this value.</td>
</tr>
<tr>
<td><strong>const Glyph &amp; GetGlyph</strong> (Uint32 CodePoint) const</td>
<td>Get the description of a glyph (character) given by its unicode value.</td>
</tr>
<tr>
<td><strong>const Image &amp; GetImage</strong> () const</td>
<td>Get the image containing the rendered characters (glyphs).</td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>static const Font &amp; DefaultFont()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get the SFML default built-in font (Arial).</td>
</tr>
</tbody>
</table>
Detailed Description

Font is the low-level class for loading and manipulating character fonts.

This class is meant to be used by sf::String

Definition at line 54 of file Font.hpp.
## Constructor & Destructor Documentation

<table>
<thead>
<tr>
<th>sf::Font::Font ( )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default constructor.</td>
</tr>
<tr>
<td>Definition at line 64 of file Font.cpp.</td>
</tr>
</tbody>
</table>
### Member Function Documentation

<table>
<thead>
<tr>
<th>Function</th>
<th>Documentation</th>
</tr>
</thead>
</table>
| `unsigned int sf::Font::GetCharacterSize ( ) const` | Get the base size of characters in the font; All glyphs dimensions are based on this value.  
Get the base size of characters in the font; All glyphs dimensions are based on this value.  
**Returns:**  
Base size of characters  
Definition at line 120 of file `Font.cpp`. |
| `const Font & sf::Font::GetDefaultFont ( ) [static]` | Get the SFML default built-in font (Arial).  
**Returns:**  
Instance of the default font  
Definition at line 159 of file `Font.cpp`. |
| `const Glyph & sf::Font::GetGlyph ( Uint32 CodePoint ) const` | Get the description of a glyph (character) given by its unicode value.  
Get the description of a glyph (character) given by its unicode value. |
**Parameters:**

- CodePoint : *Unicode* value of the character to get

**Returns:**

- Glyph's visual settings, or an invalid glyph if character not found

Definition at line 130 of file `Font.cpp`.

---

**const Image & sf::Font::GetImage ( ) const**

Get the image containing the rendered characters (glyphs).

**Returns:**

- *Image* containing glyphs

Definition at line 150 of file `Font.cpp`.

---

**bool sf::Font::LoadFromFile ( const std::string & Filename, unsigned int CharSize = 30, const Unicode::Text & Charset = ourDefaultCharset )**

Load the font from a file.

**Parameters:**

- Filename : *Font* file to load
- CharSize : Size of characters in bitmap - the bigger, the higher quality (30 by default)
- Charset : Characters set to generate (by default, contains the ISO-8859-1 printable characters)

**Returns:**
True if loading was successful

Definition at line 74 of file Font.cpp.

```cpp
bool sf::Font::LoadFromMemory(const char * Data,
                              std::size_t SizelnBytes,
                              unsigned int CharSize = 30,
                              const Unicode::Text & Charset = ourDefaultCharset)
```

Load the font from a file in memory.

**Parameters:**

- **Data**: Pointer to the data to load
- **SizelnBytes**: Size of the data, in bytes
- **CharSize**: Size of characters in bitmap - the bigger, the higher quality (30 by default)
- **Charset**: Characters set to generate (by default, contains the ISO-8859-1 printable characters)

**Returns:**

True if loading was successful

Definition at line 93 of file Font.cpp.

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

- **Font.hpp**
- **Font.cpp**

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
sf::Ftp
This class provides methods for manipulating the FTP protocol (described in RFC 959). More...

#include <Ftp.hpp>

Inheritance diagram for sf::Ftp:

List of all members.
## Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DataChannel</strong></td>
<td></td>
</tr>
<tr>
<td><strong>DirectoryResponse</strong></td>
<td>Specialization of FTP response returning a directory. <strong><a href="#">More...</a></strong></td>
</tr>
<tr>
<td><strong>ListingResponse</strong></td>
<td>Specialization of FTP response returning a filename listing. <strong><a href="#">More...</a></strong></td>
</tr>
<tr>
<td><strong>Response</strong></td>
<td>This class wraps a FTP response, which is basically:</td>
</tr>
<tr>
<td></td>
<td>- a status code</td>
</tr>
<tr>
<td></td>
<td>- a message</td>
</tr>
</tbody>
</table>

**More...**
Public Types

<table>
<thead>
<tr>
<th>enum TransferMode { Binary, Ascii, Ebcdic }</th>
</tr>
</thead>
</table>

Enumeration of transfer modes. More...
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>~Ftp()</td>
<td>Destructor -- close the connection with the server.</td>
</tr>
</tbody>
</table>
| Response     | **Connect** (const IPAddress &Server, unsigned short Port=21, float Timeout=0.f)  
               Connect to the specified FTP server. |
| Response     | **Login** ()  
               Log in using anonymous account. |
| Response     | **Login** (const std::string &UserName, const std::string &Password)  
               Log in using a username and a password. |
| Response     | **Disconnect** ()  
               Close the connection with FTP server. |
| Response     | **KeepAlive** ()  
               Send a null command just to prevent from being disconnected. |
| DirectoryResponse | **GetWorkingDirectory** ()  
                        Get the current working directory. |
| ListingResponse | **GetDirectoryListing** (const std::string &Directory="")  
                       Get the contents of the given directory (subdirectories and files). |
| Response     | **ChangeDirectory** (const std::string &Directory)  
                       Change the current working directory. |
| Response     | **ParentDirectory** ()  
                       Go to the parent directory of the current one. |
| Response     | **MakeDirectory** (const std::string &Name)  
                       Create a new directory. |
| Response     | **DeleteDirectory** (const std::string &Name)  
                       Remove an existing directory. |
| Response     | **RenameFile** (const std::string &File, const std::string &NewName)  
                       Rename a file. |
| Response     | **DeleteFile** (const std::string &Name)  
                       Remove an existing file. |
| Response     | **Download** (const std::string &DistantFile, const std::string &DestPath, TransferMode Mode=Binary)  
                       Download a file from the server. |
| Response     | **Upload** (const std::string &LocalFile, const std::string &DestPath, TransferMode Mode=Binary)  
                       Upload a file to the server. |
<table>
<thead>
<tr>
<th>class</th>
<th>DataChannel</th>
</tr>
</thead>
</table>

Friends
Detailed Description

This class provides methods for manipulating the FTP protocol (described in RFC 959).

It provides easy access and transfers to remote directories and files on a FTP server

Definition at line 47 of file Ftp.hpp.
**Member Enumeration Documentation**

<table>
<thead>
<tr>
<th>Enumerations</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>sf::Ftp::TransferMode</strong></td>
</tr>
</tbody>
</table>

Enumeration of transfer modes.

**Enumerator:**

- **Binary**  Binary mode (file is transfered as a sequence of bytes).
- **Ascii** Text mode using ASCII encoding.
- **Ebcdic** Text mode using EBCDIC encoding.

Definition at line 54 of file `Ftp.hpp`. 
Constructor & Destructor Documentation

sf::Ftp::~Ftp ( )

Destructor -- close the connection with the server.

Definition at line 186 of file Ftp.cpp.
Member Function Documentation

Ftp::Response sf::Ftp::ChangeDirectory ( const std::string & Directory )

Change the current working directory.

Parameters:
    Directory : New directory, relative to the current one

Returns:
    Server response to the request

Definition at line 291 of file Ftp.cpp.

Ftp::Response sf::Ftp::Connect ( const IPAddress & Server, unsigned short Port = 21, float Timeout = 0.f )

Connect to the specified FTP server.

Parameters:
    Server : FTP server to connect to
    Port : Port used for connection (21 by default, standard FTP port)
    Timeout : Maximum time to wait, in seconds (0 by default, means no timeout)

Returns:
    Server response to the request

Definition at line 195 of file Ftp.cpp.
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Parameters</th>
<th>Returns</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ftp::Response sf::Ftp::DeleteDirectory ( const std::string &amp; Name )</td>
<td>Remove an existing directory.</td>
<td>Name : Name of the directory to remove</td>
<td>Server response to the request</td>
<td>Definition at line 318 of file Ftp.cpp.</td>
</tr>
<tr>
<td>Ftp::Response sf::Ftp::DeleteFile ( const std::string &amp; Name )</td>
<td>Remove an existing file.</td>
<td>Name : File to remove</td>
<td>Server response to the request</td>
<td>Definition at line 340 of file Ftp.cpp.</td>
</tr>
<tr>
<td>Ftp::Response sf::Ftp::Disconnect ( )</td>
<td>Close the connection with FTP server.</td>
<td></td>
<td>Server response to the request</td>
<td>Definition at line 231 of file Ftp.cpp.</td>
</tr>
</tbody>
</table>
**Ftp::Response** `sf::Ftp::Download ( const std::string & DistantFile, const std::string & DestPath, TransferMode Mode = Binary )`

Download a file from the server.

**Parameters:**
- *DistantFile*: Path of the distant file to download
- *DestPath*: Where to put to file on the local computer
- *Mode*: Transfer mode (binary by default)

**Returns:**
Server response to the request

Definition at line 349 of file *Ftp.cpp*.

**Ftp::ListingResponse** `sf::Ftp::GetDirectoryListing ( const std::string & Directory )`

Get the contents of the given directory (subdirectories and files).

Get the contents of the given directory (subdirectories and files).

**Parameters:**
- *Directory*: Directory to list ("" by default, the current one)

**Returns:**
Server response to the request

Definition at line 264 of file *Ftp.cpp*. 
**Ftp::DirectoryResponse** sf::Ftp::GetWorkingDirectory()

Get the current working directory.

**Returns:**
Server response to the request

Definition at line **254** of file *Ftp.cpp*.

**Ftp::Response** sf::Ftp::KeepAlive()

Send a null command just to prevent from being disconnected.

**Returns:**
Server response to the request

Definition at line **245** of file *Ftp.cpp*.

**Ftp::Response** sf::Ftp::Login (const std::string & *UserName*,
                                 const std::string & *Password*)

Log in using a username and a password.

**Parameters:**
- *UserName* : User name
- *Password* : Password

**Returns:**
Server response to the request

Definition at line **218** of file *Ftp.cpp*.
Ftp::Response sf::Ftp::Login ( )

Log in using anonymous account.

Returns:
Server response to the request

Definition at line 209 of file Ftp.cpp.

Ftp::Response sf::Ftp::MakeDirectory ( const std::string & Name )

Create a new directory.

Parameters:
Name : Name of the directory to create

Returns:
Server response to the request

Definition at line 309 of file Ftp.cpp.

Ftp::Response sf::Ftp::ParentDirectory ( )

Go to the parent directory of the current one.

Returns:
Server response to the request

Definition at line 300 of file Ftp.cpp.

Ftp::Response sf::Ftp::RenameFile ( const std::string & File, const std::string & NewName )
 Rename a file.

**Parameters:**
- *File*: File to rename
- *NewName*: New name

**Returns:**
Server response to the request

Definition at line **327** of file **Ftp.cpp**.

```cpp
Ftp::Response sf::Ftp::Upload ( const std::string & LocalFile,
                               const std::string & DestPath,
                               TransferMode Mode = Binary
)
```

Upload a file to the server.

**Parameters:**
- *LocalFile*: Path of the local file to upload
- *DestPath*: Where to put to file on the server
- *Mode*: Transfer mode (binary by default)

**Returns:**
Server response to the request

Definition at line **396** of file **Ftp.cpp**.

The documentation for this class was generated from the following files:
• Ftp.hpp
• Ftp.cpp

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
sf::Ftp::DirectoryResponse
sf::Ftp::Ftp::DirectoryResponse Class Reference

Specialization of FTP response returning a directory. More...

#include <Ftp.hpp>

Inheritance diagram for sf::Ftp::Ftp::DirectoryResponse:

List of all members.
Public Types

```c
enum Status {
    RestartMarkerReply = 110,
    ServiceReadySoon = 120,
    DataConnectionAlreadyOpened = 125,
    OpeningDataConnection = 150,
    Ok = 200,
    PointlessCommand = 202,
    SystemStatus = 211,
    DirectoryStatus = 212,
    FileStatus = 213,
    HelpMessage = 214,
    SystemType = 215,
    ServiceReady = 220,
    ClosingConnection = 221,
    DataConnectionOpened = 225,
    ClosingDataConnection = 226,
    EnteringPassiveMode = 227,
    LoggedIn = 230,
    FileActionOk = 250,
    DirectoryOk = 257,
    NeedPassword = 331,
    NeedAccountToLogIn = 332,
    NeedInformation = 350,
    ServiceUnavailable = 421,
    DataConnectionUnavailable = 425,
    TransferAborted = 426,
    FileActionAborted = 450,
    LocalError = 451,
    InsufficientStorageSpace = 452,
    CommandUnknown = 500,
    ParametersUnknown = 501,
    CommandNotImplemented = 502,
    BadCommandSequence = 503,
    ParameterNotImplemented = 504,
    NotLoggedln = 530,
    NeedAccountToStore = 532,
    FileUnavailable = 550,
    PageTypeUnknown = 551,
    NotEnoughMemory = 552,
    FileNameNotAllowed = 553,
    InvalidResponse = 1000,
    ConnectionFailed = 1001,
    ConnectionClosed = 1002,
    InvalidFile = 1003
};
```

Enumerate all the valid status codes returned in a FTP response. More...
Public Member Functions

<table>
<thead>
<tr>
<th>Public Member Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DirectoryResponse (Response Resp)</td>
<td>Default constructor.</td>
</tr>
<tr>
<td>const std::string &amp; GetDirectory () const</td>
<td>Get the directory returned in the response.</td>
</tr>
<tr>
<td>bool IsOk () const</td>
<td>Convenience function to check if the response status code means a success.</td>
</tr>
<tr>
<td>Status GetStatus () const</td>
<td>Get the response status code.</td>
</tr>
<tr>
<td>const std::string &amp; GetMessage () const</td>
<td>Get the full message contained in the response.</td>
</tr>
</tbody>
</table>
Detailed Description

Specialization of FTP response returning a directory.

Definition at line 182 of file Ftp.hpp.
Member Enumeration Documentation

<table>
<thead>
<tr>
<th>Enumerated Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>sf::Ftp::Response::Status</em> [inherited]</td>
<td>Enumerate all the valid status codes returned in a FTP response.</td>
</tr>
</tbody>
</table>

**Enumerator:**

- **RestartMarkerReply**: Restart marker reply.
- **ServiceReadySoon**: Service ready in N minutes.
- **DataConnectionAlreadyOpened**: Data connection already opened, transfer starting.
- **OpeningDataConnection**: File status ok, about to open data connection.
- **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 an official system name from the list in the Assigned Numbers document.
- **ServiceReady**: Service ready for new user.
- **ClosingConnection**: Service closing control connection.
- **DataConnectionOpened**: Data connection open, no transfer in progress.
- **ClosingDataConnection**: Closing data connection, requested file action successful.
- **EnteringPassiveMode**: Entering passive mode.
- **LoggedIn**: User logged in, proceed. Logged out if appropriate.
- **FileActionOk**: Requested file action ok.
- **DirectoryOk**: PATHNAME created.
- **NeedPassword**: User name ok, need password.
<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NeedAccountToLogIn</td>
<td>Need account for login.</td>
</tr>
<tr>
<td>NeedInformation</td>
<td>Requested file action pending further information.</td>
</tr>
<tr>
<td>ServiceUnavailable</td>
<td>Service not available, closing control connection.</td>
</tr>
<tr>
<td>DataConnectionUnavailable</td>
<td>Can't open data connection.</td>
</tr>
<tr>
<td>TransferAborted</td>
<td>Connection closed, transfer aborted.</td>
</tr>
<tr>
<td>FileActionAborted</td>
<td>Requested file action not taken.</td>
</tr>
<tr>
<td>LocalError</td>
<td>Requested action aborted, local error in processing.</td>
</tr>
<tr>
<td>InsufficientStorageSpace</td>
<td>Requested action not taken; insufficient storage space in system, file unavailable.</td>
</tr>
<tr>
<td>CommandUnknown</td>
<td>Syntax error, command unrecognized.</td>
</tr>
<tr>
<td>ParametersUnknown</td>
<td>Syntax error in parameters or arguments.</td>
</tr>
<tr>
<td>CommandNotImplemented</td>
<td>Command not implemented.</td>
</tr>
<tr>
<td>BadCommandSequence</td>
<td>Bad sequence of commands.</td>
</tr>
<tr>
<td>ParameterNotImplemented</td>
<td>Command not implemented for that parameter.</td>
</tr>
<tr>
<td>NotLoggedIn</td>
<td>Not logged in.</td>
</tr>
<tr>
<td>NeedAccountToStore</td>
<td>Need account for storing files.</td>
</tr>
<tr>
<td>FileUnavailable</td>
<td>Requested action not taken, file unavailable.</td>
</tr>
<tr>
<td>PageTypeUnknown</td>
<td>Requested action aborted, page type unknown.</td>
</tr>
<tr>
<td>NotEnoughMemory</td>
<td>Requested file action aborted, exceeded storage allocation.</td>
</tr>
<tr>
<td>FilenameNotAllowed</td>
<td>Requested action not taken, file name not allowed.</td>
</tr>
<tr>
<td>InvalidResponse</td>
<td><strong>Response</strong> is not a valid FTP one.</td>
</tr>
<tr>
<td>ConnectionFailed</td>
<td>Connection with server failed.</td>
</tr>
<tr>
<td>ConnectionClosed</td>
<td>Connection with server closed.</td>
</tr>
<tr>
<td>InvalidFile</td>
<td>Invalid file to upload / download.</td>
</tr>
</tbody>
</table>

Definition at line 74 of file *Ftp.hpp*. 
sf::Ftp::Ftp::DirectoryResponse::DirectoryResponse (Ftp::Response Resp)

Default constructor.

**Parameters:**

Resp : Source response

Definition at line 123 of file Ftp.cpp.
### Member Function Documentation

**const std::string & sf::Ftp::Ftp::DirectoryResponse::GetDirectory ( ) const**

Get the directory returned in the response.

**Returns:**
Directory name

Definition at line 139 of file *Ftp.cpp*.

**const std::string & sf::Ftp::Ftp::Response::GetMessage ( ) const [inherited]**

Get the full message contained in the response.

**Returns:**
The response message

Definition at line 114 of file *Ftp.cpp*.

**Ftp::Response::Status sf::Ftp::Ftp::Response::GetStatus ( ) const [inherited]**

Get the response status code.

**Returns:**
Status code

Definition at line 105 of file *Ftp.cpp*.

**bool sf::Ftp::Ftp::Response::IsOk ( ) const [inherited]**
Convenience function to check if the response status code means a success.

Convenience function to check if the response status code means a success.

**Returns:**
True if status is success (code < 400)

Definition at line 96 of file **Ftp.cpp**.

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

- **Ftp.hpp**
- **Ftp.cpp**
sf::Ftp::ListingResponse
sf::Ftp::Ftp::ListingResponse Class Reference

Specialization of FTP response returning a filename listing.

More...

#include <Ftp.hpp>

Inheritance diagram for sf::Ftp::Ftp::ListingResponse:

List of all members.
Public Types

```cpp
enum Status {
    RestartMarkerReply = 110,
    ServiceReadySoon = 120,
    OpeningDataConnectionAlreadyOpened = 125,
    OpeningDataConnection = 150,
    Ok = 200,
    PointlessCommand = 202,
    SystemStatus = 211,
    DirectoryStatus = 212,
    FileStatus = 213,
    HelpMessage = 214,
    SystemType = 215,
    ServiceReady = 220,
    ClosingConnection = 221,
    DataConnectionOpened = 225,
    ClosingDataConnection = 226,
    EnteringPassiveMode = 227,
    LoggedIn = 230,
    FileActionOk = 250,
    DirectoryOk = 257,
    NeedPassword = 331,
    NeedAccountToLogIn = 332,
    NeedInformation = 350,
    ServiceUnavailable = 421,
    DataConnectionUnavailable = 425,
    TransferAborted = 426,
    FileActionAborted = 450,
    LocalError = 451,
    InsufficientStorageSpace = 452,
    CommandUnknown = 500,
    ParametersUnknown = 501,
    CommandNotImplemented = 502,
    BadCommandSequence = 503,
    ParameterNotImplemented = 504,
    NotLoggedln = 530,
    NeedAccountToStore = 532,
    FileUnavailable = 550,
    PageTypeUnknown = 551,
    NotEnoughMemory = 552,
    FileNameNotAllowed = 553,
    InvalidResponse = 1000,
    ConnectionFailed = 1001,
    ConnectionClosed = 1002,
    InvalidFile = 1003
}
```

Enumerate all the valid status codes returned in a FTP response. More...
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ListingResponse</strong> (Response Resp, const std::vector&lt; char &gt; &amp;Data)</td>
<td>Default constructor.</td>
</tr>
<tr>
<td>std::size_t getCount() const</td>
<td>Get the number of filenames in the listing.</td>
</tr>
<tr>
<td>const std::string &amp; GetFilename(const std::size_t Index) const</td>
<td>Get the Index-th filename in the directory.</td>
</tr>
<tr>
<td>bool IsOk() const</td>
<td>Convenience function to check if the response status code means a success.</td>
</tr>
<tr>
<td>Status GetStatus() const</td>
<td>Get the response status code.</td>
</tr>
<tr>
<td>const std::string &amp; GetMessage() const</td>
<td>Get the full message contained in the response.</td>
</tr>
</tbody>
</table>
Detailed Description

Specialization of FTP response returning a filename listing.

Definition at line 214 of file Ftp.hpp.
## Member Enumeration Documentation

<table>
<thead>
<tr>
<th>Enum</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>enum sf::Ftp::Response::Status</strong> [inherited]</td>
<td>Enumerate all the valid status codes returned in a FTP response.</td>
</tr>
</tbody>
</table>

### Enumerator:

- **RestartMarkerReply**: Restart marker reply.
- **ServiceReadySoon**: Service ready in N minutes.
- **DataConnectionAlreadyOpened**: Data connection already opened, transfer starting.
- **OpeningDataConnection**: File status ok, about to open data connection.
- **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 an official system name from the list in the Assigned Numbers document.
- **ServiceReady**: Service ready for new user.
- **ClosingConnection**: Service closing control connection.
- **DataConnectionOpened**: Data connection open, no transfer in progress.
- **ClosingDataConnection**: Closing data connection, requested file action successful.
- **EnteringPassiveMode**: Entering passive mode.
- **Loggedln**: User logged in, proceed. Logged out if appropriate.
- **FileActionOk**: Requested file action ok.
- **DirectoryOk**: PATHNAME created.
- **NeedPassword**: User name ok, need password.
<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NeedAccountToLogIn</td>
<td>Need account for login.</td>
</tr>
<tr>
<td>NeedInformation</td>
<td>Requested file action pending further information.</td>
</tr>
<tr>
<td>ServiceUnavailable</td>
<td>Service not available, closing control connection.</td>
</tr>
<tr>
<td>DataConnectionUnavailable</td>
<td>Can't open data connection.</td>
</tr>
<tr>
<td>TransferAborted</td>
<td>Connection closed, transfer aborted.</td>
</tr>
<tr>
<td>FileActionAborted</td>
<td>Requested file action not taken.</td>
</tr>
<tr>
<td>LocalError</td>
<td>Requested action aborted, local error in processing.</td>
</tr>
<tr>
<td>InsufficientStorageSpace</td>
<td>Requested action not taken; insufficient storage space in system, file unavailable.</td>
</tr>
<tr>
<td>CommandUnknown</td>
<td>Syntax error, command unrecognized.</td>
</tr>
<tr>
<td>ParametersUnknown</td>
<td>Syntax error in parameters or arguments.</td>
</tr>
<tr>
<td>CommandNotImplemented</td>
<td>Command not implemented.</td>
</tr>
<tr>
<td>BadCommandSequence</td>
<td>Bad sequence of commands.</td>
</tr>
<tr>
<td>ParameterNotImplemented</td>
<td>Command not implemented for that parameter.</td>
</tr>
<tr>
<td>NotLoggedln</td>
<td>Not logged in.</td>
</tr>
<tr>
<td>NeedAccountToStore</td>
<td>Need account for storing files.</td>
</tr>
<tr>
<td>FileUnavailable</td>
<td>Requested action not taken, file unavailable.</td>
</tr>
<tr>
<td>PageTypeUnknown</td>
<td>Requested action aborted, page type unknown.</td>
</tr>
<tr>
<td>NotEnoughMemory</td>
<td>Requested file action aborted, exceeded storage allocation.</td>
</tr>
<tr>
<td>FilenameNotAllowed</td>
<td>Requested action not taken, file name not allowed.</td>
</tr>
<tr>
<td>InvalidResponse</td>
<td><strong>Response</strong> is not a valid FTP one.</td>
</tr>
<tr>
<td>ConnectionFailed</td>
<td>Connection with server failed.</td>
</tr>
<tr>
<td>ConnectionClosed</td>
<td>Connection with server closed.</td>
</tr>
<tr>
<td>InvalidFile</td>
<td>Invalid file to upload / download.</td>
</tr>
</tbody>
</table>

Definition at line 74 of file Ftp.hpp.
**Constructor & Destructor Documentation**

```cpp
sf::Ftp::Ftp::ListingResponse::ListingResponse ( Ftp::Response const std::vector< char > & )
```

Default constructor.

**Parameters:**
- **Resp**: Source response
- **Data**: Data containing the raw listing

Definition at line 148 of file Ftp.cpp.
### Member Function Documentation

#### `std::size_t sf::Ftp::Ftp::ListingResponse::GetCount()` const

Get the number of filenames in the listing.

**Returns:**
Total number of filenames

Definition at line 168 of file `Ftp.cpp`.

#### `const std::string & sf::Ftp::Ftp::ListingResponse::GetFilename(std::size_t)`

Get the Index-th filename in the directory.

**Parameters:**

Index : Index of the filename to get

**Returns:**
Index-th filename

Definition at line 177 of file `Ftp.cpp`.

#### `const std::string & sf::Ftp::Ftp::Response::GetMessage()` const [inherited]

Get the full message contained in the response.

**Returns:**

The response message

Definition at line 114 of file `Ftp.cpp`. 
Ftp::Response::Status sf::Ftp::Ftp::Response::GetStatus ( ) const [inherited]

Get the response status code.

**Returns:**
Status code

Definition at line 105 of file Ftp.cpp.

bool sf::Ftp::Ftp::Response::IsOk ( ) const [inherited]

Convenience function to check if the response status code means a success.

Convenience function to check if the response status code means a success.

**Returns:**
True if status is success (code < 400)

Definition at line 96 of file Ftp.cpp.

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

- **Ftp.hpp**
- **Ftp.cpp**

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
sf::Ftp::Response
sf::Ftp::Ftp::Response Class Reference

This class wraps a FTP response, which is basically:

- a status code
- a message.

More...

#include <Ftp.hpp>

Inheritance diagram for sf::Ftp::Ftp::Response:

List of all members.
Public Types

```c
enum Status {
    RestartMarkerReply = 110,
    ServiceReadySoon = 120,
    DataConnectionAlreadyOpened = 125,
    OpeningDataConnection = 150,
    Ok = 200,
    PointlessCommand = 202,
    SystemStatus = 211,
    DirectoryStatus = 212,
    FileStatus = 213,
    HelpMessage = 214,
    SystemType = 215,
    ServiceReady = 220,
    ClosingConnection = 221,
    DataConnectionOpened = 225,
    ClosingDataConnection = 226,
    EnteringPassiveMode = 227,
   LoggedIn = 230,
    FileActionOk = 250,
    DirectoryOk = 257,
    NeedPassword = 331,
    NeedAccountToLogIn = 332,
    NeedInformation = 350,
    ServiceUnavailable = 421,
    DataConnectionUnavailable = 425,
    TransferAborted = 426,
    FileActionAborted = 450,
    LocalError = 451,
    InsufficientStorageSpace = 452,
    CommandUnknown = 500,
    ParametersUnknown = 501,
    CommandNotImplemented = 502,
    BadCommandSequence = 503,
    ParameterNotImplemented = 504,
    NotLoggedln = 530,
    NeedAccountToStore = 532,
    FileUnavailable = 550,
    PageTypeUnknown = 551,
    NotEnoughMemory = 552,
    FilenameNotAllowed = 553,
    InvalidResponse = 1000,
    ConnectionFailed = 1001,
    ConnectionClosed = 1002,
    InvalidFile = 1003
}
```

Enumerate all the valid status codes returned in a FTP response. More...
# Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Response</code></td>
<td>(Status Code=InvalidResponse, const std::string &amp;Message=&quot;&quot;) Default constructor.</td>
</tr>
<tr>
<td><code>bool IsOk()</code></td>
<td>Convenience function to check if the response status code means a success.</td>
</tr>
<tr>
<td><code>Status GetStatus()</code></td>
<td>Get the response status code.</td>
</tr>
<tr>
<td><code>const std::string &amp; GetMessage()</code></td>
<td>Get the full message contained in the response.</td>
</tr>
</tbody>
</table>
Detailed Description

This class wraps a FTP response, which is basically:

- a status code
- a message.

Definition at line 66 of file Ftp.hpp.
Member Enumeration Documentation

<table>
<thead>
<tr>
<th>Enum</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sf::Ftp::Response::Status</code></td>
<td>Enumerate all the valid status codes returned in a FTP response.</td>
</tr>
<tr>
<td><code>RestartMarkerReply</code></td>
<td>Restart marker reply.</td>
</tr>
<tr>
<td><code>ServiceReadySoon</code></td>
<td>Service ready in N minutes.</td>
</tr>
<tr>
<td><code>DataConnectionAlreadyOpened</code></td>
<td>Data connection already opened, transfer starting.</td>
</tr>
<tr>
<td><code>OpeningDataConnection</code></td>
<td>File status ok, about to open data connection.</td>
</tr>
<tr>
<td><code>Ok</code></td>
<td>Command ok.</td>
</tr>
<tr>
<td><code>PointlessCommand</code></td>
<td>Command not implemented.</td>
</tr>
<tr>
<td><code>SystemStatus</code></td>
<td>System status, or system help reply.</td>
</tr>
<tr>
<td><code>DirectoryStatus</code></td>
<td>Directory status.</td>
</tr>
<tr>
<td><code>FileStatus</code></td>
<td>File status.</td>
</tr>
<tr>
<td><code>HelpMessage</code></td>
<td>Help message.</td>
</tr>
<tr>
<td><code>SystemType</code></td>
<td>NAME system type, where NAME is an official system name from the list in the Assigned Numbers document.</td>
</tr>
<tr>
<td><code>ServiceReady</code></td>
<td>Service ready for new user.</td>
</tr>
<tr>
<td><code>ClosingConnection</code></td>
<td>Service closing control connection.</td>
</tr>
<tr>
<td><code>DataConnectionOpened</code></td>
<td>Data connection open, no transfer in progress.</td>
</tr>
<tr>
<td><code>ClosingDataConnection</code></td>
<td>Closing data connection, requested file action successful.</td>
</tr>
<tr>
<td><code>EnteringPassiveMode</code></td>
<td>Entering passive mode.</td>
</tr>
<tr>
<td><code>Loggedln</code></td>
<td>User logged in, proceed. Logged out if appropriate.</td>
</tr>
<tr>
<td><code>FileActionOk</code></td>
<td>Requested file action ok.</td>
</tr>
<tr>
<td><code>DirectoryOk</code></td>
<td>PATHNAME created.</td>
</tr>
<tr>
<td><code>NeedPassword</code></td>
<td>User name ok, need password.</td>
</tr>
<tr>
<td>Error Code</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>NeedAccountToLogIn</td>
<td>Need account for login.</td>
</tr>
<tr>
<td>NeedInformation</td>
<td>Requested file action pending further information.</td>
</tr>
<tr>
<td>ServiceUnavailable</td>
<td>Service not available, closing control connection.</td>
</tr>
<tr>
<td>DataConnectionUnavailable</td>
<td>Can't open data connection.</td>
</tr>
<tr>
<td>TransferAborted</td>
<td>Connection closed, transfer aborted.</td>
</tr>
<tr>
<td>FileActionAborted</td>
<td>Requested file action not taken.</td>
</tr>
<tr>
<td>LocalError</td>
<td>Requested action aborted, local error in processing.</td>
</tr>
<tr>
<td>InsufficientStorageSpace</td>
<td>Requested action not taken; insufficient storage space in system, file unavailable.</td>
</tr>
<tr>
<td>CommandUnknown</td>
<td>Syntax error, command unrecognized.</td>
</tr>
<tr>
<td>ParametersUnknown</td>
<td>Syntax error in parameters or arguments.</td>
</tr>
<tr>
<td>CommandNotImplemented</td>
<td>Command not implemented.</td>
</tr>
<tr>
<td>BadCommandSequence</td>
<td>Bad sequence of commands.</td>
</tr>
<tr>
<td>ParameterNotImplemented</td>
<td>Command not implemented for that parameter.</td>
</tr>
<tr>
<td>NotLoggedIn</td>
<td>Not logged in.</td>
</tr>
<tr>
<td>NeedAccountToStore</td>
<td>Need account for storing files.</td>
</tr>
<tr>
<td>FileUnavailable</td>
<td>Requested action not taken, file unavailable.</td>
</tr>
<tr>
<td>PageTypeUnknown</td>
<td>Requested action aborted, page type unknown.</td>
</tr>
<tr>
<td>NotEnoughMemory</td>
<td>Requested file action aborted, exceeded storage allocation.</td>
</tr>
<tr>
<td>FilenameNotAllowed</td>
<td>Requested action not taken, file name not allowed.</td>
</tr>
<tr>
<td>InvalidResponse</td>
<td>Response is not a valid FTP one.</td>
</tr>
<tr>
<td>ConnectionFailed</td>
<td>Connection with server failed.</td>
</tr>
<tr>
<td>ConnectionClosed</td>
<td>Connection with server closed.</td>
</tr>
<tr>
<td>InvalidFile</td>
<td>Invalid file to upload / download.</td>
</tr>
</tbody>
</table>

Definition at line 74 of file Ftp.hpp.
Constructor & Destructor Documentation

sf::Ftp::Ftp::Response::Response ( Status Code = InvalidResponse const std::string & Message = "" )

Default constructor.

**Parameters:**

- **Code**: Response status code (InvalidResponse by default)
- **Message**: Response message (empty by default)

Definition at line 84 of file Ftp.cpp.
# Member Function Documentation

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Returns</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>const std::string &amp; sf::Ftp::Ftp::Response::GetMessage ( ) const</code></td>
<td>Get the full message contained in the response.</td>
<td>The response message</td>
<td>line 114</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Ftp.cpp</td>
</tr>
<tr>
<td><code>Ftp::Response::Status sf::Ftp::Ftp::Response::getStatus ( ) const</code></td>
<td>Get the response status code.</td>
<td>Status code</td>
<td>line 105</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Ftp.cpp</td>
</tr>
<tr>
<td><code>bool sf::Ftp::Ftp::Response::IsOk ( ) const</code></td>
<td>Convenience function to check if the response status code means a success.</td>
<td>True if status is success (code &lt; 400)</td>
<td>line 96</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Ftp.cpp</td>
</tr>
</tbody>
</table>
The documentation for this class was generated from the following files:

- Ftp.hpp
- Ftp.cpp
sf::Glyph
sf::Glyph Class Reference

Structure describing a glyph (a visual character). More...

#include <Glyph.hpp>

List of all members.
## Public Member Functions

<table>
<thead>
<tr>
<th><strong>Glyph ()</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Default constructor.</td>
</tr>
</tbody>
</table>
## Public Attributes

<table>
<thead>
<tr>
<th>Type</th>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td><strong>Advance</strong></td>
<td>Offset to move horizontally to the next character.</td>
</tr>
<tr>
<td>IntRect</td>
<td><strong>Rectangle</strong></td>
<td>Bounding rectangle of the glyph, in relative coordinates.</td>
</tr>
<tr>
<td>FloatRect</td>
<td><strong>TexCoords</strong></td>
<td>Texture coordinates of the glyph inside the bitmap font.</td>
</tr>
</tbody>
</table>
Detailed Description

Structure describing a glyph (a visual character).

Definition at line 40 of file Glyph.hpp.
## Constructor & Destructor Documentation

<table>
<thead>
<tr>
<th>sf::Glyph::Glyph( ) [inline]</th>
</tr>
</thead>
</table>

Default constructor.

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

## Member Data Documentation

<table>
<thead>
<tr>
<th>Type</th>
<th>Member Variable</th>
<th>Description</th>
<th>Definition Line</th>
<th>File</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>sf::Glyph::Advance</td>
<td>Offset to move horizontally to the next character.</td>
<td>53</td>
<td>Glyph.hpp</td>
</tr>
<tr>
<td>IntRect</td>
<td>sf::Glyph::Rectangle</td>
<td>Bounding rectangle of the glyph, in relative coordinates.</td>
<td>54</td>
<td>Glyph.hpp</td>
</tr>
<tr>
<td>FloatRect</td>
<td>sf::Glyph::TexCoords</td>
<td>Texture coordinates of the glyph inside the bitmap font.</td>
<td>55</td>
<td>Glyph.hpp</td>
</tr>
</tbody>
</table>

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

- **Glyph.hpp**

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by **doxygen 1.5.2** ::
sf::Http
sf::Http Class Reference

This class provides methods for manipulating the HTTP protocol (described in RFC 1945). [More...]

#include <Http.hpp>

Inheritance diagram for sf::Http:

![Inheritance Diagram](image)

[List of all members.]
### Classes

<table>
<thead>
<tr>
<th>class</th>
<th>Request</th>
</tr>
</thead>
</table>
|       | *This class wraps an HTTP request, which is basically:*
|       |   - a header with a method, a target URI, and a set of field/value pairs
|       |   - an optional body (for POST requests). |

More...

<table>
<thead>
<tr>
<th>class</th>
<th>Response</th>
</tr>
</thead>
</table>
|       | *This class wraps an HTTP response, which is basically:*
|       |   - a header with a status code and a set of field/value pairs
|       |   - a body (the content of the requested resource). |

More...
### Public Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Http ()</strong></td>
<td><em>Default constructor.</em></td>
</tr>
<tr>
<td><strong>Http (const std::string &amp;Host, unsigned short Port=0)</strong></td>
<td><em>Construct the Http instance with the target host.</em></td>
</tr>
<tr>
<td><strong>void SetHost (const std::string &amp;Host, unsigned short Port=0)</strong></td>
<td><em>Set the target host.</em></td>
</tr>
<tr>
<td><strong>Response SendRequest (const Request &amp;Req, float Timeout=0.f)</strong></td>
<td><em>Send a HTTP request and return the server's response.</em></td>
</tr>
</tbody>
</table>
**Detailed Description**

This class provides methods for manipulating the HTTP protocol (described in RFC 1945).

It can connect to a website, get its files, send requests, etc.

Definition at line 45 of file `Http.hpp`. 
## sf::Http::Http

*sf::Http::Http ( )*

Default constructor.

Definition at line 307 of file *Http.cpp*.

*sf::Http::Http ( const std::string & *Host*, unsigned short *Port* = 0 )*

Construct the *Http* instance with the target host.

**Parameters:**

- *Host* : Web server to connect to
- *Port* : Port to use for connection (0 by default -- use the standard port of the protocol used)

Definition at line 318 of file *Http.cpp*. 
Member Function Documentation

Http::Response sf::Http::SendRequest ( const Request & Req, 
float 
Timeout = 0.f)

Send a HTTP request and return the server's response.
You must be connected to a host before sending requests. Any missing mandatory header field will be added with an appropriate value. Warning: this function waits for the server's response and may not return instantly; use a thread if you don't want to block your application.

Parameters:
- Req : Request to send
- Timeout : Maximum time to wait, in seconds (0 by default, means no timeout)

Returns:
- Server's response

You must be connected to a host before sending requests. Any missing mandatory header field will be added with an appropriate value. Warning: this function waits for the server's response and may not return instantly; use a thread if you don't want to block your application.

Definition at line 366 of file Http.cpp.

void sf::Http::SetHost ( const std::string & Host, 
unsigned short Port = 0)
Set the target host.

**Parameters:**

- **Host**: Web server to connect to
- **Port**: Port to use for connection (0 by default -- use the standard port of the protocol used)

Definition at line 327 of file **Http.cpp**.

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

- **Http.hpp**
- **Http.cpp**
sf::Http::Request
sf::Http::Http::Request Class Reference

This class wraps an HTTP request, which is basically:

- a header with a method, a target URI, and a set of field/value pairs
- an optional body (for POST requests).

More...

#include <Http.hpp>

List of all members.
### Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>Method {</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Get,</td>
</tr>
<tr>
<td></td>
<td>Post,</td>
</tr>
<tr>
<td></td>
<td>Head</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
</tbody>
</table>

`Enumerate the available HTTP methods for a request. More...`
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Request</strong> (Method RequestMethod=Get, const std::string &amp;URI=&quot;/&quot;, const std::string &amp;Body=&quot;&quot;)</td>
<td></td>
</tr>
<tr>
<td>Default constructor.</td>
<td></td>
</tr>
<tr>
<td>void <strong>SetField</strong> (const std::string &amp;Field, const std::string &amp;Value)</td>
<td></td>
</tr>
<tr>
<td>Set the value of a field; the field is added if it doesn't exist.</td>
<td></td>
</tr>
<tr>
<td>void <strong>SetMethod</strong> (Method RequestMethod)</td>
<td></td>
</tr>
<tr>
<td>Set the request method.</td>
<td></td>
</tr>
<tr>
<td>void <strong>SetURI</strong> (const std::string &amp;URI)</td>
<td></td>
</tr>
<tr>
<td>Set the target URI of the request.</td>
<td></td>
</tr>
<tr>
<td>void <strong>SetHttpVersion</strong> (unsigned int Major, unsigned int Minor)</td>
<td></td>
</tr>
<tr>
<td>Set the HTTP version of the request.</td>
<td></td>
</tr>
<tr>
<td>void <strong>SetBody</strong> (const std::string &amp;Body)</td>
<td></td>
</tr>
<tr>
<td>Set the body of the request.</td>
<td></td>
</tr>
</tbody>
</table>
Friends

class  Http
Detailed Description

This class wraps an HTTP request, which is basically:

- a header with a method, a target URI, and a set of field/value pairs
- an optional body (for POST requests).

Definition at line 54 of file Http.hpp.
# Member Enumeration Documentation

<table>
<thead>
<tr>
<th>enum sf::Http::Request::Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enumerate the available HTTP methods for a request.</td>
</tr>
</tbody>
</table>

**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.

Definition at line 61 of file **Http.hpp**.
## Constructor & Destructor Documentation

```cpp
sf::Http::Http::Request::Request ( RequestMethod RequestMethod = Get,  
const std::string & URI = "/",  
const std::string & Body = ""
)
```

Default constructor.

**Parameters:**
- `RequestMethod` : Method to use for the request (Get by default)
- `URI` : Target URI ("/" by default -- index page)
- `Body` : Content of the request's body (empty by default)

Definition at line 56 of file `Http.cpp`.

void sf::Http::Http::Request::SetBody ( const std::string & Body )

Set the body of the request.

This parameter is optional and makes sense only for POST requests. This parameter is empty by default

**Parameters:**

*Body* : Content of the request body

This parameter is optional and makes sense only for POST requests. This parameter is empty by default

Definition at line 114 of file Http.cpp.

void sf::Http::Http::Request::SetField ( const std::string & Field, const std::string & Value )

Set the value of a field; the field is added if it doesn't exist.

**Parameters:**

*Field* : Name of the field to set (case-insensitive)
*Value* : Value of the field

Definition at line 68 of file Http.cpp.

void sf::Http::Http::Request::SetHttpVersion ( unsigned int Major, unsigned int Minor )


Set the HTTP version of the request.

This parameter is 1.0 by default

**Parameters:**

- **Major**: Major version number
- **Minor**: Minor version number

This parameter is 1.0 by default

Definition at line 102 of file **Http.cpp**.

```cpp
void sf::Http::Http::Request::SetMethod ( Method RequestMethod )
```

Set the request method.

This parameter is **Http::Request::Get** by default

**Parameters:**

- **RequestMethod**: Method to use for the request

This parameter is Get by default

Definition at line 78 of file **Http.cpp**.

```cpp
void sf::Http::Http::Request::SetURI ( const std::string & URI )
```

Set the target URI of the request.

This parameter is "/" by default

**Parameters:**

- **URI**: URI to request, local to the host
This parameter is "/" by default

Definition at line 88 of file Http.cpp.

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

- **Http.hpp**
- **Http.cpp**
sf::Http::Response
sf::Http::Http::Response Class Reference

This class wraps an HTTP response, which is basically:

- a header with a status code and a set of field/value pairs
- a body (the content of the requested resource).

More...

#include <Http.hpp>

List of all members.
**Public Types**

```c
enum Status {
    Ok = 200,
    Created = 201,
    Accepted = 202,
    NoContent = 204,
    MultipleChoices = 300,
    MovedPermanently = 301,
    MovedTemporarily = 302,
    NotModified = 304,
    BadRequest = 400,
    Unauthorized = 401,
    Forbidden = 403,
    NotFound = 404,
    InternalServerError = 500,
    NotImplemented = 501,
    BadGateway = 502,
    ServiceUnavailable = 503,
    InvalidResponse = 1000,
    ConnectionFailed = 1001
}
```

*Enumerate all the valid status codes returned in a HTTP response.* [More...](#)
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Response ()</td>
<td>Default constructor.</td>
</tr>
<tr>
<td>const std::string &amp; GetField(const std::string &amp;Field) const</td>
<td>Get the value of a field.</td>
</tr>
<tr>
<td>Status GetStatus () const</td>
<td>Get the header's status code.</td>
</tr>
<tr>
<td>unsigned int GetMajorHttpVersion () const</td>
<td>Get the major HTTP version number of the response.</td>
</tr>
<tr>
<td>unsigned int GetMinorHttpVersion () const</td>
<td>Get the major HTTP version number of the response.</td>
</tr>
<tr>
<td>const std::string &amp; GetBody () const</td>
<td>Get the body of the response.</td>
</tr>
</tbody>
</table>
Friends

class Http
Detailed Description

This class wraps an HTTP response, which is basically:

- a header with a status code and a set of field/value pairs
- a body (the content of the requested resource).

Definition at line 168 of file Http.hpp.
Member Enumeration Documentation

<table>
<thead>
<tr>
<th>Enum</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enum sf::Http::Response::Status</td>
<td>Enumerate all the valid status codes returned in a HTTP response.</td>
</tr>
<tr>
<td><strong>Ok</strong></td>
<td>Most common code returned when operation was successful.</td>
</tr>
<tr>
<td><strong>Created</strong></td>
<td>The resource has successfully been created.</td>
</tr>
<tr>
<td><strong>Accepted</strong></td>
<td>The request has been accepted, but will be processed later by the server.</td>
</tr>
<tr>
<td><strong>NoContent</strong></td>
<td>Sent when the server didn't send any data in return.</td>
</tr>
<tr>
<td><strong>MultipleChoices</strong></td>
<td>The requested page can be accessed from several locations.</td>
</tr>
<tr>
<td><strong>MovedPermanently</strong></td>
<td>The requested page has permanently moved to a new location.</td>
</tr>
<tr>
<td><strong>MovedTemporarily</strong></td>
<td>The requested page has temporarily moved to a new location.</td>
</tr>
<tr>
<td><strong>NotModified</strong></td>
<td>For conditionnal requests, means the requested page hasn't changed and doesn't need to be refreshed.</td>
</tr>
<tr>
<td><strong>BadRequest</strong></td>
<td>The server couldn't understand the request (syntax error).</td>
</tr>
<tr>
<td><strong>Unauthorized</strong></td>
<td>The requested page needs an authentification to be accessed.</td>
</tr>
<tr>
<td><strong>Forbidden</strong></td>
<td>The requested page cannot be accessed at all, even with authentification.</td>
</tr>
<tr>
<td><strong>NotFound</strong></td>
<td>The requested page doesn't exist.</td>
</tr>
<tr>
<td><strong>InternalServerError</strong></td>
<td>The server encountered an unexpected error.</td>
</tr>
<tr>
<td><strong>NotImplemented</strong></td>
<td>The server doesn't implement a requested feature.</td>
</tr>
<tr>
<td><strong>BadGateway</strong></td>
<td>The gateway server has received an error from the source server.</td>
</tr>
<tr>
<td><strong>ServiceNotAvailable</strong></td>
<td>The server is temporarily unavailable (overloaded, in maintenance, ...).</td>
</tr>
<tr>
<td>InvalidResponse</td>
<td>Response is not a valid HTTP one.</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>ConnectionFailed</td>
<td>Connection with server failed.</td>
</tr>
</tbody>
</table>

Definition at line 176 of file `Http.hpp`. 
sf::Http::Http::Response::Response ( )

Default constructor.

Definition at line 169 of file Http.cpp.
Member Function Documentation

**const std::string & sf::Http::Http::Response::GetBody ( ) const**

Get the body of the response.

The body can 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

The body can 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)

Definition at line 230 of file Http.cpp.

**const std::string & sf::Http::Http::Response::GetField ( const std::string & Field ) const**

Get the value of a field.

**Parameters:**

- **Field**: Name of the field to get (case-insensitive)

**Returns:**
Value of the field, or empty string if not found

Definition at line 181 of file Http.cpp.

```
unsigned int sf::Http::Http::Response::GetMajorHttpVersion ( ) const

Get the major HTTP version number of the response.

Returns:
   Major version number

Definition at line 208 of file Http.cpp.
```

```
unsigned int sf::Http::Http::Response::GetMinorHttpVersion ( ) const

Get the major HTTP version number of the response.

Returns:
   Major version number

Definition at line 217 of file Http.cpp.
```

```
Http::Response::Status sf::Http::Http::Response::GetStatus ( ) const

Get the header's status code.

Returns:
   Header's status code

Definition at line 199 of file Http.cpp.
```
The documentation for this class was generated from the following files:

- **Http.hpp**
- **Http.cpp**

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
sf::Image
sf::Image Class Reference

**Image** is the low-level class for loading and manipulating images. [More...]

```
#include <Image.hpp>
```

Inheritance diagram for sf::Image:

```
< Image >" shape="rect" coords="0,0,139,24">
```

[List of all members.]
**Public Member Functions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Image ()</strong></td>
<td>Default constructor.</td>
</tr>
<tr>
<td><strong>Image (const Image &amp;Copy)</strong></td>
<td>Copy constructor.</td>
</tr>
<tr>
<td><strong>Image (unsigned int Width, unsigned int Height, const Color &amp;Col=Color(0, 0, 0, 255))</strong></td>
<td>Construct an empty image.</td>
</tr>
<tr>
<td>*<em>Image (unsigned int Width, unsigned int Height, const Uint8 <em>Data)</em></em></td>
<td>Construct the image from pixels in memory.</td>
</tr>
<tr>
<td><strong>~Image ()</strong></td>
<td>Destructor.</td>
</tr>
<tr>
<td><strong>bool LoadFromFile (const std::string &amp;Filename)</strong></td>
<td>Load the image from a file.</td>
</tr>
<tr>
<td>*<em>bool LoadFromMemory (const char <em>Data, std::size_t SizeInBytes)</em></em></td>
<td>Load the image from a file in memory.</td>
</tr>
<tr>
<td>*<em>bool LoadFromPixels (unsigned int Width, unsigned int Height, const Uint8 <em>Data)</em></em></td>
<td>Load the image directly from an array of pixels.</td>
</tr>
<tr>
<td><strong>bool SaveToFile (const std::string &amp;Filename) const</strong></td>
<td>Save the content of the image to a file.</td>
</tr>
<tr>
<td><strong>bool Create (unsigned int Width, unsigned int Height, Color Col=Color(0, 0, 0, 255))</strong></td>
<td>Create an empty image.</td>
</tr>
<tr>
<td><strong>void CreateMaskFromColor (Color ColorKey, Uint8 Alpha=0)</strong></td>
<td>Create transparency mask from a specified color key.</td>
</tr>
<tr>
<td><strong>void Copy (const Image &amp;Source, unsigned int DestX, unsigned int DestY, const IntRect &amp;SourceRect=IntRect(0, 0, 0, 0), bool ApplyAlpha=false)</strong></td>
<td>Copy pixels from another image onto this one.</td>
</tr>
<tr>
<td><strong>bool CopyScreen (RenderWindow &amp;Window, const IntRect &amp;SourceRect=IntRect(0, 0, 0, 0))</strong></td>
<td>Create the image from the current contents of the given window.</td>
</tr>
<tr>
<td><strong>void SetPixel (unsigned int X, unsigned int Y, const Color &amp;Col)</strong></td>
<td>Change the color of a pixel.</td>
</tr>
<tr>
<td><strong>const Color &amp; GetPixel (unsigned int X, unsigned int Y) const</strong></td>
<td>Get a pixel from the image.</td>
</tr>
<tr>
<td><strong>const Uint8 * GetPixelsPtr () const</strong></td>
<td>Get a read-only pointer to the array of pixels (RGBA 8 bits integers components) Array size is <code>GetWidth() x GetHeight() x 4</code> This pointer becomes invalid if you reload or resize the image.</td>
</tr>
<tr>
<td><strong>void Bind () const</strong></td>
<td>Bind the image for rendering.</td>
</tr>
<tr>
<td><strong>void SetSmooth (bool Smooth)</strong></td>
<td>Enable or disable image smooth filter.</td>
</tr>
<tr>
<td><strong>unsigned int GetWidth () const</strong></td>
<td>Return the width of the image.</td>
</tr>
<tr>
<td>Type</td>
<td>Function</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>unsigned int</td>
<td>GetHeight () const</td>
</tr>
<tr>
<td>bool</td>
<td>IsSmooth () const</td>
</tr>
<tr>
<td>FloatRect</td>
<td>GetTexCoords (const IntRect &amp;Rect) const</td>
</tr>
<tr>
<td>Image &amp;</td>
<td>operator= (const Image &amp;Other)</td>
</tr>
<tr>
<td>static unsigned int</td>
<td>GetValidTextureSize (unsigned int Size)</td>
</tr>
<tr>
<td>---------------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td></td>
<td>Get a valid texture size according to hardware support.</td>
</tr>
</tbody>
</table>
Detailed Description

**Image** is the low-level class for loading and manipulating images.

Definition at line 46 of file Image.hpp.
Constructor & Destructor Documentation

sf::Image::Image()

Default constructor.

Definition at line 43 of file Image.cpp.

sf::Image::Image (const Image & Copy)

Copy constructor.

**Parameters:**

*Copy* : instance to copy

Definition at line 60 of file Image.cpp.

sf::Image::Image (unsigned int Width,
unsigned int Height,
const Color & Col = color(0, 0, 0, 255))

Construct an empty image.

**Parameters:**

*Width* : Image width
*Height* : Image height
*Col* : Image color (black by default)

Definition at line 79 of file Image.cpp.
**sf::Image::Image** (unsigned int \textit{Width},
unsigned int \textit{Height},
const Uint8 * \textit{Data})

Construct the image from pixels in memory.

**Parameters:**

- \textit{Width} : Image width
- \textit{Height} : Image height
- \textit{Data} : Pointer to the pixels in memory (assumed format is RGBA)

Definition at line 96 of file *Image.cpp.*

**sf::Image::~Image** ( )

Destructor.

Definition at line 113 of file *Image.cpp.*
## Member Function Documentation

**void sf::Image::Bind ( ) const**

Bind the image for rendering.

Definition at line 473 of file `Image.cpp`.

**void sf::Image::Copy ( const Image & Source, unsigned int DestX, unsigned int DestY, const IntRect & SourceRect = IntRect(0, 0, 0, 0), bool ApplyAlpha = false )**

Copy pixels from another image onto this one.

This function does a slow pixel copy and should only be used at initialization time

**Parameters:**

- **Source**: Source image to copy
- **DestX**: X coordinate of the destination position
- **DestY**: Y coordinate of the destination position
- **SourceRect**: Sub-rectangle of the source image to copy (empty by default - entire image)
- **ApplyAlpha**: Should the copy take in account the source transparency? (false by default)

This function does a slow pixel copy and should only be used at initialization time

Definition at line 270 of file `Image.cpp`. 
bool sf::Image::CopyScreen (RenderWindow & Window, 
const IntRect & SourceRect = IntRect(0, 0, 0))

Create the image from the current contents of the given window.

Create the image from the current contents of the given window.

Parameters:

Window : Window to capture
SourceRect : Sub-rectangle of the screen to copy (empty by default - entire image)

Returns:

True if copy was successful

Definition at line 358 of file Image.cpp.

bool sf::Image::Create (unsigned int Width, 
unsigned int Height, 
Color Col = color(0, 0, 0, 255))

Create an empty image.

Parameters:

Width : Image width
Height : Image height
Col : Image color (black by default)

Returns:
True if creation was successful

Definition at line 222 of file Image.cpp.

```cpp
void sf::Image::CreateMaskFromColor ( Color ColorKey,
                                       Uint8 Alpha = 0
                                  )
```

Create transparency mask from a specified color key.

**Parameters:**
- **ColorKey**: Color to become transparent
- **Alpha**: Alpha value to use for transparent pixels (0 by default)

Definition at line 249 of file Image.cpp.

```cpp
unsigned int sf::Image::GetHeight ( ) const
```

Return the height of the image.

**Returns:**
- Height in pixels

Definition at line 526 of file Image.cpp.

```cpp
const Color & sf::Image::GetPixel ( unsigned int X,
                                     unsigned int Y
                                 ) const
```

Get a pixel from the image.

**Parameters:**
**Returns:**

*Color* of pixel (X, Y)

Definition at line 431 of file *Image.cpp*.

---

```cpp
const Uint8 * sf::Image::GetPixelsPtr() const
```

Get a read-only pointer to the array of pixels (RGBA 8 bits integers components) Array size is *GetWidth()* x *GetHeight()* x 4 This pointer becomes invalid if you reload or resize the image.

Get a read-only pointer to the array of pixels (RGBA 8 bits integers components) Array size is *GetWidth()* x *GetHeight()* x 4 This pointer becomes invalid if you reload or resize the image.

**Returns:**

Const pointer to the array of pixels

Definition at line 453 of file *Image.cpp*.

---

```cpp
FloatRect sf::Image::GetTexCoords(const IntRect & Rect) const
```

Convert a subrect expressed in pixels, into float texture coordinates.

Convert a subrect expressed in pixels, into float texture coordinates.
**Parameters:**  
*Rect*: Sub-rectangle of image to convert

**Returns:**  
Texture coordinates corresponding to the sub-rectangle

Definition at line 545 of file *Image.cpp*.

```cpp
unsigned int sf::Image::GetValidTextureSize(unsigned int Size) static
```

Get a valid texture size according to hardware support.

**Parameters:**  
Size: Size to convert

**Returns:**  
Valid nearest size (greater than or equal to specified size)

Definition at line 560 of file *Image.cpp*.

```cpp
unsigned int sf::Image::GetWidth() const
```

Return the width of the image.

**Returns:**  
Width in pixels

Definition at line 517 of file *Image.cpp*.

```cpp
bool sf::Image::IsSmooth() const
```

Tells whether the smooth filtering is enabled or not.
Returns:
   True if image smoothing is enabled

Definition at line 535 of file Image.cpp.

```cpp
bool sf::Image::LoadFromFile ( const std::string & Filename )
```

Load the image from a file.

**Parameters:**
   
   *Filename* : Path of the image file to load

**Returns:**
   True if loading was successful

Definition at line 123 of file Image.cpp.

```cpp
bool sf::Image::LoadFromMemory ( const char * Data,
                                      std::size_t SizelnBytes )
```

Load the image from a file in memory.

**Parameters:**
   
   *Data* : Pointer to the file data in memory
   *SizelnBytes* : Size of the data to load, in bytes

**Returns:**
   True if loading was successful

Definition at line 145 of file Image.cpp.
Load the image directly from an array of pixels.

**Parameters:**
- **Width**: Image width
- **Height**: Image height
- **Data**: Pointer to the pixels in memory (assumed format is RGBA)

**Returns:**
- True if loading was successful

Definition at line 174 of file `Image.cpp`.

Assignment operator.

**Parameters:**
- **Other**: instance to assign

**Returns:**
- Reference to the image

Definition at line 585 of file `Image.cpp`.

Save the content of the image to a file.
**Parameters:**

*Filename* : Path of the file to save (overwritten if already exist)

**Returns:**

True if saving was successful

Definition at line 209 of file Image.cpp.

```cpp
void sf::Image::SetPixel ( unsigned int X, unsigned int Y, const Color & Col )
```

Change the color of a pixel.

**Parameters:**

* X : X coordinate of pixel in the image
* Y : Y coordinate of pixel in the image
* Col : New color for pixel (X, Y)

Definition at line 408 of file Image.cpp.

```cpp
void sf::Image::SetSmooth ( bool Smooth )
```

Enable or disable image smooth filter.

Enable or disable image smoothing filter.

This parameter is enabled by default

**Parameters:**

* Smooth : True to enable smoothing filter, false to disable it
Definition at line 490 of file Image.cpp.

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

- **Image.hpp**
- **Image.cpp**

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved  ::
 Documentation generated by doxygen 1.5.2  ::
sf::Input
sf::Input Class Reference

Input handles real-time input from keyboard and mouse. More...

#include <Input.hpp>

Inheritance diagram for sf::Input:

List of all members.
Public Member Functions

**Input ()**  
Default constructor.

**bool IsKeyDown (Key::Code KeyCode) const**  
Get the state of a key.

**bool IsMouseButtonButtonDown (Mouse::Button Button) const**  
Get the state of a mouse button.

**bool IsJoystickButtonDown (unsigned int JoyId, unsigned int Button) const**  
Get the state of a joystick button.

**int GetMouseX () const**  
Get the mouse X position.

**int GetMouseY () const**  
Get the mouse Y position.

**float GetJoystickAxis (unsigned int JoyId, Joy::Axis Axis) const**  
Get a joystick axis position.
Detailed Description

**Input** handles real-time input from keyboard and mouse.

Use it instead of events to handle continuous moves and more game-friendly inputs

Definition at line 44 of file *Input.hpp*.
## Constructor & Destructor Documentation

<table>
<thead>
<tr>
<th>sf::Input::Input ( )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default constructor.</td>
</tr>
<tr>
<td>Definition at line 36 of file <code>Input.cpp</code>.</td>
</tr>
</tbody>
</table>
Member Function Documentation

float sf::Input::GetJoystickAxis ( unsigned int JoyId, Joy::Axis Axis ) const

Get a joystick axis position.

**Parameters:**
- **JoyId**: Identifier of the joystick to check (0 or 1)
- **Axis**: Axis to get

**Returns:**
Current axis position, in the range [-100, 100] (except for POV, which is [0, 360])

Definition at line 95 of file Input.cpp.

```cpp
int sf::Input::GetMouseX ( ) const
```

Get the mouse X position.

Get the mouse left position.

**Returns:**
Current mouse left position, relative to owner window

Definition at line 77 of file Input.cpp.

```cpp
int sf::Input::GetMouseY ( ) const
```

Get the mouse Y position.
Get the mouse top position.

**Returns:**
Current mouse top position, relative to owner window

Definition at line 86 of file `Input.cpp`.

```cpp
bool sf::Input::IsJoystickButtonDown(unsigned int JoyId, unsigned int Button) const
```

Get the state of a joystick button.

**Parameters:**
- *JoyId*: Identifier of the joystick to check (0 or 1)
- *Button*: Button to check

**Returns:**
True if button is down, false if button is up

Definition at line 65 of file `Input.cpp`.

```cpp
bool sf::Input::IsKeyDown(Key::Code KeyCode) const
```

Get the state of a key.

**Parameters:**
- *KeyCode*: Key to check

**Returns:**
True if key is down, false if key is up

Definition at line 47 of file `Input.cpp`. 
bool sf::Input::IsMouseButtonButtonDown ( Mouse::Button Button ) const

Get the state of a mouse button.

**Parameters:**

*Button*: Button to check

**Returns:**

True if button is down, false if button is up

Definition at line 56 of file Input.cpp.

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

- Input.hpp
- Input.cpp
sf::IPAddress
sf::IPAddress Class Reference

**IPAddress** provides easy manipulation of IP v4 addresses. [More...](#)

```cpp
#include <IPAddress.hpp>
```

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPAddress ()</td>
<td>Default constructor -- constructs an invalid address.</td>
</tr>
<tr>
<td>IPAddress (const std::string &amp;Address)</td>
<td>Construct the address from a string.</td>
</tr>
<tr>
<td>IPAddress (const char *Address)</td>
<td>Construct the address from a C-style string ; Needed for implicit conversions from literal strings to IPAddress to work.</td>
</tr>
<tr>
<td>IPAddress (Uint8 Byte0, Uint8 Byte1, Uint8 Byte2, Uint8 Byte3)</td>
<td>Construct the address from 4 bytes.</td>
</tr>
<tr>
<td>IPAddress (Uint32 Address)</td>
<td>Construct the address from a 32-bits integer.</td>
</tr>
<tr>
<td>bool IsValid () const</td>
<td>Tell if the address is a valid one.</td>
</tr>
<tr>
<td>std::string ToString () const</td>
<td>Get a string representation of the address.</td>
</tr>
<tr>
<td>Uint32 ToInteger () const</td>
<td>Get an integer representation of the address.</td>
</tr>
<tr>
<td>bool operator==(const IPAddress &amp;Other) const</td>
<td>Comparison operator ==.</td>
</tr>
<tr>
<td>bool operator!=(const IPAddress &amp;Other) const</td>
<td>Comparison operator !=.</td>
</tr>
<tr>
<td>bool operator&lt;(const IPAddress &amp;Other) const</td>
<td>Comparison operator &lt;.</td>
</tr>
<tr>
<td>bool operator&gt;(const IPAddress &amp;Other) const</td>
<td>Comparison operator &gt;.</td>
</tr>
<tr>
<td>bool operator&lt;=(const IPAddress &amp;Other) const</td>
<td>Comparison operator &lt;=.</td>
</tr>
<tr>
<td>bool operator&gt;=(const IPAddress &amp;Other) const</td>
<td>Comparison operator &gt;=.</td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>static IPAddress</th>
<th>GetLocalAddress ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get the computer's local IP address (from the LAN point of view).</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static IPAddress</th>
<th>GetPublicAddress (float Timeout=0.f)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get the computer's public IP address (from the web point of view).</td>
<td></td>
</tr>
</tbody>
</table>
### Static Public Attributes

<table>
<thead>
<tr>
<th>static const IPAddress</th>
<th>LocalHost</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>Local host address (to connect to the same computer).</em></td>
</tr>
</tbody>
</table>
Detailed Description

**IPAddress** provides easy manipulation of IP v4 addresses.

Definition at line 42 of file IPAddress.hpp.
Constructor & Destructor Documentation

<table>
<thead>
<tr>
<th>sf::IPAddress::IPAddress ( )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default constructor -- constructs an invalid address.</td>
</tr>
<tr>
<td>Default constructor.</td>
</tr>
<tr>
<td>Definition at line 45 of file IPAddress.cpp.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sf::IPAddress::IPAddress ( const std::string &amp; Address )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Construct the address from a string.</td>
</tr>
<tr>
<td><strong>Parameters:</strong></td>
</tr>
<tr>
<td>Address : IP address (&quot;xxx.xxx.xxx.xxx&quot;) or network name</td>
</tr>
<tr>
<td>Definition at line 55 of file IPAddress.cpp.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sf::IPAddress::IPAddress ( const char * Address )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Construct the address from a C-style string ; Needed for implicit conversions from literal strings to IPAddress to work.</td>
</tr>
<tr>
<td>Construct the address from a C-style string ; Needed for implicit conversions from literal strings to IPAddress to work.</td>
</tr>
<tr>
<td><strong>Parameters:</strong></td>
</tr>
<tr>
<td>Address : IP address (&quot;xxx.xxx.xxx.xxx&quot;) or network name</td>
</tr>
<tr>
<td>Definition at line 82 of file IPAddress.cpp.</td>
</tr>
</tbody>
</table>
### sf::IPAddress::IPAddress (Uint8 Byte0, Uint8 Byte1, Uint8 Byte2, Uint8 Byte3)

Construct the address from 4 bytes.

**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

Definition at line 108 of file IPAddress.cpp.

### sf::IPAddress::IPAddress (Uint32 Address)

Construct the address from a 32-bits integer.

**Parameters:**
- `Address` : 4 bytes of the address packed into a 32-bits integer

Definition at line 117 of file IPAddress.cpp.
### IPAddress sf::IPAddress::GetLocalAddress ( ) [static]

Get the computer's local IP address (from the LAN point of view).

**Returns:**
- Local IP address

Definition at line 156 of file IPAddress.cpp.

### IPAddress sf::IPAddress::GetPublicAddress ( float Timeout = 0.f ) [static]

Get the computer's public IP address (from the web point of view).

The only way to get a public address is to ask it to a distant website; as a consequence, this function may be very slow -- use it as few as possible!

**Parameters:**
- **Timeout**: Maximum time to wait, in seconds (0 by default: no timeout)

**Returns:**
- Public IP address

Definition at line 204 of file IPAddress.cpp.

### bool sf::IPAddress::IsValid ( ) const

Tell if the address is a valid one.
**Returns:**
True if address has a valid syntax

Definition at line 126 of file IPAddress.cpp.

```cpp
bool sf::IPAddress::operator!= ( const IPAddress & Other ) const
```

Comparison operator !=.

**Parameters:**
Other : Address to compare

**Returns:**
True if *this != Other

Definition at line 235 of file IPAddress.cpp.

```cpp
bool sf::IPAddress::operator< ( const IPAddress & Other ) const
```

Comparison operator <.

**Parameters:**
Other : Address to compare

**Returns:**
True if *this < Other

Definition at line 244 of file IPAddress.cpp.

```cpp
bool sf::IPAddress::operator<= ( const IPAddress & Other ) const
```

Comparison operator <=.
**Parameters:**

*Other* : Address to compare

**Returns:**

True if *this <= Other

Definition at line 262 of file IPAddress.cpp.

```cpp
bool sf::IPAddress::operator==( const IPAddress & Other ) const

Comparison operator ==.

**Parameters:**

*Other* : Address to compare

**Returns:**

True if *this == Other

Definition at line 226 of file IPAddress.cpp.

```cpp
bool sf::IPAddress::operator>( const IPAddress & Other ) const

Comparison operator >.

**Parameters:**

*Other* : Address to compare

**Returns:**

True if *this > Other

Definition at line 253 of file IPAddress.cpp.
### bool sf::IPAddress::operator>= ( const IPAddress & Other ) const

Comparison operator >=.

**Parameters:**

*Other* : Address to compare

**Returns:**

True if *this >= Other

Definition at line 271 of file IPAddress.cpp.

### Uint32 sf::IPAddress::ToInteger ( ) const

Get an integer representation of the address.

**Returns:**

32-bits integer containing the 4 bytes of the address, in system endianness

Definition at line 147 of file IPAddress.cpp.

### std::string sf::IPAddress::ToString ( ) const

Get a string representation of the address.

**Returns:**

*String* representation of the IP address ("xxx.xxx.xxx.xxx")

Definition at line 135 of file IPAddress.cpp.
Member Data Documentation

<table>
<thead>
<tr>
<th>const IPAddress sf::IPAddress::LocalHost [static]</th>
</tr>
</thead>
</table>

Local host address (to connect to the same computer).

Static member data.

Definition at line 196 of file IPAddress.hpp.

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

- IPAddress.hpp
- IPAddress.cpp

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
<table>
<thead>
<tr>
<th>Main Page</th>
<th>Namespaces</th>
<th>Classes</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class List</td>
<td>Class Index</td>
<td>Class Hierarchy</td>
<td>Class Members</td>
</tr>
</tbody>
</table>

sf::Listener
sf::Listener Class Reference

**Listener** is a global interface for defining the audio listener properties; the audio listener is the point in the scene from where all the sounds are heard. [More...](#)

#include `<Listener.hpp>`

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

<table>
<thead>
<tr>
<th>Function Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static void</td>
<td>SetGlobalVolume (float Volume)</td>
<td>Change the global volume of all the sounds.</td>
</tr>
<tr>
<td>static float</td>
<td>GetGlobalVolume ()</td>
<td>Get the current value of the global volume of all the sounds.</td>
</tr>
<tr>
<td>static void</td>
<td>SetPosition (float X, float Y, float Z)</td>
<td>Change the position of the listener (take 3 values).</td>
</tr>
<tr>
<td>static void</td>
<td>SetPosition (const Vector3f &amp;Position)</td>
<td>Change the position of the listener (take a 3D vector).</td>
</tr>
<tr>
<td>static Vector3f</td>
<td>GetPosition ()</td>
<td>Get the current position of the listener.</td>
</tr>
<tr>
<td>static void</td>
<td>SetTarget (float X, float Y, float Z)</td>
<td>Change the orientation of the listener (the point he must look at) (take 3 values).</td>
</tr>
<tr>
<td>static void</td>
<td>SetTarget (const Vector3f &amp;Target)</td>
<td>Change the orientation of the listener (the point he must look at) (take a 3D vector).</td>
</tr>
<tr>
<td>static Vector3f</td>
<td>GetTarget ()</td>
<td>Get the current orientation of the listener (the point he’s looking at).</td>
</tr>
</tbody>
</table>
**Detailed Description**

**Listener** is a global interface for defining the audio listener properties; the audio listener is the point in the scene from where all the sounds are heard.

Definition at line 42 of file `Listener.hpp`. 
### Member Function Documentation

**float sf::Listener::GetGlobalVolume( ) [static]**

Get the current value of the global volume of all the sounds.

**Returns:**
Current global volume, in the range [0, 100]

Definition at line 46 of file `Listener.cpp`.

**Vector3f sf::Listener::GetPosition( ) [static]**

Get the current position of the listener.

**Returns:**
Position of the listener in the world

Definition at line 76 of file `Listener.cpp`.

**Vector3f sf::Listener::GetTarget( ) [static]**

Get the current orientation of the listener (the point he's looking at).

Get the current orientation of the listener (the point he's looking at).

**Returns:**
Position of the point the listener is looking at

Definition at line 110 of file `Listener.cpp`. 
### void sf::Listener::SetGlobalVolume (float Volume) [static]

Change the global volume of all the sounds.

The default volume is 100

**Parameters:**

- **Volume**: New global volume, in the range [0, 100]

Definition at line 37 of file **Listener.cpp**.

### void sf::Listener::SetPosition (const Vector3f &Position) [static]

Change the position of the listener (take a 3D vector).

The default position is (0, 0, 0)

**Parameters:**

- **Position**: Position of the listener in the world

Definition at line 67 of file **Listener.cpp**.

### void sf::Listener::SetPosition (float X, float Y, float Z) [static]

Change the position of the listener (take 3 values).

The default position is (0, 0, 0)

**Parameters:**
$X,Y,Z$ : Position of the listener in the world

Definition at line 58 of file **Listener.cpp**.

```cpp
void sf::Listener::SetTarget(const Vector3f & Target) [static]
```

Change the orientation of the listener (the point he must look at) (take a 3D vector).

Change the orientation of the listener (the point he must look at) (take a 3D vector).

The default target is (0, 0, -1)

**Parameters:**

- **Target** : Position of the point the listener must look at

Definition at line 100 of file **Listener.cpp**.

```cpp
void sf::Listener::SetTarget(float X,
                              float Y,
                              float Z
                           ) [static]
```

Change the orientation of the listener (the point he must look at) (take 3 values).

Change the orientation of the listener (the point he must look at) (take 3 values).

The default target is (0, 0, -1)

**Parameters:**
$X,Y,Z$ : Position of the point the listener must look at

Definition at line 89 of file Listener.cpp.

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

- Listener.hpp
- Listener.cpp

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
sf::Lock
sf::Lock Class Reference

**Lock** is an exception-safe automatic wrapper for locking and unlocking mutexes. [More...](#)

```cpp
#include <Lock.hpp>
```

Inheritance diagram for sf::Lock:

![Inheritance Diagram](#)

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Lock (Mutex &amp;Mutex)</strong></td>
<td>Construct the lock with a target mutex (lock it).</td>
</tr>
<tr>
<td><strong>~Lock ()</strong></td>
<td>Destructor (unlocks the mutex).</td>
</tr>
</tbody>
</table>
**Detailed Description**

**Lock** is an exception-safe automatic wrapper for locking and unlocking mutexes.

Definition at line 42 of file **Lock.hpp**.
**Constructor & Destructor Documentation**

sf::Lock::Lock ( Mutex & Mutex )

Construct the lock with a target mutex (lock it).

**Parameters:**

- **Mutex** : Mutex to lock

Definition at line 37 of file Lock.cpp.

sf::Lock::~Lock ( )

Destructor (unlocks the mutex).

Definition at line 47 of file Lock.cpp.

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

- Lock.hpp
- Lock.cpp

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
sf::Matrix3
sf::Matrix3 Class Reference

Utility class to manipulate 3x3 matrices representing 2D transformations. [More...](#)

```cpp
#include <Matrix3.hpp>
```

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matrix3 ()</td>
<td>Default constructor (builds an identity matrix).</td>
</tr>
<tr>
<td>Matrix3 (float a00, float a01, float a02, float a10, float a11, float a12, float a20, float a21, float a22)</td>
<td>Construct a matrix from its 9 elements.</td>
</tr>
<tr>
<td>void SetFromTransformations (const Vector2f &amp;Center, const Vector2f &amp;Translation, float Rotation, const Vector2f &amp;Scale)</td>
<td>Build a matrix from a set of transformations.</td>
</tr>
<tr>
<td>Vector2f Transform (const Vector2f &amp;Point) const</td>
<td>Transform a point by the matrix.</td>
</tr>
<tr>
<td>Matrix3 GetInverse () const</td>
<td>Return the inverse of the matrix.</td>
</tr>
<tr>
<td>const float * Get4x4Elements () const</td>
<td>Return the elements of the matrix as a 4x4, in an array of 16 floats.</td>
</tr>
<tr>
<td>float operator() (unsigned int Row, unsigned int Col) const</td>
<td>Operator () overloads to access the matrix elements.</td>
</tr>
<tr>
<td>float &amp; operator() (unsigned int Row, unsigned int Col)</td>
<td></td>
</tr>
<tr>
<td>Matrix3 operator* (const Matrix3 &amp;Mat) const</td>
<td>Operator * overload to multiply two matrices.</td>
</tr>
<tr>
<td>Matrix3 &amp; operator*= (const Matrix3 &amp;Mat)</td>
<td>Operator *= overload to multiply-assign two matrices.</td>
</tr>
</tbody>
</table>
Static Public Attributes

```
static const Matrix3 Identity
Identity matrix.
```
Detailed Description

Utility class to manipulate 3x3 matrices representing 2D transformations.

Definition at line 42 of file Matrix3.hpp.
## Constructor & Destructor Documentation

<table>
<thead>
<tr>
<th>sf::Matrix3::Matrix3 ( )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default constructor (builds an identity matrix).</td>
</tr>
</tbody>
</table>

| sf::Matrix3::Matrix3 ( float a00,  
| float a01,  
| float a02,  
| float a10,  
| float a11,  
| float a12,  
| float a20,  
| float a21,  
<table>
<thead>
<tr>
<th>float a22 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Construct a matrix from its 9 elements.</td>
</tr>
</tbody>
</table>
### Member Function Documentation

**const float* sf::Matrix3::Get4x4Elements() const**

Return the elements of the matrix as a 4x4, in an array of 16 floats.

**Returns:**
Pointer to the 4x4 matrix elements

**Matrix3 sf::Matrix3::GetInverse() const**

Return the inverse of the matrix.

**Returns:**
A new matrix which is the inverse of this

**float sf::Matrix3::operator()(unsigned int Row, unsigned int Col) const**

Operator () overloads to access the matrix elements.

**Parameters:**
- **Row**: Element row (0 based)
- **Col**: Element column (0 based)

**Returns:**
Matrix element (Row, Col)

**Matrix3 sf::Matrix3::operator*(const Matrix3 & Mat) const**
Operator * overload to multiply two matrices.

**Parameters:**

\[ \text{Mat} \] : Matrix to multiply

**Returns:**

this * Mat

---

Matrix3& sf::Matrix3::operator*= ( const Matrix3 & \text{Mat} )

Operator *\= overload to multiply-assign two matrices.

**Parameters:**

\[ \text{Mat} \] : Matrix to multiply

**Returns:**

this * Mat

---

void sf::Matrix3::SetFromTransformations ( const Vector2f & \text{Center}, const Vector2f & \text{Translation}, float \text{Rotation}, const Vector2f & \text{Scale} )

Build a matrix from a set of transformations.

**Parameters:**

\text{Center} : Origin for the transformations
\text{Translation} : Translation offset
\text{Rotation} : Rotation angle in degrees
\text{Scale} : Scaling factors
Transform a point by the matrix.

**Parameters:**

*Point*: Point to transform

**Returns:**

Transformed point
Member Data Documentation

const Matrix3 sf::Matrix3::Identity [static]

Identity matrix.

Definition at line 133 of file Matrix3.hpp.

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

- Matrix3.hpp
- Matrix3.cpp
sf::Music
sf::Music Class Reference

**Music** defines a big sound played using streaming, so usually what we call a music :) [More...](#)

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

Inheritance diagram for sf::Music:

```
sf::Thread --> sf::Sound

sf::SoundStream --> sf::Music
```

[List of all members.](#)
Public Types

enum Status

Enumeration of the sound states. More...
Public Member Functions

<table>
<thead>
<tr>
<th>Music (std::size_t BufferSize=44100)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Construct the music with a buffer size.</td>
</tr>
<tr>
<td>~Music ()</td>
</tr>
<tr>
<td>Destructor.</td>
</tr>
<tr>
<td>bool OpenFromFile (const std::string &amp;Filename)</td>
</tr>
<tr>
<td>Open a music file (doesn't play it -- call Play() for that).</td>
</tr>
<tr>
<td>bool OpenFromMemory (const char *Data, std::size_t SizeInBytes)</td>
</tr>
<tr>
<td>Open a music file from memory (doesn't play it -- call Play() for that).</td>
</tr>
<tr>
<td>float GetDuration () const</td>
</tr>
<tr>
<td>Get the music duration.</td>
</tr>
<tr>
<td>void Play ()</td>
</tr>
<tr>
<td>Start playing the audio stream.</td>
</tr>
<tr>
<td>void Stop ()</td>
</tr>
<tr>
<td>Stop playing the audio stream.</td>
</tr>
<tr>
<td>unsigned int GetChannelsCount () const</td>
</tr>
<tr>
<td>Return the number of channels (1 = mono, 2 = stereo).</td>
</tr>
<tr>
<td>unsigned int GetSampleRate () const</td>
</tr>
<tr>
<td>Get the stream sample rate.</td>
</tr>
<tr>
<td>Status GetStatus () const</td>
</tr>
<tr>
<td>Get the status of the stream (stopped, paused, playing).</td>
</tr>
<tr>
<td>float GetPlayingOffset () const</td>
</tr>
<tr>
<td>Get the current playing position of the stream.</td>
</tr>
<tr>
<td>void SetLoop (bool Loop)</td>
</tr>
<tr>
<td>Set the stream loop state.</td>
</tr>
<tr>
<td>bool GetLoop () const</td>
</tr>
<tr>
<td>Tell whether or not the stream is looping.</td>
</tr>
</tbody>
</table>
Protected Member Functions

```c
void Initialize (unsigned int ChannelsCount, unsigned int SampleRate)
    Set the audio stream parameters, you must call it before Play().
```
Detailed Description

Music defines a big sound played using streaming, so usually what we call a music :).

Definition at line 47 of file Music.hpp.
**Member Enumeration Documentation**

<table>
<thead>
<tr>
<th><strong>enum sf::Sound::Status</strong> [inherited]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enumeration of the sound states.</td>
</tr>
<tr>
<td>Definition at line 52 of file Sound.hpp.</td>
</tr>
</tbody>
</table>
**Constructor & Destructor Documentation**

```cpp
sf::Music::Music ( std::size_t BufferSize = 44100 ) [explicit]
```

Construct the music with a buffer size.

**Parameters:**

- `BufferSize`: Size of the internal buffer, expressed in number of samples (ie. size taken by the music in memory) (44100 by default)

Definition at line 40 of file `Music.cpp`.

```cpp
sf::Music::~Music ( )
```

Destructor.

Definition at line 52 of file `Music.cpp`. 
# Member Function Documentation

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Return Type</th>
<th>Description</th>
<th>Returns:</th>
<th>Definition at line</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>unsigned int sf::SoundStream::GetChannelsCount( ) const [inherited]</code></td>
<td>unsigned int</td>
<td>Return the number of channels (1 = mono, 2 = stereo).</td>
<td>Number of channels</td>
<td>126 of file SoundStream.cpp.</td>
</tr>
<tr>
<td><code>float sf::Music::GetDuration( ) const</code></td>
<td>float</td>
<td>Get the music duration.</td>
<td>Music duration, in seconds</td>
<td>148 of file Music.cpp.</td>
</tr>
<tr>
<td><code>bool sf::SoundStream::GetLoop( ) const [inherited]</code></td>
<td>bool</td>
<td>Tell whether or not the stream is looping.</td>
<td>Tell whether or not the music is looping.</td>
<td></td>
</tr>
</tbody>
</table>
**Returns:**
True if the music is looping, false otherwise

Reimplemented from **sf::Sound**.

Definition at line 180 of file **SoundStream.cpp**.

---

**float sf::SoundStream::GetPlayingOffset() const [inherited]**

Get the current playing position of the stream.

**Returns:**
Current playing position, expressed in seconds

Reimplemented from **sf::Sound**.

Definition at line 162 of file **SoundStream.cpp**.

---

**unsigned int sf::SoundStream::GetSampleRate() const [inherited]**

Get the stream sample rate.

Get the sound frequency (sample rate).

**Returns:**
Stream frequency (number of samples per second)

Definition at line 135 of file **SoundStream.cpp**.

---

**Sound::Status sf::SoundStream::GetStatus() const [inherited]**
Get the status of the stream (stopped, paused, playing).

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

**Returns:**
Current status of the sound

Reimplemented from `sf::Sound`.

Definition at line 144 of file `SoundStream.cpp`.

```cpp
void sf::SoundStream::Initialize(unsigned int ChannelsCount, unsigned int SampleRate)
```

Set the audio stream parameters, you must call it before `Play()`.

**Parameters:**
- `ChannelsCount` : Number of channels
- `SampleRate` : Sample rate

Definition at line 64 of file `SoundStream.cpp`.

```cpp
bool sf::Music::OpenFromFile(const std::string &Filename)
```

Open a music file (doesn't play it -- call `Play()` for that).

**Parameters:**
- `Filename` : Path of the music file to open

**Returns:**
True if loading has been successful
bool sf::Music::OpenFromMemory ( const char * Data,
    std::size_t SizeInBytes
  )

Open a music file from memory (doesn't play it -- call Play() for that).

Parameters:
  Data : Pointer to the file data in memory
  SizeInBytes : Size of the data to load, in bytes

Returns:
  True if loading has been successful

Definition at line 91 of file Music.cpp.

void sf::SoundStream::Play ( ) [inherited]

Start playing the audio stream.

Reimplemented from sf::Sound.

Definition at line 85 of file SoundStream.cpp.

void sf::SoundStream::SetLoop ( bool Loop ) [inherited]

Set the stream loop state.

Set the music loop state.
This parameter is disabled by default

**Parameters:**

*Loop*: True to play in loop, false to play once

Reimplemented from *sf::Sound*.

Definition at line 171 of file *SoundStream.cpp*.

```cpp
void sf::SoundStream::Stop() [inherited]
```

Stop playing the audio stream.

Reimplemented from *sf::Sound*.

Definition at line 115 of file *SoundStream.cpp*.

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

- *Music.hpp*
- *Music.cpp*

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::

Documentation generated by doxygen 1.5.2 ::
sf::Mutex
sf::Mutex Class Reference

**Mutex** defines a mutex (MUTual EXclusion) object, that allows a thread to lock critical instructions to avoid simultaneous access with other threads. [More...]

```
#include <Mutex.hpp>
```

Inheritance diagram for sf::Mutex:

![Inheritance Diagram](image)

[List of all members.]
Public Member Functions

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mutex ()</strong></td>
<td>Default constructor.</td>
</tr>
<tr>
<td><strong>~Mutex ()</strong></td>
<td>Destructor.</td>
</tr>
<tr>
<td><strong>void Lock ()</strong></td>
<td>Lock the mutex.</td>
</tr>
<tr>
<td><strong>void Unlock ()</strong></td>
<td>Unlock the mutex.</td>
</tr>
</tbody>
</table>
Detailed Description

**Mutex** defines a mutex (MUTual EXclusion) object, that allows a thread to lock critical instructions to avoid simultaneous access with other threads.

The Win32 version uses critical sections, as it is faster than mutexes. See **Lock** for an efficient way of using it.

Definition at line 45 of file *Win32/Mutex.hpp*. 
## Constructor & Destructor Documentation

<table>
<thead>
<tr>
<th>sf::Mutex::Mutex ( )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default constructor.</td>
</tr>
<tr>
<td>Definition at line 36 of file Mutex.cpp.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sf::Mutex::~Mutex ( )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Destructor.</td>
</tr>
<tr>
<td>Definition at line 45 of file Mutex.cpp.</td>
</tr>
</tbody>
</table>
## Member Function Documentation

<table>
<thead>
<tr>
<th>void sf::Mutex::Lock ( )</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Lock</strong> the mutex.</td>
</tr>
<tr>
<td>Definition at line 54 of file Mutex.cpp.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void sf::Mutex::Unlock ( )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unlock the mutex.</td>
</tr>
<tr>
<td>Definition at line 63 of file Mutex.cpp.</td>
</tr>
</tbody>
</table>

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

- **Win32/Mutex.hpp**
- **Mutex.cpp**
sf::NonCopyable
Utility base class to easily declare non-copyable classes.

#include <NonCopyable.hpp>

Inheritance diagram for sf::NonCopyable:

List of all members.
### Protected Member Functions

<table>
<thead>
<tr>
<th>NonCopyable ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>The default constructor won't be generated, so provide it.</em></td>
</tr>
</tbody>
</table>
Detailed Description

Utility base class to easily declare non-copyable classes.

Just inherit from `NonCopyable` to get a non-copyable class.

Definition at line 40 of file `NonCopyable.hpp`.
## Constructor & Destructor Documentation

<table>
<thead>
<tr>
<th>sf::NonCopyable::NonCopyable ( ) [inline, protected]</th>
</tr>
</thead>
<tbody>
<tr>
<td>The default constructor won't be generated, so provide it.</td>
</tr>
<tr>
<td>Definition at line 48 of file NonCopyable.hpp.</td>
</tr>
</tbody>
</table>

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

- **NonCopyable.hpp**

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
sf::Packet
Packet wraps data to send / to receive through the network. More...

#include <Packet.hpp>

List of all members.
<table>
<thead>
<tr>
<th>Public Member Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Packet()</code></td>
</tr>
<tr>
<td>Default constructor.</td>
</tr>
<tr>
<td>virtual <code>~Packet()</code></td>
</tr>
<tr>
<td>Virtual destructor.</td>
</tr>
<tr>
<td>void <code>Append</code> (const void *Data, std::size_t SizeInBytes)</td>
</tr>
<tr>
<td>Append data to the end of the packet.</td>
</tr>
<tr>
<td>void <code>Clear()</code></td>
</tr>
<tr>
<td>Clear the packet data.</td>
</tr>
<tr>
<td>const char * <code>GetData</code> () const</td>
</tr>
<tr>
<td>Get a pointer to the data contained in the packet</td>
</tr>
<tr>
<td>Warning: the returned pointer may be invalid after you append data to the packet.</td>
</tr>
<tr>
<td>std::size_t <code>GetDataSize</code> () const</td>
</tr>
<tr>
<td>Get the size of the data contained in the packet.</td>
</tr>
<tr>
<td>bool <code>EndOfPacket</code> () const</td>
</tr>
<tr>
<td>Tell if the reading position has reached the end of the packet.</td>
</tr>
<tr>
<td>operator bool () const</td>
</tr>
<tr>
<td>Return the validity of packet.</td>
</tr>
<tr>
<td><code>Packet &amp; operator&gt;&gt; (bool &amp;Data)</code></td>
</tr>
<tr>
<td>Operator &gt;&gt; overloads to extract data from the packet.</td>
</tr>
<tr>
<td><code>Packet &amp; operator&gt;&gt; (Int8 &amp;Data)</code></td>
</tr>
<tr>
<td><code>Packet &amp; operator&gt;&gt; (Uint8 &amp;Data)</code></td>
</tr>
<tr>
<td><code>Packet &amp; operator&gt;&gt; (Int16 &amp;Data)</code></td>
</tr>
<tr>
<td><code>Packet &amp; operator&gt;&gt; (Uint16 &amp;Data)</code></td>
</tr>
<tr>
<td><code>Packet &amp; operator&gt;&gt; (Int32 &amp;Data)</code></td>
</tr>
<tr>
<td><code>Packet &amp; operator&gt;&gt; (Uint32 &amp;Data)</code></td>
</tr>
<tr>
<td><code>Packet &amp; operator&gt;&gt; (float &amp;Data)</code></td>
</tr>
<tr>
<td><code>Packet &amp; operator&gt;&gt; (double &amp;Data)</code></td>
</tr>
<tr>
<td><code>Packet &amp; operator&gt;&gt; (char *Data)</code></td>
</tr>
<tr>
<td><code>Packet &amp; operator&gt;&gt; (std::string &amp;Data)</code></td>
</tr>
<tr>
<td><code>Packet &amp; operator&gt;&gt; (wchar_t *Data)</code></td>
</tr>
<tr>
<td><code>Packet &amp; operator&gt;&gt; (std::wstring &amp;Data)</code></td>
</tr>
<tr>
<td><code>Packet &amp; operator&lt;&lt;(bool Data)</code></td>
</tr>
<tr>
<td>Operator &lt;&lt; overloads to put data into the packet.</td>
</tr>
<tr>
<td><code>Packet &amp; operator&lt;&lt;(Int8 Data)</code></td>
</tr>
<tr>
<td><code>Packet &amp; operator&lt;&lt;(Uint8 Data)</code></td>
</tr>
<tr>
<td><code>Packet &amp; operator&lt;&lt;(Int16 Data)</code></td>
</tr>
<tr>
<td><code>Packet &amp; operator&lt;&lt;(Uint16 Data)</code></td>
</tr>
<tr>
<td><code>Packet &amp; operator&lt;&lt;(Int32 Data)</code></td>
</tr>
<tr>
<td><code>Packet &amp; operator&lt;&lt;(Uint32 Data)</code></td>
</tr>
<tr>
<td><code>Packet &amp; operator&lt;&lt;(float Data)</code></td>
</tr>
<tr>
<td><code>Packet &amp; operator&lt;&lt;(double Data)</code></td>
</tr>
<tr>
<td><code>Packet &amp; operator&lt;&lt;(const char *Data)</code></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td><strong>Packet</strong> &amp;</td>
</tr>
<tr>
<td><strong>Packet</strong> &amp;</td>
</tr>
<tr>
<td><strong>Packet</strong> &amp;</td>
</tr>
</tbody>
</table>
Friends

<table>
<thead>
<tr>
<th>class</th>
<th>SocketTCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>SocketUDP</td>
</tr>
</tbody>
</table>
Detailed Description

Packet wraps data to send / to receive through the network.

Definition at line 41 of file Packet.hpp.
# Constructor & Destructor Documentation

<table>
<thead>
<tr>
<th>sf::Packet::Packet ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default constructor.</td>
</tr>
<tr>
<td>Definition at line 38 of file <em>Packet.cpp</em>.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sf::Packet::~Packet () [virtual]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtual destructor.</td>
</tr>
<tr>
<td>Definition at line 49 of file <em>Packet.cpp</em>.</td>
</tr>
</tbody>
</table>
## 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 bytes to append
- `SizeInBytes` : Number of bytes to append

Definition at line 58 of file `Packet.cpp`.

### void sf::Packet::Clear ( )

Clear the packet data.

Definition at line 72 of file `Packet.cpp`.

### bool sf::Packet::EndOfPacket ( ) const

Tell if the reading position has reached the end of the packet.

**Returns:**
True if all data have been read into the packet

Definition at line 103 of file `Packet.cpp`.

### const char * sf::Packet::GetData ( ) const
Get a pointer to the data contained in the packet. Warning: the returned pointer may be invalid after you append data to the packet.

Get a pointer to the data contained in the packet. Warning: the returned pointer may be invalid after you append data to the packet.

**Returns:**
- Pointer to the data

Definition at line 85 of file `Packet.cpp`.

<table>
<thead>
<tr>
<th>std::size_t sf::Packet::GetDataSize ( ) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get the size of the data contained in the packet.</td>
</tr>
</tbody>
</table>

**Returns:**
- Data size, in bytes

Definition at line 94 of file `Packet.cpp`.

<table>
<thead>
<tr>
<th>sf::Packet::operator bool ( ) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return the validity of packet.</td>
</tr>
<tr>
<td>Tell if the packet is valid for reading.</td>
</tr>
</tbody>
</table>

**Returns:**
- True if last data extraction from packet was successful

Definition at line 112 of file `Packet.cpp`. 
Packet & sf::Packet::operator<<( bool Data )

Operator << overloads to put data into the packet.
Definition at line 290 of file Packet.cpp.

Packet & sf::Packet::operator>>( bool & Data )

Operator >> overloads to extract data from the packet.
Definition at line 121 of file Packet.cpp.

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

- Packet.hpp
- Packet.cpp

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
sf::PostFX
sf::PostFX Class Reference

PostFX is used to apply a post effect to a window. More...

#include <PostFX.hpp>

Inheritance diagram for sf::PostFX:

List of all members.
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PostFX ()</strong></td>
<td>Default constructor.</td>
</tr>
<tr>
<td><strong>PostFX (const PostFX &amp;Copy)</strong></td>
<td>Copy constructor.</td>
</tr>
<tr>
<td><strong>~PostFX ()</strong></td>
<td>Destructor.</td>
</tr>
<tr>
<td><strong>bool LoadFromFile (const std::string &amp;Filename)</strong></td>
<td>Load the effect from a file.</td>
</tr>
<tr>
<td><strong>bool LoadFromMemory (const std::string &amp;Effect)</strong></td>
<td>Load the effect from a text in memory.</td>
</tr>
<tr>
<td><strong>void SetParameter (const std::string &amp;Name, float X)</strong></td>
<td>Change a parameter of the effect (1 float).</td>
</tr>
<tr>
<td><strong>void SetParameter (const std::string &amp;Name, float X, float Y)</strong></td>
<td>Change a parameter of the effect (2 floats).</td>
</tr>
<tr>
<td><strong>void SetParameter (const std::string &amp;Name, float X, float Y, float Z)</strong></td>
<td>Change a parameter of the effect (3 floats).</td>
</tr>
<tr>
<td><strong>void SetParameter (const std::string &amp;Name, float X, float Y, float Z, float W)</strong></td>
<td>Change a parameter of the effect (4 floats).</td>
</tr>
<tr>
<td>*<em>void SetTexture (const std::string &amp;Name, Image <em>Texture)</em></em></td>
<td>Set a texture parameter.</td>
</tr>
<tr>
<td><strong>PostFX &amp; operator= (const PostFX &amp;Other)</strong></td>
<td>Assignment operator.</td>
</tr>
<tr>
<td><strong>void SetPosition (float X, float Y)</strong></td>
<td>Set the position of the object (take 2 values).</td>
</tr>
<tr>
<td><strong>void SetPosition (const Vector2f &amp;Position)</strong></td>
<td>Set the position of the object (take a 2D vector).</td>
</tr>
<tr>
<td><strong>void SetX (float X)</strong></td>
<td>Set the X position of the object.</td>
</tr>
<tr>
<td><strong>void SetY (float Y)</strong></td>
<td>Set the Y position of the object.</td>
</tr>
<tr>
<td><strong>void SetScale (float ScaleX, float ScaleY)</strong></td>
<td>Set the scale of the object (take 2 values).</td>
</tr>
<tr>
<td><strong>void SetScale (const Vector2f &amp;Scale)</strong></td>
<td>Set the scale of the object (take a 2D vector).</td>
</tr>
<tr>
<td><strong>void SetScaleX (float FactorX)</strong></td>
<td>Set the X scale factor of the object.</td>
</tr>
<tr>
<td><strong>void SetScaleY (float FactorY)</strong></td>
<td>Set the Y scale factor of the object.</td>
</tr>
<tr>
<td><strong>void SetCenter (float CenterX, float CenterY)</strong></td>
<td>Set the center of the object, in coordinates relative to the top-left of the object (take 2 values).</td>
</tr>
</tbody>
</table>
**void** **SetCenter** (const **Vector2f** &Center)

Set the center of the object, in coordinates relative to the top-left of the object (take a 2D vector).

**void** **SetRotation** (float Rotation)

Set the orientation of the object.

**void** **SetColor** (const **Color** &Col)

Set the color of the object.

**void** **SetBlendMode** (Blend::Mode Mode)

Set the blending mode for the object.

**const Vector2f** & **GetPosition** () const

Get the position of the object.

**const Vector2f** & **GetScale** () const

Get the current scale of the object.

**const Vector2f** & **GetCenter** () const

Get the center of the object.

**float** **GetRotation** () const

Get the orientation of the object.

**const Color** & **GetColor** () const

Get the color of the object.

**Blend::Mode** **GetBlendMode** () const

Get the current blending mode.

**void** **Move** (float OffsetX, float OffsetY)

Move the object of a given offset (take 2 values).

**void** **Move** (const **Vector2f** &Offset)

Move the object of a given offset (take a 2D vector).

**void** **Scale** (float FactorX, float FactorY)

Scale the object (take 2 values).

**void** **Scale** (const **Vector2f** &Factor)

Scale the object (take a 2D vector).

**void** **Rotate** (float Angle)

Rotate the object.

**sf::Vector2f** **TransformToLocal** (const **sf::Vector2f** &Point) const

Transform a point from global coordinates into local coordinates (ie it applies the inverse of object’s center, translation, rotation and scale to the point).

**sf::Vector2f** **TransformToGlobal** (const **sf::Vector2f** &Point) const

Transform a point from local coordinates into global coordinates (ie it applies the object's center, translation, rotation and scale to the point).
**Static Public Member Functions**

<table>
<thead>
<tr>
<th>static bool</th>
<th>CanUsePostFX ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Tell whether or not the system supports post-effects.</em></td>
<td></td>
</tr>
</tbody>
</table>
**Protected Member Functions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual void Render (RenderTarget &amp;Target) const</td>
<td>/see Drawable::Render</td>
</tr>
<tr>
<td>const Matrix3 &amp; GetMatrix () const</td>
<td>Get the transform matrix of the drawable.</td>
</tr>
<tr>
<td>const Matrix3 &amp; GetInverseMatrix () const</td>
<td>Get the inverse transform matrix of the drawable.</td>
</tr>
</tbody>
</table>
Detailed Description

PostFX is used to apply a post effect to a window.

Definition at line 43 of file PostFX.hpp.
# Constructor & Destructor Documentation

<table>
<thead>
<tr>
<th>sf::PostFX::PostFX()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default constructor.</td>
</tr>
<tr>
<td>Definition at line 43 of file PostFX.cpp.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sf::PostFX::PostFX(const PostFX &amp; Copy)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy constructor.</td>
</tr>
<tr>
<td><strong>Parameters:</strong></td>
</tr>
<tr>
<td>Copy : Instance to copy</td>
</tr>
<tr>
<td>Definition at line 54 of file PostFX.cpp.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sf::PostFX::~PostFX()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Destructor.</td>
</tr>
<tr>
<td>Definition at line 73 of file PostFX.cpp.</td>
</tr>
</tbody>
</table>
Member Function Documentation

### bool sf::PostFX::CanUsePostFX ( ) [static]

Tell whether or not the system supports post-effects.

**Returns:**
True if the system can use post-effects

Definition at line 265 of file PostFX.cpp.

### Blend::Mode sf::Drawable::GetBlendMode ( ) const [inherited]

Get the current blending mode.

**Returns:**
Current blending mode

Definition at line 258 of file Drawable.cpp.

### const Vector2f & sf::Drawable::GetCenter ( ) const [inherited]

Get the center of the object.

**Returns:**
Current position of the center

Definition at line 231 of file Drawable.cpp.

### const Color & sf::Drawable::GetColor ( ) const [inherited]
Get the color of the object.

**Returns:**
Current color

Definition at line 249 of file **Drawable.cpp**.

```cpp
const Matrix3 & sf::Drawable::GetInverseMatrix() const [protected, inherited]
```

Get the inverse transform matrix of the drawable.

**Returns:**
Inverse transform matrix

Definition at line 350 of file **Drawable.cpp**.

```cpp
const Matrix3 & sf::Drawable::GetMatrix() const [protected, inherited]
```

Get the transform matrix of the drawable.

**Returns:**
Transform matrix

Definition at line 334 of file **Drawable.cpp**.

```cpp
const Vector2f & sf::Drawable::GetPosition() const [inherited]
```

Get the position of the object.

**Returns:**
Current position
**float sf::Drawable::GetRotation ( ) const [inherited]**

Get the orientation of the object.

Rotation is always in the range \([0, 360]\)

**Returns:**
Current rotation, in degrees

Definition at line 240 of file Drawable.cpp.

**const Vector2f & sf::Drawable::GetScale ( ) const [inherited]**

Get the current scale of the object.

**Returns:**
Current scale factor (always positive)

Definition at line 222 of file Drawable.cpp.

**bool sf::PostFX::LoadFromFile ( const std::string & Filename )**

Load the effect from a file.

**Parameters:**

*Filename* : Path of the effect file to load

**Returns:**
True on success
bool sf::PostFX::LoadFromMemory ( const std::string & Effect )

Load the effect from a text in memory.

**Parameters:**

Effect : String containing the effect code

**Returns:**

True on success

Definition at line 112 of file PostFX.cpp.

void sf::Drawable::Move ( const Vector2f & Offset ) [inherited]

Move the object of a given offset (take a 2D vector).

**Parameters:**

Offset : Amount of units to move the object of

Definition at line 278 of file Drawable.cpp.

void sf::Drawable::Move ( float OffsetX,
                          float OffsetY
                      ) [inherited]

Move the object of a given offset (take 2 values).

**Parameters:**

OffsetX : X offset
OffsetY : Y offset
Definition at line 268 of file Drawable.cpp.

PostFX & sf::PostFX::operator= ( const PostFX & Other )

Assignment operator.

Parameters:
  Other : Instance to assign

Returns:
  Reference to the post-effect

Definition at line 249 of file PostFX.cpp.

void sf::PostFX::Render ( RenderTarget & Target ) const [protected, virtual]

/see Drawable::Render

Implements sf::Drawable.

Definition at line 280 of file PostFX.cpp.

void sf::Drawable::Rotate ( float Angle ) [inherited]

Rotate the object.

Parameters:
  Angle : Angle of rotation, in degrees

Definition at line 306 of file Drawable.cpp.

void sf::Drawable::Scale ( const Vector2f & Factor ) [inherited]
Scale the object (take a 2D vector).

**Parameters:**

*Factor*: Scaling factors (both values must be strictly positive)

Definition at line 297 of file `Drawable.cpp`.

```cpp
void sf::Drawable::Scale ( float FactorX,
                          float FactorY ) [inherited]
```

Scale the object (take 2 values).

**Parameters:**

*FactorX*: Scaling factor on X (must be strictly positive)

*FactorY*: Scaling factor on Y (must be strictly positive)

Definition at line 287 of file `Drawable.cpp`.

```cpp
void sf::Drawable::SetBlendMode ( Blend::Mode Mode ) [inherited]
```

Set the blending mode for the object.

The default blend mode is `Blend::Alpha`

**Parameters:**

*Mode*: New blending mode

The default blend mode is `Blend::Alpha`

Definition at line 204 of file `Drawable.cpp`. 
void sf::Drawable::SetCenter ( const Vector2f & Center ) [inherited]

Set the center of the object, in coordinates relative to the top-left of the object (take a 2D vector).

Set the center of the object, in coordinates relative to the top-left of the object (take a 2D vector).

The default center is (0, 0)

**Parameters:**

- `Center` : New center

Definition at line 171 of file Drawable.cpp.

void sf::Drawable::SetCenter ( float centerX, float centerY ) [inherited]

Set the center of the object, in coordinates relative to the top-left of the object (take 2 values).

Set the center of the object, in coordinates relative to the top-left of the object (take 2 values).

The default center is (0, 0)

**Parameters:**

- `centerX` : X coordinate of the center
- `centerY` : Y coordinate of the center

The default center is (0, 0)
Definition at line 157 of file Drawable.cpp.

```cpp
void sf::Drawable::SetColor (const Color & Col) [inherited]
```

Set the color of the object.

The default color is white

**Parameters:**
- `Col`: New color

The default color is white

Definition at line 194 of file Drawable.cpp.

```cpp
void sf::PostFX::SetParameter (const std::string & Name,
                                float          X,
                                float          Y,
                                float          Z,
                                float          W)
```

Change a parameter of the effect (4 floats).

**Parameters:**
- `Name`: Parameter name in the effect
- `X, Y, Z, W`: Values to assign

Definition at line 199 of file PostFX.cpp.

```cpp
void sf::PostFX::SetParameter (const std::string & Name,
                                float          X,
                                float          Y,
```

```cpp
```
Change a parameter of the effect (3 floats).

**Parameters:**
- **Name**: Parameter name in the effect
- **X,Y,Z**: Values to assign

Definition at line **176** of file **PostFX.cpp**.

```cpp
void sf::PostFX::SetParameter ( const std::string & Name,
    float X,
    float Y )
```

Change a parameter of the effect (2 floats).

**Parameters:**
- **Name**: Parameter name in the effect
- **X,Y**: Values to assign

Definition at line **153** of file **PostFX.cpp**.

```cpp
void sf::PostFX::SetParameter ( const std::string & Name,
    float X )
```

Change a parameter of the effect (1 float).

**Parameters:**
- **Name**: Parameter name in the effect
- **X**: Value to assign
void sf::Drawable::setPosition ( const Vector2f & Position ) [inherited]

Set the position of the object (take a 2D vector).

**Parameters:**

Position : New position

Definition at line 75 of file Drawable.cpp.

void sf::Drawable::setPosition ( float X, float Y ) [inherited]

Set the position of the object (take 2 values).

**Parameters:**

X : New X coordinate
Y : New Y coordinate

Definition at line 65 of file Drawable.cpp.

void sf::Drawable::setRotation ( float Rotation ) [inherited]

Set the orientation of the object.

**Parameters:**

Rotation : Angle of rotation, in degrees

Definition at line 180 of file Drawable.cpp.
### void sf::Drawable::SetScale (const Vector2f & Scale) [inherited]

Set the scale of the object (take a 2D vector).

**Parameters:**
- *Scale*: New scale (both values must be strictly positive)

Definition at line 117 of file Drawable.cpp.

### void sf::Drawable::SetScale (float ScaleX, float ScaleY) [inherited]

Set the scale of the object (take 2 values).

**Parameters:**
- *ScaleX*: New horizontal scale (must be strictly positive)
- *ScaleY*: New vertical scale (must be strictly positive)

Definition at line 107 of file Drawable.cpp.

### void sf::Drawable::SetScaleX (float FactorX) [inherited]

Set the X scale factor of the object.

**Parameters:**
- *X*: New X scale factor

Definition at line 127 of file Drawable.cpp.

### void sf::Drawable::SetScaleY (float FactorY) [inherited]
Set the Y scale factor of the object.

**Parameters:**

- **Y**: New Y scale factor

Definition at line 141 of file `Drawable.cpp`.

```cpp
void sf::PostFX::SetTexture ( const std::string & Name, Image * Texture )
```

Set a texture parameter.

**Parameters:**

- **Name**: Texture name in the effect
- **Texture**: Image to set (pass NULL to use content of current framebuffer)

Definition at line 222 of file `PostFX.cpp`.

```cpp
void sf::Drawable::SetX ( float X ) [inherited]
```

Set the X position of the object.

**Parameters:**

- **X**: New X coordinate

Definition at line 85 of file `Drawable.cpp`.

```cpp
void sf::Drawable::SetY ( float Y ) [inherited]
```

Set the Y position of the object.
**Parameters:**

- \( Y \) : New \( Y \) coordinate

Definition at line 96 of file `Drawable.cpp`.

```cpp
sf::Vector2f sf::Drawable::TransformToGlobal ( const sf::Vector2f & Point )
```

Transform a point from local coordinates into global coordinates (ie it applies the object's center, translation, rotation and scale to the point).

**Parameters:**

- \( Point \) : Point to transform

**Returns:**

Transformed point

Definition at line 325 of file `Drawable.cpp`.

```cpp
sf::Vector2f sf::Drawable::TransformToLocal ( const sf::Vector2f & Point )
```

Transform a point from global coordinates into local coordinates (ie it applies the inverse of object's center, translation, rotation and scale to the point).

Transform a point from global coordinates into local coordinates (ie it applies the inverse of object's center, translation, rotation and scale to the point).
**Parameters:**

*Point* : Point to transform

**Returns:**

Transformed point

Definition at line 316 of file *Drawable.cpp*.

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

- *PostFX.hpp*
- *PostFX.cpp*

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::

Documentation generated by *doxygen 1.5.2* ::
sf::Randomizer
Randomizer is an utility class for generating pseudo-random numbers. [More...](#)

```
#include <Randomizer.hpp>
```

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

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static void</td>
<td><strong>SetSeed</strong> (unsigned int Seed)</td>
<td>Set the seed for the generator.</td>
</tr>
<tr>
<td>static unsigned int</td>
<td><strong>GetSeed</strong> ()</td>
<td>Get the seed used to generate random numbers the generator.</td>
</tr>
<tr>
<td>static float</td>
<td><strong>Random</strong> (float Begin, float End)</td>
<td>Get a random float number in a given range.</td>
</tr>
<tr>
<td>static int</td>
<td><strong>Random</strong> (int Begin, int End)</td>
<td>Get a random integer number in a given range.</td>
</tr>
</tbody>
</table>
Detailed Description

**Randomizer** is an utility class for generating pseudo-random numbers.

Definition at line 40 of file *Randomizer.hpp*. 
### Member Function Documentation

**unsigned int sf::Randomizer::GetSeed ( ) [static]**

Get the seed used to generate random numbers the generator.

**Returns:**
- Current seed

Definition at line 67 of file **Randomizer.cpp**.

---

**int sf::Randomizer::Random ( int Begin, int End ) [static]**

Get a random integer number in a given range.

**Returns:**
- Start : Start of the range
- End : End of the range
- Random number in [Begin, End]

Definition at line 88 of file **Randomizer.cpp**.

---

**float sf::Randomizer::Random ( float Begin, float End ) [static]**

Get a random float number in a given range.
**Returns:**

Start : Start of the range

End : End of the range

Random number in [Begin, End]

Definition at line 76 of file Randomizer.cpp.

```c
void sf::Randomizer::SetSeed ( unsigned int Seed ) [static]
```

Set the seed for the generator.

Using a known seed allows you to reproduce the same sequence of random number

**Parameters:**

Seed : Number to use as the seed

Using a known seed allows you to reproduce the same sequence of random number

Definition at line 57 of file Randomizer.cpp.

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

- [Randomizer.hpp](Randomizer.hpp)
- [Randomizer.cpp](Randomizer.cpp)

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::

Documentation generated by doxygen 1.5.2 ::
sf::Rect
sf::Rect< T > Class Template Reference

Rect is an utility class for manipulating rectangles. More...

#include <Rect.hpp>

List of all members.
**Public Member Functions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Rect ()</code></td>
<td>Default constructor.</td>
</tr>
<tr>
<td><code>Rect (T LeftCoord, T TopCoord, T RightCoord, T BottomCoord)</code></td>
<td>Construct the rectangle from its coordinates.</td>
</tr>
<tr>
<td><code>T GetWidth ()</code> const</td>
<td>Get the width of the rectangle.</td>
</tr>
<tr>
<td><code>T GetHeight ()</code> const</td>
<td>Get the height of the rectangle.</td>
</tr>
<tr>
<td><code>void Offset (T OffsetX, T OffsetY)</code></td>
<td>Move the whole rectangle by the given offset.</td>
</tr>
<tr>
<td><code>bool Contains (T X, T Y) const</code></td>
<td>Check if a point is inside the rectangle’s area.</td>
</tr>
<tr>
<td><code>bool Intersects (const Rect&lt;T&gt; &amp;Rectangle, Rect&lt;T&gt; *OverlappingRect=NULL) const</code></td>
<td>Check intersection between two rectangles.</td>
</tr>
</tbody>
</table>
## Public Attributes

<table>
<thead>
<tr>
<th>T</th>
<th>Left</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Left coordinate of the rectangle.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>T</th>
<th>Top</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Top coordinate of the rectangle.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>T</th>
<th>Right</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Right coordinate of the rectangle.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>T</th>
<th>Bottom</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Bottom coordinate of the rectangle.</td>
</tr>
</tbody>
</table>
Detailed Description

template<typename T>
class sf::Rect< T >

Rect is an utility class for manipulating rectangles.

Template parameter defines the type of coordinates (integer, float, ...)

Definition at line 41 of file Rect.hpp.
## Constructor & Destructor Documentation

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>template&lt;typename T&gt;</td>
<td>sf::Rect&lt;T&gt;::Rect()</td>
</tr>
<tr>
<td></td>
<td>Default constructor.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>template&lt;typename T&gt;</td>
<td>sf::Rect&lt;T&gt;::Rect(T LeftCoord, T TopCoord, T RightCoord, T BottomCoord)</td>
</tr>
<tr>
<td></td>
<td>Construct the rectangle from its coordinates.</td>
</tr>
</tbody>
</table>

**Parameters:**

- `LeftCoord`: Left coordinate of the rectangle
- `TopCoord`: Top coordinate of the rectangle
- `RightCoord`: Right coordinate of the rectangle
- `BottomCoord`: Bottom coordinate of the rectangle
Member Function Documentation

```cpp
template<typename T>
bool sf::Rect<T>::Contains ( T X,
                            T Y
                       ) const
```

Check if a point is inside the rectangle's area.

**Parameters:**
- \( X \) : X coordinate of the point to test
- \( Y \) : Y coordinate of the point to test

**Returns:**
True if the point is inside

```cpp
template<typename T>
T sf::Rect<T>::GetHeight ( ) const
```

Get the height of the rectangle.

**Returns:**
Height of rectangle

```cpp
template<typename T>
T sf::Rect<T>::GetWidth ( ) const
```

Get the width of the rectangle.

**Returns:**
Width of rectangle
template<typename T>

bool sf::Rect< T >::Intersects ( const Rect< T > & Rectangle, Rect< T > * OverlappingRect = NULL )

Check intersection between two rectangles.

**Parameters:**

- *Rectangle*: Rectangle to test
- *OverlappingRect*: Rectangle to be filled with overlapping rect (NULL by default)

**Returns:**
True if rectangles overlap

template<typename T>

void sf::Rect< T >::Offset ( T OffsetX, T OffsetY )

Move the whole rectangle by the given offset.

**Parameters:**

- *OffsetX*: Horizontal offset
- *OffsetY*: Vertical offset
### Member Data Documentation

<table>
<thead>
<tr>
<th>Template</th>
<th>Description</th>
<th>Definition Line</th>
<th>File</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>T sf::Rect&lt;T&gt;::Bottom</code></td>
<td>Bottom coordinate of the rectangle.</td>
<td>115</td>
<td>Rect.hpp</td>
</tr>
<tr>
<td><code>T sf::Rect&lt;T&gt;::Left</code></td>
<td>Left coordinate of the rectangle.</td>
<td>112</td>
<td>Rect.hpp</td>
</tr>
<tr>
<td><code>T sf::Rect&lt;T&gt;::Right</code></td>
<td>Right coordinate of the rectangle.</td>
<td>114</td>
<td>Rect.hpp</td>
</tr>
<tr>
<td><code>T sf::Rect&lt;T&gt;::Top</code></td>
<td>Top coordinate of the rectangle.</td>
<td>113</td>
<td>Rect.hpp</td>
</tr>
</tbody>
</table>

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

- **Rect.hpp**
sf::RenderTarget
sf::RenderTarget Class Reference

Base class for all render targets (window, image, . More...

```
#include <RenderTarget.hpp>
```

Inheritance diagram for sf::RenderTarget:

![Inheritance diagram](diagram.png)

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual ~RenderTarget ()</td>
<td>Destructor.</td>
</tr>
<tr>
<td>void Clear (const Color &amp;FillColor=Color(0, 0, 0))</td>
<td>Clear the entire target with a single color.</td>
</tr>
<tr>
<td>virtual void Draw (const Drawable &amp;Object)</td>
<td>Draw something into the target.</td>
</tr>
<tr>
<td>virtual unsigned int GetWidth () const =0</td>
<td>Get the width of the rendering region of the target.</td>
</tr>
<tr>
<td>virtual unsigned int GetHeight () const =0</td>
<td>Get the height of the rendering region of the target.</td>
</tr>
<tr>
<td>void SetView (const View &amp;NewView)</td>
<td>Change the current active view.</td>
</tr>
<tr>
<td>const View &amp; GetView () const</td>
<td>Get the current view.</td>
</tr>
<tr>
<td>View &amp; GetDefaultView ()</td>
<td>Get the default view of the window for read / write.</td>
</tr>
<tr>
<td>void PreserveOpenGLStates (bool Preserve)</td>
<td>Tell SFML to preserve external OpenGL states, at the expense of more CPU charge.</td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RenderTarget</td>
<td>Default constructor.</td>
</tr>
<tr>
<td>Initialize</td>
<td>Called by the derived class when it's ready to be initialized.</td>
</tr>
</tbody>
</table>
Detailed Description

Base class for all render targets (window, image, ..)

Definition at line 43 of file RenderTarget.hpp.
## Constructor & Destructor Documentation

### sf::RenderTarget::~RenderTarget ( ) [virtual]

Destructor.

Definition at line 51 of file `RenderTarget.cpp`.

### sf::RenderTarget::RenderTarget ( ) [protected]

Default constructor.

Definition at line 39 of file `RenderTarget.cpp`.  

---
### Member Function Documentation

**void sf::RenderTarget::Clear ( const Color & FillColor = color(0, 0, 0) )**

Clear the entire target with a single color.

**Parameters:**

FillColor : Color to use to clear the render target

Definition at line 60 of file RenderTarget.cpp.

**void sf::RenderTarget::Draw ( const Drawable & Object ) [virtual]**

Draw something into the target.

Draw something on the window.

**Parameters:**

Object : Object to draw

Definition at line 76 of file RenderTarget.cpp.

**View & sf::RenderTarget::GetDefaultView ( )**

Get the default view of the window for read / write.

**Returns:**

Default view

Definition at line 147 of file RenderTarget.cpp.
### virtual unsigned int sf::RenderTarget::GetHeight() const [pure virtual]

Get the height of the rendering region of the target.

**Returns:**
Height in pixels

Implemented in **sf::RenderWindow**.

### const View & sf::RenderTarget::GetView() const

Get the current view.

**Returns:**
Current view active in the window

Definition at line 138 of file **RenderTarget.cpp**.

### virtual unsigned int sf::RenderTarget::GetWidth() const [pure virtual]

Get the width of the rendering region of the target.

**Returns:**
Width in pixels

Implemented in **sf::RenderWindow**.

### void sf::RenderTarget::Initialize() [protected]

Called by the derived class when it's ready to be initialized.

Definition at line 170 of file **RenderTarget.cpp**.
void sf::RenderTarget::PreserveOpenGLStates ( bool  
Preserve  )

Tell SFML to preserve external OpenGL states, at the expense of more CPU charge.

Tell SFML to preserve external OpenGL states, at the expense of more CPU charge.

Use this function if you don’t want SFML to mess up your own OpenGL states (if any). Don’t enable state preservation if not needed, as it will allow SFML to do internal optimizations and improve performances. This parameter is false by default.

Parameters:

Preserve  : True to preserve OpenGL states, false to let SFML optimize

Definition at line 161 of file RenderTarget.cpp.

void sf::RenderTarget::SetView ( const View &  
NewView  )

Change the current active view.

Parameters:

NewView  : New view to use (pass GetDefaultView() to set the default view)

Definition at line 129 of file RenderTarget.cpp.
The documentation for this class was generated from the following files:

- **RenderTarget.hpp**
- **RenderTarget.cpp**
sf::RenderWindow
**sf::RenderWindow Class Reference**

Simple wrapper for *sf::Window* that allows easy 2D rendering.

More...

```cpp
#include <RenderWindow.hpp>
```

Inheritance diagram for sf::RenderWindow:

![Inheritance Diagram](image)

List of all members.
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>RenderWindow()</code></td>
<td>Default constructor.</td>
</tr>
<tr>
<td>`RenderWindow(VideoMode Mode, const std::string &amp;Title, unsigned long WindowStyle=Style::Resize</td>
<td>Style::Close, const WindowSettings &amp;Params=WindowSettings())`</td>
</tr>
<tr>
<td><code>RenderWindow(WindowHandle Handle, const WindowSettings &amp;Params=WindowSettings())</code></td>
<td>Construct the window from an existing control.</td>
</tr>
<tr>
<td><code>~RenderWindow()</code></td>
<td>Destructor.</td>
</tr>
<tr>
<td><code>virtual unsigned int GetWidth() const</code></td>
<td>Get the width of the rendering region of the window.</td>
</tr>
<tr>
<td><code>virtual unsigned int GetHeight() const</code></td>
<td>Get the height of the rendering region of the window.</td>
</tr>
<tr>
<td><code>Image Capture()</code></td>
<td>Save the content of the window to an image.</td>
</tr>
<tr>
<td><code>sf::Vector2f ConvertCoords(unsigned int WindowX, unsigned int WindowY, const View *TargetView=NULL)</code></td>
<td>Convert a point in window coordinates into view coordinates.</td>
</tr>
<tr>
<td>`void Create(VideoMode Mode, const std::string &amp;Title, unsigned long WindowStyle=Style::Resize</td>
<td>Style::Close, const WindowSettings &amp;Params=WindowSettings())`</td>
</tr>
<tr>
<td><code>void Create(WindowHandle Handle, const WindowSettings &amp;Params=WindowSettings())</code></td>
<td>Create (or recreate) the window from an existing control.</td>
</tr>
<tr>
<td><code>void Close()</code></td>
<td>Close (destroy) the window.</td>
</tr>
<tr>
<td><code>bool IsOpened() const</code></td>
<td>Tell whether or not the window is opened (ie.</td>
</tr>
<tr>
<td><code>const WindowSettings &amp; GetSettings() const</code></td>
<td>Get the creation settings of the window.</td>
</tr>
<tr>
<td><code>bool GetEvent(Event &amp;EventReceived)</code></td>
<td>Get the event on top of events stack, if any, and pop it.</td>
</tr>
<tr>
<td><code>void UseVerticalSync(bool Enabled)</code></td>
<td>Enable / disable vertical synchronization.</td>
</tr>
<tr>
<td><code>void ShowMouseCursor(bool Show)</code></td>
<td>Show or hide the mouse cursor.</td>
</tr>
<tr>
<td><code>void SetCursorPosition(unsigned int Left, unsigned int Top)</code></td>
<td>Change the position of the mouse cursor.</td>
</tr>
<tr>
<td><code>void SetPosition(int Left, int Top)</code></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>void SetSize(unsigned int Width, unsigned int Height)</code></td>
<td>Change the size of the rendering region of the window.</td>
</tr>
<tr>
<td><code>void Show(bool State)</code></td>
<td>Show or hide the window.</td>
</tr>
<tr>
<td><code>void EnableKeyRepeat(bool Enabled)</code></td>
<td>Enable or disable automatic key-repeat.</td>
</tr>
<tr>
<td><code>void SetIcon(unsigned int Width, unsigned int Height, const Uint8 *Pixels)</code></td>
<td>Change the window's icon.</td>
</tr>
<tr>
<td><code>bool SetActive(bool Active=true)</code> const</td>
<td>Activate or deactivate the window as the current target for rendering.</td>
</tr>
<tr>
<td><code>void Display()</code></td>
<td>Display the window on screen.</td>
</tr>
<tr>
<td><code>const Input &amp; GetInput()</code> const</td>
<td>Get the input manager of the window.</td>
</tr>
<tr>
<td><code>void SetFramerateLimit(unsigned int Limit)</code></td>
<td>Limit the framerate to a maximum fixed frequency.</td>
</tr>
<tr>
<td><code>float GetFrameTime()</code> const</td>
<td>Get time elapsed since last frame.</td>
</tr>
<tr>
<td><code>void SetJoystickThreshold(float Threshold)</code></td>
<td>Change the joystick threshold, i.e.</td>
</tr>
<tr>
<td><code>void Clear(const Color &amp;FillColor=Color(0, 0, 0))</code></td>
<td>Clear the entire target with a single color.</td>
</tr>
<tr>
<td><code>virtual void Draw(const Drawable &amp;Object)</code></td>
<td>Draw something into the target.</td>
</tr>
<tr>
<td><code>void SetView(const View &amp;NewView)</code></td>
<td>Change the current active view.</td>
</tr>
<tr>
<td><code>const View &amp; GetView()</code> const</td>
<td>Get the current view.</td>
</tr>
<tr>
<td><code>View &amp; GetDefaultView()</code></td>
<td>Get the default view of the window for read / write.</td>
</tr>
<tr>
<td><code>void PreserveOpenGLStates(bool Preserve)</code></td>
<td>Tell SFML to preserve external OpenGL states, at the expense of more CPU charge.</td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>void</th>
<th>Initialize ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>Called by the derived class when it's ready to be initialized.</em></td>
</tr>
</tbody>
</table>
Detailed Description

Simple wrapper for sf::Window that allows easy 2D rendering.

Definition at line 45 of file RenderWindow.hpp.
Constructor & Destructor Documentation

sf::RenderWindow::RenderWindow ( )

Default constructor.

Definition at line 40 of file RenderWindow.cpp.

sf::RenderWindow::RenderWindow ( VideoMode  
    const std::string &  
    unsigned long  
    const WindowSettings &  
    Params = WindowSettings )

Construct the window.

Parameters:

  Mode : Video mode to use  
  Title : Title of the window  
  WindowStyle : Window style, see sf::Style (Resize | Close by default)  
  Params : Creation parameters (see default constructor for default values)

Definition at line 49 of file RenderWindow.cpp.

sf::RenderWindow::RenderWindow ( WindowHandle  
    const WindowSettings &  
    Params = WindowSettings )

Construct the window from an existing control.

Parameters:
**Handle**: Platform-specific handle of the control
**Params**: Creation parameters (see default constructor for default values)

Definition at line **58** of file *RenderWindow.cpp*.

```
sf::RenderWindow::~RenderWindow() [virtual]
```

Destructor.

Definition at line **67** of file *RenderWindow.cpp*. 
**Member Function Documentation**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sf::RenderWindow::Capture</code></td>
<td>Save the content of the window to an image. Returns: <code>Image</code> instance containing the contents of the screen</td>
<td>Definition at line 107 of file <code>RenderWindow.cpp</code>.</td>
</tr>
<tr>
<td><code>sf::RenderTarget::Clear</code></td>
<td>Clear the entire target with a single color. Parameters: <code>FillColor</code> : <code>Color</code> to use to clear the render target</td>
<td>Definition at line 60 of file <code>RenderTarget.cpp</code>.</td>
</tr>
<tr>
<td><code>sf::Window::Close</code></td>
<td>Close (destroy) the window. The <code>sf::Window</code> instance remains valid and you can call Create to recreate the window</td>
<td>Definition at line 165 of file <code>Window.cpp</code>.</td>
</tr>
</tbody>
</table>
sf::Vector2f sf::RenderWindow::ConvertCoords ( unsigned int WindowX, unsigned int WindowY, const View * TargetView = )

Convert a point in window coordinates into view coordinates.

**Parameters:**
- **WindowX**: X coordinate of the point to convert, relative to the window
- **WindowY**: Y coordinate of the point to convert, relative to the window
- **TargetView**: Target view to convert the point to (NULL by default -- uses the current view)

**Returns:**
Converted point

Definition at line 142 of file RenderWindow.cpp.

void sf::Window::Create ( WindowHandle Handle, const WindowSettings & Params = WindowSettings )

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

Create the window from an existing control.

**Parameters:**
- **Handle**: Platform-specific handle of the control
- **Params**: Creation parameters (see default constructor for default values)

Definition at line 147 of file Window.cpp.

void sf::Window::Create ( VideoMode Mode, const std::string & Title,
unsigned long WindowStyle = Style::Resize | Style::Close
const WindowSettings & Params = WindowSettings()
)

Create (or recreate) the window.
Create the window.

**Parameters:**

- **Mode**: Video mode to use
- **Title**: Title of the window
- **WindowStyle**: Window style, see `sf::Style` (Resize | Close by default)
- **Params**: Creation parameters (see default constructor for default values)

Definition at line 104 of file Window.cpp.

```cpp
void sf::Window::Display ( ) [inherited]
```

Display the window on screen.

Definition at line 353 of file Window.cpp.

```cpp
void sf::RenderTarget::Draw ( const Drawable & Object ) [virtual, inherited]
```

Draw something into the target.
Draw something on the window.

**Parameters:**

- **Object**: Object to draw

Definition at line 76 of file RenderTarget.cpp.
### void sf::Window::EnableKeyRepeat ( bool Enabled ) [inherited]

Enable or disable automatic key-repeat.

Automatic key-repeat is enabled by default

**Parameters:**

*Enabled*: True to enable, false to disable

Automatic key-repeat is enabled by default

Definition at line **317** of file **Window.cpp**.

---

### View & sf::RenderTarget::GetDefaultView ( ) [inherited]

Get the default view of the window for read / write.

**Returns:**

Default view

Definition at line **147** of file **RenderTarget.cpp**.

---

### bool sf::Window::GetEvent ( Event & EventReceived ) [inherited]

Get the event on top of events stack, if any, and pop it.

Get the event on top of events stack, if any.

**Parameters:**

*EventReceived*: Event to fill, if any

**Returns:**

True if an event was returned, false if events stack was
float sf::Window::GetFrameTime() const [inherited]

Get time elapsed since last frame.

**Returns:**
- Time elapsed, in seconds

Definition at line 394 of file Window.cpp.

unsigned int sf::RenderWindow::GetHeight() const [virtual]

Get the height of the rendering region of the window.

**Returns:**
- Height in pixels

Implements sf::RenderTarget.

Definition at line 98 of file RenderWindow.cpp.

const Input & sf::Window::GetInput() const [inherited]

Get the input manager of the window.

**Returns:**
- Reference to the input

Definition at line 376 of file Window.cpp.
**const WindowSettings & sf::Window::GetSettings ( ) const [inherited]**

Get the creation settings of the window.

**Returns:**
Structure containing the creation settings

Definition at line 209 of file `Window.cpp`.

**const View & sf::RenderTarget::GetView ( ) const [inherited]**

Get the current view.

**Returns:**
Current view active in the window

Definition at line 138 of file `RenderTarget.cpp`.

**unsigned int sf::RenderWindow::GetWidth ( ) const [virtual]**

Get the width of the rendering region of the window.

**Returns:**
Width in pixels

Implements `sf::RenderTarget`.

Definition at line 89 of file `RenderWindow.cpp`.

**void sf::RenderTarget::Initialize ( ) [protected, inherited]**
Called by the derived class when it's ready to be initialized.

Definition at line 170 of file RenderTarget.cpp.

```cpp
bool sf::Window::IsOpened() const [inherited]
```

Tell whether or not the window is opened (ie. has been created). Note that a hidden window (Show(false)) will still return true

**Returns:**
- True if the window is opened

has been created). Note that a hidden window (Show(false)) will still return true

Definition at line 182 of file Window.cpp.

```cpp
void sf::RenderTarget::PreserveOpenGLStates(bool Preserve) [inherited]
```

Tell SFML to preserve external OpenGL states, at the expense of more CPU charge.

Tell SFML to preserve external OpenGL states, at the expense of more CPU charge.

Use this function if you don't want SFML to mess up your own OpenGL states (if any). Don't enable state preservation if not needed, as it will allow SFML to do internal optimizations and improve performances. This parameter is false by default.

**Parameters:**
Preserve : True to preserve OpenGL states, false to let SFML optimize

Use this function if you don't want SFML to mess up your own OpenGL states (if any). Don't enable state preservation if not needed, as it will allow SFML to do internal optimizations and improve performances. This parameter is false by default.

Definition at line 161 of fileRenderTarget.cpp.

```cpp
bool sf::Window::SetActive ( bool Active = true ) const [inherited]
```

Activate of deactivate the window as the current target for rendering.

Activate of deactivate the window as the current target for rendering.

**Parameters:**

  * **Active** : True to activate, false to deactivate (true by default)

**Returns:**

  True if operation was successful, false otherwise

Definition at line 338 of fileWindow.cpp.

```cpp
void sf::Window::SetCursorPosition ( unsigned int Left, unsigned int Top ) [inherited]
```

Change the position of the mouse cursor.

**Parameters:**

  * **Left** : Left coordinate of the cursor, relative to the window
Top : Top coordinate of the cursor, relative to the window

Definition at line 260 of file Window.cpp.

```cpp
void sf::Window::SetFramerateLimit ( unsigned int Limit ) [inherited]
```

Limit the framerate to a maximum fixed frequency.

Set the framerate at a fixed frequency.

**Parameters:**

- **Limit**: Framerate limit, in frames per seconds (use 0 to disable limit)

Definition at line 385 of file Window.cpp.

```cpp
void sf::Window::SetIcon ( unsigned int Width,
             unsigned int Height,
             const Uint8 * Pixels ) [inherited]
```

Change the window's icon.

**Parameters:**

- **Width**: Icon's width, in pixels
- **Height**: Icon's height, in pixels
- **Pixels**: Pointer to the pixels in memory, format must be RGBA 32 bits

Definition at line 327 of file Window.cpp.

```cpp
void sf::Window::SetJoystickThreshold ( float Threshold ) [inherited]
```

Change the joystick threshold, ie.
the value below which no move event will be generated

**Parameters:**

*Threshold*: New threshold, in range [0, 100]

Definition at line 404 of file Window.cpp.

```cpp
void sf::Window::SetPosition(int Left, int Top) [inherited]
```

Change the position of the window on screen.

Only works for top-level windows

**Parameters:**

*Left*: Left position

*Top*: Top position

Definition at line 276 of file Window.cpp.

```cpp
void sf::Window::SetSize(unsigned int Width, unsigned int Height) [inherited]
```

Change the size of the rendering region of the window.

**Parameters:**

*Width*: New width

*Height*: New height
void sf::RenderTarget::SetView ( const View & NewView ) [inherited]

Change the current active view.

Parameters:

\[
\text{NewView} : \text{New view to use (pass GetDefaultView() to set the default view)}
\]

Definition at line 129 of file RenderTarget.cpp.

void sf::Window::Show ( bool State ) [inherited]

Show or hide the window.

Parameters:

\[
\text{State} : \text{True to show, false to hide}
\]

Definition at line 303 of file Window.cpp.

void sf::Window::ShowMouseCursor ( bool Show ) [inherited]

Show or hide the mouse cursor.

Parameters:

\[
\text{Show} : \text{True to show, false to hide}
\]

Definition at line 250 of file Window.cpp.

void sf::Window::UseVerticalSync ( bool Enabled ) [inherited]
Enable / disable vertical synchronization.

**Parameters:**

*Enabled*: True to enable v-sync, false to deactivate

Definition at line 240 of file *Window.cpp*.

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

- **RenderWindow.hpp**
- **RenderWindow.cpp**
sf::Resource
sf::Resource< T > Class Template Reference

Base class for every resource that needs to notify dependent classes about its destruction. More...

#include <Resource.hpp>

List of all members.
### Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Resource()</code></td>
<td>Default constructor.</td>
</tr>
<tr>
<td><code>~Resource()</code></td>
<td>Destructor.</td>
</tr>
<tr>
<td><code>Resource&lt;T&gt;&amp; operator=(const Resource&lt;T&gt;&amp; Other)</code></td>
<td>Assignment operator.</td>
</tr>
</tbody>
</table>
**Friends**

<table>
<thead>
<tr>
<th>class</th>
<th>ResourcePtr&lt; T &gt;</th>
</tr>
</thead>
</table>


Detailed Description

template<typename T>
class sf::Resource< T >

Base class for every resource that needs to notify dependent classes about its destruction.

Definition at line 50 of file Resource.hpp.
Constructor & Destructor Documentation

template<typename T>

sf::Resource<T>::Resource() [protected]

Default constructor.

template<typename T>

sf::Resource<T>::Resource(const Resource<T> & Copy) [protected]

Copy constructor.

**Parameters:**

*Copy*: Resource to copy

template<typename T>

sf::Resource<T>::~Resource() [protected]

Destructor.
Assignment operator.

**Parameters:**

*Other*: `Resource` to copy

**Returns:**

Reference to this

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

- **Resource.hpp**

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::

Documentation generated by doxygen 1.5.2 ::
sf::ResourcePtr
sf::ResourcePtr< T > Class Template Reference

Safe pointer to a T resource (inheriting from sf::Resource<T>), its pointer is automatically reseted when the resource is destroyed. More...

#include <Resource.hpp>

List of all members.
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ResourcePtr()</code></td>
<td>Default constructor.</td>
</tr>
<tr>
<td>~<code>ResourcePtr()</code></td>
<td>Destructor.</td>
</tr>
<tr>
<td><code>operator= (const T *Resource)</code></td>
<td>Assignment operator from a raw resource.</td>
</tr>
<tr>
<td><code>operator const T * () const</code></td>
<td>Cast operator to implicitly convert the resource pointer to its raw pointer type.</td>
</tr>
<tr>
<td><code>operator* () const</code></td>
<td>Operator * overload to return a reference to the actual resource.</td>
</tr>
<tr>
<td><code>operator-&gt; () const</code></td>
<td>Operator -&gt; overload to return a pointer to the actual resource.</td>
</tr>
<tr>
<td><code>OnResourceDestroyed ()</code></td>
<td>Function called when the observed resource is about to be destroyed.</td>
</tr>
</tbody>
</table>
**Detailed Description**

`template<typename T>
class sf::ResourcePtr< T >`

Safe pointer to a T resource (inheriting from sf::Resource<T>), its pointer is automatically reseted when the resource is destroyed.

Definition at line 116 of file Resource.hpp.
Constructor & Destructor Documentation

```
template<typename T>
sf::ResourcePtr<T>::ResourcePtr( )
```

Default constructor.

```
template<typename T>
sf::ResourcePtr<T>::ResourcePtr(const T* Resource)
```

Construct from a raw resource.

**Parameters:**
- `Resource`: Internal resource

```
template<typename T>
sf::ResourcePtr<T>::ResourcePtr(const ResourcePtr<T>& Copy)
```

Copy constructor.

**Parameters:**
- `Copy`: Instance to copy

```
template<typename T>
sf::ResourcePtr<T>::~ResourcePtr( )
```

Destructor.
# Member Function Documentation

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>template&lt;typename T&gt; void sf::ResourcePtr&lt;T&gt;::OnResourceDestroyed()</code></td>
<td>Function called when the observed resource is about to be destroyed.</td>
</tr>
<tr>
<td><code>template&lt;typename T&gt; sf::ResourcePtr&lt;T&gt;::operator const T *( ) const</code></td>
<td>Cast operator to implicitly convert the resource pointer to its raw pointer type. This might be dangerous in the general case, but in this context it is safe enough to define this operator. <strong>Returns:</strong> Pointer to the actual resource</td>
</tr>
<tr>
<td><code>template&lt;typename T&gt; const T&amp; sf::ResourcePtr&lt;T&gt;::operator*( ) const</code></td>
<td>Operator * overload to return a reference to the actual resource. <strong>Returns:</strong> Reference to the internal resource</td>
</tr>
<tr>
<td><code>template&lt;typename T&gt; const T* sf::ResourcePtr&lt;T&gt;::operator-&gt;( ) const</code></td>
<td></td>
</tr>
</tbody>
</table>


Operator -> overload to return a pointer to the actual resource.

**Returns:**
- Pointer to the internal resource

```
template<typename T>
ResourcePtr<T>& sf::ResourcePtr<T>::operator= (const T* Resource )
```

Assignment operator from a raw resource.

**Parameters:**
- `Resource`: Resource to assign

**Returns:**
- Reference to this

```
template<typename T>
ResourcePtr<T>& sf::ResourcePtr<T>::operator= (const ResourcePtr<T>& Other )
```

Assignment operator from another `ResourcePtr`.

**Parameters:**
- `Other`: Resource pointer to assign

**Returns:**
- Reference to this

---

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

- `Resource.hpp`
sf::Selector
sf::Selector< Type > Class Template Reference

**Selector** allow reading from multiple sockets without blocking. [More...](#)

```cpp
#include <Selector.hpp>
```

Inheritance diagram for sf::Selector< Type >:

[sf::SelectorBase](#) -> sf::Selector< Type >

List of all members.
Public Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>Add (Type Socket)</td>
<td>Add a socket to watch.</td>
</tr>
<tr>
<td>void</td>
<td>Remove (Type Socket)</td>
<td>Remove a socket.</td>
</tr>
<tr>
<td>void</td>
<td>Clear ()</td>
<td>Remove all sockets.</td>
</tr>
<tr>
<td>unsigned int</td>
<td>Wait (float Timeout=0.f)</td>
<td>Wait and collect sockets which are ready for reading.</td>
</tr>
<tr>
<td>Type</td>
<td>GetSocketReady (unsigned int Index)</td>
<td>After a call to Wait(), get the Index-th socket which is ready for reading.</td>
</tr>
</tbody>
</table>
## Private Member Functions

<table>
<thead>
<tr>
<th>void</th>
<th><strong>Add</strong> (SocketHelper::SocketType Socket)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>Add a socket to watch.</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>Remove</strong> (SocketHelper::SocketType Socket)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>Remove a socket.</em></td>
</tr>
</tbody>
</table>
Detailed Description

template<typename Type>
class sf::Selector< Type >

Selector allow reading from multiple sockets without blocking.

It's a kind of multiplexer

Definition at line 44 of file Selector.hpp.
**Member Function Documentation**

<table>
<thead>
<tr>
<th>Function</th>
<th>Documentation</th>
</tr>
</thead>
</table>
| Add | `template<typename Type >
void sf::Selector< Type >::Add ( Type Socket )`

Add a socket to watch.

**Parameters:**

- **Socket**: Socket to add

| Clear | `template<typename Type >
void sf::Selector< Type >::Clear ( )`

Remove all sockets.

Reimplemented from `sf::SelectorBase`.

| GetSocketReady | `template<typename Type >
Type sf::Selector< Type >::GetSocketReady ( unsigned int Index )`

After a call to `Wait()`, get the Index-th socket which is ready for reading.

The total number of sockets ready is the integer returned by the previous call to `Wait()`

**Parameters:**

- **Index**: Index of the socket to get

**Returns:**

The Index-th socket

Reimplemented from `sf::SelectorBase`. |
### Remove

Remove a socket.

**Parameters:**
- `Socket`: Socket to remove

### Wait

Wait and collect sockets which are ready for reading.

This functions will return either when at least one socket is ready, or when the given time is out.

**Parameters:**
- `Timeout`: Timeout, in seconds (0 by default: no timeout)

**Returns:**
- Number of sockets ready to be read

Reimplemented from `sf::SelectorBase`.

---

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

- **Selector.hpp**
sf::SelectorBase
sf::SelectorBase Class Reference

Private base class for selectors. [More...](#)

```cpp
#include <SelectorBase.hpp>
```

Inheritance diagram for sf::SelectorBase:

< Type >" shape="rect" coords="0,56,127,80">

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>SelectorBase()</code></td>
<td>Default constructor.</td>
</tr>
<tr>
<td><code>void Add(SocketHelper::SocketType Socket)</code></td>
<td>Add a socket to watch.</td>
</tr>
<tr>
<td><code>void Remove(SocketHelper::SocketType Socket)</code></td>
<td>Remove a socket.</td>
</tr>
<tr>
<td><code>void Clear()</code></td>
<td>Remove all sockets.</td>
</tr>
<tr>
<td><code>unsigned int Wait(float Timeout=0.f)</code></td>
<td>Wait and collect sockets which are ready for reading.</td>
</tr>
<tr>
<td><code>SocketHelper::SocketType GetSocketReady(unsigned int Index)</code></td>
<td>After a call to <code>Wait()</code>, get the Index-th socket which is ready for reading.</td>
</tr>
</tbody>
</table>
Detailed Description

Private base class for selectors.

As `Selector` is a template class, this base is needed so that every system call get compiled in SFML (not inlined)

Definition at line 43 of file `SelectorBase.hpp`. 

-----------
## Constructor & Destructor Documentation

<table>
<thead>
<tr>
<th>sf::SelectorBase::SelectorBase ( )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default constructor.</td>
</tr>
<tr>
<td>Definition at line 40 of file <strong>SelectorBase.cpp</strong>.</td>
</tr>
</tbody>
</table>
Member Function Documentation

**void sf::SelectorBase::Add ( SocketHelper::SocketType Socket )**

Add a socket to watch.

**Parameters:**

*Socket*: Socket to add

Definition at line 50 of file *SelectorBase.cpp*.

**void sf::SelectorBase::Clear ( )**

Remove all sockets.

Reimplemented in *sf::Selector< Type >.*

Definition at line 72 of file *SelectorBase.cpp*.

**SocketHelper::SocketType sf::SelectorBase::GetSocketReady ( unsigned int Index )**

After a call to *Wait()*, get the Index-th socket which is ready for reading.

After a call to *Wait()*, get the Index-th socket which is ready for reading.

The total number of sockets ready is the integer returned by the previous call to *Wait()*

**Parameters:**

*Index*: Index of the socket to get
Returns:
The Index-th socket

The total number of sockets ready is the integer returned by the previous call to `Wait()`

Reimplemented in `sf::Selector< Type >`.

Definition at line 108 of file `SelectorBase.cpp`.

```cpp
void sf::SelectorBase::Remove ( SocketHelper::SocketType Socket )
```

Remove a socket.

**Parameters:**

- `Socket` : Socket to remove

Definition at line 63 of file `SelectorBase.cpp`.

```cpp
unsigned int sf::SelectorBase::Wait ( float Timeout = 0.f )
```

Wait and collect sockets which are ready for reading.

This functions will return either when at least one socket is ready, or when the given time is out

**Parameters:**

- `Timeout` : Timeout, in seconds (0 by default : no timeout)

**Returns:**

- Number of sockets ready to be read

This functions will return either when at least one socket is
ready, or when the given time is out

Reimplemented in \texttt{sf::Selector< Type >}.

Definition at line 86 of file \texttt{SelectorBase.cpp}.

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

- \texttt{SelectorBase.hpp}
- \texttt{SelectorBase.cpp}
sf::Shape
sf::Shape Class Reference

**Shape** defines a drawable convex shape; it also defines helper functions to draw simple shapes like lines, rectangles, circles, etc.  

#include <Shape.hpp>

Inheritance diagram for sf::Shape:

![Inheritance Diagram](image)

[List of all members.](#)
## Classes

<table>
<thead>
<tr>
<th>struct</th>
<th>Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defines a simple 2D point.</td>
<td></td>
</tr>
</tbody>
</table>
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Shape</strong> ()</td>
<td>Default constructor.</td>
</tr>
<tr>
<td>void AddPoint (float X, float Y, const Color &amp;Col=Color(255, 255, 255), const Color &amp;OutlineCol=Color(0, 0, 0))</td>
<td>Add a point to the shape.</td>
</tr>
<tr>
<td>void AddPoint (const Vector2f &amp;Position, const Color &amp;Col=Color(255, 255, 255), const Color &amp;OutlineCol=Color(0, 0, 0))</td>
<td>Add a point to the shape.</td>
</tr>
<tr>
<td>unsigned int GetNbPoints () const</td>
<td>Get the number of points composing the shape.</td>
</tr>
<tr>
<td>void EnableFill (bool Enable)</td>
<td>Enable or disable filling the shape.</td>
</tr>
<tr>
<td>void EnableOutline (bool Enable)</td>
<td>Enable or disable drawing the shape outline.</td>
</tr>
<tr>
<td>void SetPointPosition (unsigned int Index, const Vector2f &amp;Position)</td>
<td>Set the position of a point.</td>
</tr>
<tr>
<td>void SetPointPosition (unsigned int Index, float X, float Y)</td>
<td>Set the position of a point.</td>
</tr>
<tr>
<td>void SetPointColor (unsigned int Index, const Color &amp;Col)</td>
<td>Set the color of a point.</td>
</tr>
<tr>
<td>void SetPointOutlineColor (unsigned int Index, const Color &amp;OutlineCol)</td>
<td>Set the outline color of a point.</td>
</tr>
<tr>
<td>void SetOutlineWidth (float Width)</td>
<td>Change the width of the shape outline.</td>
</tr>
<tr>
<td>const Vector2f &amp; GetPointPosition (unsigned int Index) const</td>
<td>Get the position of a point.</td>
</tr>
<tr>
<td>const Color &amp; GetPointColor (unsigned int Index) const</td>
<td>Get the color of a point.</td>
</tr>
<tr>
<td>const Color &amp; GetPointOutlineColor (unsigned int Index) const</td>
<td>Get the outline color of a point.</td>
</tr>
<tr>
<td>float GetOutlineWidth () const</td>
<td>Get the width of the shape outline.</td>
</tr>
<tr>
<td>void SetPosition (float X, float Y)</td>
<td>Set the position of the object (take 2 values).</td>
</tr>
<tr>
<td>void SetPosition (const Vector2f &amp;Position)</td>
<td>Set the position of the object (take a 2D vector).</td>
</tr>
<tr>
<td>void SetX (float X)</td>
<td>Set the X position of the object.</td>
</tr>
<tr>
<td>void SetY (float Y)</td>
<td>Set the Y position of the object.</td>
</tr>
<tr>
<td>void SetScale (float ScaleX, float ScaleY)</td>
<td>Set the scale of the object (take 2 values).</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>SetScale</code> (const <code>Vector2f</code> &amp;Scale)</td>
<td>Set the scale of the object (take 2 values).</td>
</tr>
<tr>
<td><code>SetScaleX</code> (float FactorX)</td>
<td>Set the X scale factor of the object.</td>
</tr>
<tr>
<td><code>SetScaleY</code> (float FactorY)</td>
<td>Set the Y scale factor of the object.</td>
</tr>
<tr>
<td><code>SetCenter</code> (float centerX, float centerY)</td>
<td>Set the center of the object, in coordinates relative to the top-left of the object (take 2 values).</td>
</tr>
<tr>
<td><code>SetCenter</code> (const <code>Vector2f</code> &amp;Center)</td>
<td>Set the center of the object, in coordinates relative to the top-left of the object (take a 2D vector).</td>
</tr>
<tr>
<td><code>SetRotation</code> (float Rotation)</td>
<td>Set the orientation of the object.</td>
</tr>
<tr>
<td><code>SetColor</code> (const <code>Color</code> &amp;Col)</td>
<td>Set the color of the object.</td>
</tr>
<tr>
<td><code>SetBlendMode</code> (<code>Blend::Mode</code> Mode)</td>
<td>Set the blending mode for the object.</td>
</tr>
<tr>
<td><code>GetPosition</code> () const</td>
<td>Get the position of the object.</td>
</tr>
<tr>
<td><code>GetScale</code> () const</td>
<td>Get the current scale of the object.</td>
</tr>
<tr>
<td><code>GetCenter</code> () const</td>
<td>Get the center of the object.</td>
</tr>
<tr>
<td><code>GetRotation</code> () const</td>
<td>Get the orientation of the object.</td>
</tr>
<tr>
<td><code>GetColor</code> () const</td>
<td>Get the color of the object.</td>
</tr>
<tr>
<td><code>GetBlendMode</code> () const</td>
<td>Get the current blending mode.</td>
</tr>
<tr>
<td><code>Move</code> (float OffsetX, float OffsetY)</td>
<td>Move the object of a given offset (take 2 values).</td>
</tr>
<tr>
<td><code>Move</code> (const <code>Vector2f</code> &amp;Offset)</td>
<td>Move the object of a given offset (take a 2D vector).</td>
</tr>
<tr>
<td><code>Scale</code> (float FactorX, float FactorY)</td>
<td>Scale the object (take 2 values).</td>
</tr>
<tr>
<td><code>Scale</code> (const <code>Vector2f</code> &amp;Factor)</td>
<td>Scale the object (take a 2D vector).</td>
</tr>
<tr>
<td><code>Rotate</code> (float Angle)</td>
<td>Rotate the object.</td>
</tr>
<tr>
<td><code>TransformToLocal</code> (const <code>sf::Vector2f</code> &amp;Point) const</td>
<td>Transform a point from global coordinates into local coordinates (ie it applies the inverse of object's center, translation, rotation and scale to the point).</td>
</tr>
<tr>
<td><code>TransformToGlobal</code> (const <code>sf::Vector2f</code> &amp;Point) const</td>
<td>Transform a point from local coordinates into global coordinates (ie it applies the inverse of object's center, translation, rotation and scale to the point).</td>
</tr>
</tbody>
</table>
Transform a point from local coordinates into global coordinates (ie it applies the object's center, translation, rotation and scale to the point).
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Parameters</th>
<th>Create a shape made of a single...</th>
</tr>
</thead>
<tbody>
<tr>
<td>static <strong>Shape</strong></td>
<td>Line (float P1X, float P1Y, float P2X, float P2Y, float Thickness, const <strong>Color</strong> &amp;Col, float Outline=0.f, const <strong>Color</strong> &amp;OutlineCol=<strong>sf::Color</strong>(0, 0, 0))</td>
<td>made of a single line (use floats).</td>
</tr>
<tr>
<td>static <strong>Shape</strong></td>
<td>Line (const <strong>Vector2f</strong> &amp;P1, const <strong>Vector2f</strong> &amp;P2, float Thickness, const <strong>Color</strong> &amp;Col, float Outline=0.f, const <strong>Color</strong> &amp;OutlineCol=<strong>sf::Color</strong>(0, 0, 0))</td>
<td>made of a single line (use vectors).</td>
</tr>
<tr>
<td>static <strong>Shape</strong></td>
<td><strong>Rectangle</strong> (float P1X, float P1Y, float P2X, float P2Y, const <strong>Color</strong> &amp;Col, float Outline=0.f, const <strong>Color</strong> &amp;OutlineCol=<strong>sf::Color</strong>(0, 0, 0))</td>
<td>made of a single rectangle (use floats).</td>
</tr>
<tr>
<td>static <strong>Shape</strong></td>
<td><strong>Rectangle</strong> (const <strong>Vector2f</strong> &amp;P1, const <strong>Vector2f</strong> &amp;P2, const <strong>Color</strong> &amp;Col, float Outline=0.f, const <strong>Color</strong> &amp;OutlineCol=<strong>sf::Color</strong>(0, 0, 0))</td>
<td>made of a single rectangle (use vectors).</td>
</tr>
<tr>
<td>static <strong>Shape</strong></td>
<td><strong>Circle</strong> (float X, float Y, float Radius, const <strong>Color</strong> &amp;Col, float Outline=0.f, const <strong>Color</strong> &amp;OutlineCol=<strong>sf::Color</strong>(0, 0, 0))</td>
<td>made of a single circle (use floats).</td>
</tr>
<tr>
<td>static <strong>Shape</strong></td>
<td><strong>Circle</strong> (const <strong>Vector2f</strong> &amp;Center, float Radius, const <strong>Color</strong> &amp;Col, float Outline=0.f, const <strong>Color</strong> &amp;OutlineCol=<strong>sf::Color</strong>(0, 0, 0))</td>
<td>made of a single circle (use vectors).</td>
</tr>
</tbody>
</table>
Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual void Render (RenderTarget &amp;Target) const</td>
<td>/see Drawable::Render</td>
</tr>
<tr>
<td>const Matrix3 GetMatrix () const</td>
<td>Get the transform matrix of the drawable.</td>
</tr>
<tr>
<td>const Matrix3 GetInverseMatrix () const</td>
<td>Get the inverse transform matrix of the drawable.</td>
</tr>
</tbody>
</table>
Detailed Description

**Shape** defines a drawable convex shape; it also defines helper functions to draw simple shapes like lines, rectangles, circles, etc.

Definition at line 43 of file *Shape.hpp*. 
## Constructor & Destructor Documentation

<table>
<thead>
<tr>
<th>sf::Shape::Shape ( )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default constructor.</td>
</tr>
<tr>
<td>Definition at line 38 of file Shape.cpp.</td>
</tr>
</tbody>
</table>
Member Function Documentation

void sf::Shape::AddPoint ( const Vector2f & Position,
    const Color & Col = color(255, 255, 255),
    const Color & OutlineCol = color(0, 0, 0) )

Add a point to the shape.

Parameters:

  Position : Position of the point
  Col      : Color of the point (white by default)
  OutlineCol : Outline color of the point (black by default)

Definition at line 61 of file Shape.cpp.

Shape sf::Shape::Circle ( const Vector2f & Center,
Create a shape made of a single circle (use vectors).

**Parameters:**
- **Center**: Position of the center
- **Radius**: Radius
- **Col**: Color used to fill the circle
- **Outline**: Outline width (0 by default)
- **OutlineCol**: Color used to draw the outline (black by default)

Definition at line 276 of file Shape.cpp.

Create a shape made of a single circle (use floats).

Create a shape made of a single circle.

**Parameters:**
- **X, Y**: Position of the center
- **Radius**: Radius
- **Col**: Color used to fill the circle
- **Outline**: Outline width (0 by default)
- **OutlineCol**: Color used to draw the outline (black by default)
Definition at line 250 of file Shape.cpp.

```cpp
void sf::Shape::EnableFill ( bool Enable )
```

Enable or disable filling the shape.

Fill is enabled by default

**Parameters:**

*Enable*: True to enable, false to disable

Fill is enabled by default

Definition at line 81 of file Shape.cpp.

---

```cpp
void sf::Shape::EnableOutline ( bool Enable )
```

Enable or disable drawing the shape outline.

Outline is enabled by default

**Parameters:**

*Enable*: True to enable, false to disable

Outline is enabled by default

Definition at line 91 of file Shape.cpp.

---

```cpp
Blend::Mode sf::Drawable::GetBlendMode ( ) const [inherited]
```

Get the current blending mode.
Returns:
Current blending mode

Definition at line 258 of file Drawable.cpp.

const Vector2f & sf::Drawable::GetCenter() const [inherited]

Get the center of the object.

Returns:
Current position of the center

Definition at line 231 of file Drawable.cpp.

const Color & sf::Drawable::GetColor() const [inherited]

Get the color of the object.

Returns:
Current color

Definition at line 249 of file Drawable.cpp.

const Matrix3 & sf::Drawable::GetInverseMatrix() const [protected, inherited]

Get the inverse transform matrix of the drawable.

Returns:
Inverse transform matrix

Definition at line 350 of file Drawable.cpp.
**const Matrix3 & sf::Drawable::GetMatrix() const [protected, inherited]**

Get the transform matrix of the drawable.

**Returns:**
Transform matrix

Definition at line 334 of file **Drawable.cpp**.

**unsigned int sf::Shape::GetNbPoints() const**

Get the number of points composing the shape.

**Parameters:**
- Total number of points

Definition at line 71 of file **Shape.cpp**.

**float sf::Shape::GetOutlineWidth() const**

Get the width of the shape outline.

**Returns:**
- Current outline width

Definition at line 175 of file **Shape.cpp**.

**const Color & sf::Shape::GetPointColor(unsigned int Index) const**

Get the color of a point.

**Parameters:**
Index : Index of the point, in range [0, GetNbPoints() - 1]

**Returns:**

*Color* of the Index-th point

Definition at line 157 of file *Shape.cpp.*

const *Color* & sf::Shape::GetPointOutlineColor ( unsigned int *Index* ) const

Get the outline color of a point.

**Parameters:**

*Index* : Index of the point, in range [0, GetNbPoints() - 1]

**Returns:**

Outline color of the Index-th point

Definition at line 166 of file *Shape.cpp.*

const *Vector2f* & sf::Shape::GetPointPosition ( unsigned int *Index* ) const

Get the position of a point.

**Parameters:**

*Index* : Index of the point, in range [0, GetNbPoints() - 1]

**Returns:**

Position of the Index-th point

Definition at line 148 of file *Shape.cpp.*

const *Vector2f* & sf::Drawable::GetPosition ( ) const [inherited]
Get the position of the object.

**Returns:**
Current position

Definition at line 213 of file **Drawable.cpp**.

```cpp
float sf::Drawable::GetRotation() const [inherited]
```

Get the orientation of the object.

Rotation is always in the range [0, 360]

**Returns:**
Current rotation, in degrees

Definition at line 240 of file **Drawable.cpp**.

```cpp
const Vector2f & sf::Drawable::GetScale() const [inherited]
```

Get the current scale of the object.

**Returns:**
Current scale factor (always positive)

Definition at line 222 of file **Drawable.cpp**.

```cpp
Shape sf::Shape::Line( const Vector2f & P1,
const Vector2f & P2,
float Thickness,
const Color & Col,
float Outline = 0.f,
const Color & OutlineCol = sf::Color(0, 0, 0)
```
Create a shape made of a single line (use vectors).

**Parameters:**

- \( P1X, P1Y \): Position of the first point
- \( P2X, P2Y \): Position second point
- \( \text{Thickness} \): Line thickness
- \( \text{Col} \): **Color** used to draw the line
- \( \text{Outline} \): Outline width (0 by default)
- \( \text{OutlineCol} \): **Color** used to draw the outline (black by default)

Definition at line 212 of file **Shape.cpp**.

```cpp
Shape sf::Shape::Line ( float P1X, 
                      float P1Y, 
                      float P2X, 
                      float P2Y, 
                      float Thickness, 
                      const Color & Col, 
                      float Outline = 0.f, 
                      const Color & OutlineCol = sf::Color(0, 0, 0) 
) {
}
```

Create a shape made of a single line (use floats).

Create a shape made of a single line.

**Parameters:**

- \( P1X, P1Y \): Position of the first point
- \( P2X, P2Y \): Position second point
- \( \text{Thickness} \): Line thickness
- \( \text{Col} \): **Color** used to draw the line
- \( \text{Outline} \): Outline width (0 by default)
OutlineCol : Color used to draw the outline (black by default)

Definition at line 184 of file Shape.cpp.

```cpp
void sf::Drawable::Move (const Vector2f & Offset) [inherited]
```

Move the object of a given offset (take a 2D vector).

**Parameters:**

- `Offset` : Amount of units to move the object of

Definition at line 278 of file Drawable.cpp.

```cpp
void sf::Drawable::Move (float OffsetX, float OffsetY) [inherited]
```

Move the object of a given offset (take 2 values).

**Parameters:**

- `OffsetX` : X offset
- `OffsetY` : Y offset

Definition at line 268 of file Drawable.cpp.

```cpp
Shape sf::Shape::Rectangle (const Vector2f & P1,
const Vector2f & P2,
const Color & Col,
float Outline = 0.f,
const Color & OutlineCol = sf::Color(0, 0, 0))
```

Create a shape made of a single rectangle (use vectors).
Parameters:

- \( P1 \): Position of the first point
- \( P2 \): Position second point
- \( Col \): \texttt{Color} used to fill the rectangle
- \( Outline \): Outline width (0 by default)
- \( OutlineCol \): \texttt{Color} used to draw the outline (black by default)

Definition at line 241 of file \texttt{Shape.cpp}.

\begin{verbatim}
Shape sf::Shape::Rectangle ( float P1X, 
  float P1Y, 
  float P2X, 
  float P2Y, 
  const Color & Col, 
  float Outline = 0.f, 
  const Color & OutlineCol = sf::Color(0, 0, 0) 
)
\end{verbatim}

Create a shape made of a single rectangle (use floats).

Create a shape made of a single rectangle.

Parameters:

- \( P1X,P1Y \): Position of the first point
- \( P2X,P2Y \): Position second point
- \( Col \): \texttt{Color} used to fill the rectangle
- \( Outline \): Outline width (0 by default)
- \( OutlineCol \): \texttt{Color} used to draw the outline (black by default)

Definition at line 221 of file \texttt{Shape.cpp}.

\begin{verbatim}
void sf::Shape::Render ( RenderTarget & Target ) const [protected, virtual]
\end{verbatim}
/see Drawable::Render

Implements sf::Drawable.

Definition at line 285 of file Shape.cpp.

```cpp
void sf::Drawable::Rotate (float Angle) [inherited]
```

Rotate the object.

**Parameters:**

Angle : Angle of rotation, in degrees

Definition at line 306 of file Drawable.cpp.

```cpp
void sf::Drawable::Scale (const Vector2f & Factor) [inherited]
```

Scale the object (take a 2D vector).

**Parameters:**

Factor : Scaling factors (both values must be strictly positive)

Definition at line 297 of file Drawable.cpp.

```cpp
void sf::Drawable::Scale (float FactorX, float FactorY) [inherited]
```

Scale the object (take 2 values).

**Parameters:**

FactorX : Scaling factor on X (must be strictly positive)

FactorY : Scaling factor on Y (must be strictly positive)
**void sf::Drawable::SetBlendMode(Blend::Mode Mode)** [inherited]

Set the blending mode for the object.

The default blend mode is **Blend::Alpha**

**Parameters:**

- **Mode**: New blending mode

The default blend mode is **Blend::Alpha**

Definition at line 204 of file **Drawable.cpp**.

---

**void sf::Drawable::SetCenter(const Vector2f & Center)** [inherited]

Set the center of the object, in coordinates relative to the top-left of the object (take a 2D vector).

Set the center of the object, in coordinates relative to the top-left of the object (take a 2D vector).

The default center is (0, 0)

**Parameters:**

- **Center**: New center

The default center is (0, 0)

Definition at line 171 of file **Drawable.cpp**.
void sf::Drawable::SetCenter ( float CenterX, 
    float CenterY ) [inherited]

Set the center of the object, in coordinates relative to the top-left of the object (take 2 values).

Set the center of the object, in coordinates relative to the top-left of the object (take 2 values).

The default center is (0, 0)

Parameters:
    CenterX : X coordinate of the center
    CenterY : Y coordinate of the center

The default center is (0, 0)

Definition at line 157 of file Drawable.cpp.

void sf::Drawable::SetColor ( const Color & Col ) [inherited]

Set the color of the object.

The default color is white

Parameters:
    Col : New color

The default color is white

Definition at line 194 of file Drawable.cpp.
void sf::Shape::SetOutlineWidth ( float Width )

Change the width of the shape outline.

**Parameters:**

Width : New width

Definition at line 139 of file Shape.cpp.

void sf::Shape::SetPointColor ( unsigned int Index,
                                 const Color & Col )

Set the color of a point.

**Parameters:**

Index : Index of the point, in range [0, GetNbPoints() - 1]
Col : New color of the Index-th point

Definition at line 119 of file Shape.cpp.

void sf::Shape::SetPointOutlineColor ( unsigned int Index,
                                       const Color & OutlineCol )

Set the outline color of a point.

**Parameters:**

Index : Index of the point, in range [0, GetNbPoints() - 1]
OutlineCol : New outline color of the Index-th point

Definition at line 129 of file Shape.cpp.
### void sf::Shape::SetPointPosition

```cpp
template <typename Point>
void sf::Shape::SetPointPosition (unsigned int Index,
                                  Point& Position)
```

Set the position of a point.

**Parameters:**
- **Index**: Index of the point, in range [0, GetNbPoints() - 1]
- **Position**: New position of the Index-th point

Definition at line 100 of file Shape.cpp.

---

### void sf::Shape::SetPointPosition

```cpp
template <typename T>
void sf::Shape::SetPointPosition (unsigned int Index,
                                  const T& Position)
```

Set the position of a point.

**Parameters:**
- **Index**: Index of the point, in range [0, GetNbPoints() - 1]
- **Position**: New position of the Index-th point

Definition at line 110 of file Shape.cpp.

---

### void sf::Drawable::SetPosition

```cpp
void sf::Drawable::SetPosition (const Vector2f& Position) [inherited]
```

Set the position of the object (take a 2D vector).

**Parameters:**
- **Position**: New position
void sf::Drawable::setPosition(float X, float Y) [inherited]

Set the position of the object (take 2 values).

Parameters:
X : New X coordinate
Y : New Y coordinate

Definition at line 75 of file Drawable.cpp.

void sf::Drawable::setRotation(float Rotation) [inherited]

Set the orientation of the object.

Parameters:
Rotation : Angle of rotation, in degrees

Definition at line 65 of file Drawable.cpp.

void sf::Drawable::setScale(const Vector2f & Scale) [inherited]

Set the scale of the object (take a 2D vector).

Parameters:
Scale : New scale (both values must be strictly positive)

Definition at line 117 of file Drawable.cpp.
### void sf::Drawable::SetScale (float \textit{ScaleX}, float \textit{ScaleY}) [inherited]

Set the scale of the object (take 2 values).

**Parameters:**
- \textit{ScaleX} : New horizontal scale (must be strictly positive)
- \textit{ScaleY} : New vertical scale (must be strictly positive)

Definition at line 107 of file \texttt{Drawable.cpp}.

### void sf::Drawable::SetScaleX (float \textit{FactorX}) [inherited]

Set the X scale factor of the object.

**Parameters:**
- \textit{X} : New X scale factor

Definition at line 127 of file \texttt{Drawable.cpp}.

### void sf::Drawable::SetScaleY (float \textit{FactorY}) [inherited]

Set the Y scale factor of the object.

**Parameters:**
- \textit{Y} : New Y scale factor

Definition at line 141 of file \texttt{Drawable.cpp}.

### void sf::Drawable::SetX (float \textit{X}) [inherited]


Set the X position of the object.

**Parameters:**

$X$: New X coordinate

Definition at line 85 of file `Drawable.cpp`.

```cpp
void sf::Drawable::SetY(float Y) [inherited]
```

Set the Y position of the object.

**Parameters:**

$Y$: New Y coordinate

Definition at line 96 of file `Drawable.cpp`.

```cpp
sf::Vector2f sf::Drawable::TransformToGlobal(const sf::Vector2f & Point)
```

Transform a point from local coordinates into global coordinates (ie it applies the object's center, translation, rotation and scale to the point).

Transform a point from local coordinates into global coordinates (ie it applies the object's center, translation, rotation and scale to the point).

**Parameters:**

$Point$: Point to transform

**Returns:**

Transformed point

Definition at line 325 of file `Drawable.cpp`. 
sf::Vector2f sf::Drawable::TransformToLocal ( const sf::Vector2f & Point ) const

Transform a point from global coordinates into local coordinates (ie it applies the inverse of object's center, translation, rotation and scale to the point).

Transform a point from global coordinates into local coordinates (ie it applies the inverse of object's center, translation, rotation and scale to the point).

**Parameters:**

  *Point*: Point to transform

**Returns:**

  Transformed point

Definition at line 316 of file Drawable.cpp.

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

- Shape.hpp
- Shape.cpp
sf::SocketHelper
sf::SocketHelper Class Reference

This class defines helper functions to do all the non-portable socket stuff. More...

#include <SocketHelper.hpp>

List of all members.
### Public Types

<table>
<thead>
<tr>
<th>Type Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>typedef SOCKET</td>
<td>SocketType</td>
</tr>
<tr>
<td>typedef int</td>
<td>LengthType</td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static SocketType</td>
<td><strong>InvalidSocket</strong> ()</td>
<td>Return the value of the invalid socket.</td>
</tr>
<tr>
<td>static bool</td>
<td><strong>Close</strong> (SocketType Socket)</td>
<td>Close / destroy a socket.</td>
</tr>
<tr>
<td>static void</td>
<td><strong>SetBlocking</strong> (SocketType Socket, bool Block)</td>
<td>Set a socket as blocking or non-blocking.</td>
</tr>
<tr>
<td>static Socket::Status</td>
<td><strong>GetErrorStatus</strong> ()</td>
<td>Get the last socket error status.</td>
</tr>
</tbody>
</table>
Detailed Description

This class defines helper functions to do all the non-portable socket stuff.

This class is meant for internal use only

Definition at line 41 of file Win32/SocketHelper.hpp.
# Member Function Documentation

**bool sf::SocketHelper::Close ( SocketHelper::SocketType Socket ) [static]**

Close / destroy a socket.

**Parameters:**

Socket : Socket to close

**Returns:**

True on success

Definition at line 45 of file `SocketHelper.cpp`.

**Socket::Status sf::SocketHelper::GetErrorStatus ( ) [static]**

Get the last socket error status.

**Returns:**

Status corresponding to the last socket error

Definition at line 64 of file `SocketHelper.cpp`.

**SocketHelper::SocketType sf::SocketHelper::InvalidSocket ( ) [static]**

Return the value of the invalid socket.

**Returns:**

Unique value of the invalid socket

Definition at line 36 of file `SocketHelper.cpp`. 
void sf::SocketHelper::SetBlocking ( SocketHelper::SocketType *Socket, bool Block )

Set a socket as blocking or non-blocking.

**Parameters:**
- `Socket` : Socket to modify
- `Block` : New blocking state of the socket

Definition at line 54 of file `SocketHelper.cpp`.

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

- `Win32/SocketHelper.hpp`
- `SocketHelper.cpp`

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
sf::SocketTCP
sf::SocketTCP Class Reference

SocketTCP wraps a socket using TCP protocol to send data safely (but a bit slower). More...

#include <SocketTCP.hpp>

List of all members.
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SocketTCP ()</strong></td>
<td>Default constructor.</td>
</tr>
<tr>
<td>void <strong>SetBlocking</strong> (bool Blocking)</td>
<td>Change the blocking state of the socket.</td>
</tr>
<tr>
<td><strong>Socket::Status Connect</strong> (unsigned short Port, const IPAddress &amp;HostAddress, float Timeout=0.f)</td>
<td>Connect to another computer on a specified port.</td>
</tr>
<tr>
<td>bool <strong>Listen</strong> (unsigned short Port)</td>
<td>Listen to a specified port for incoming data or connections.</td>
</tr>
<tr>
<td><strong>Socket::Status Accept</strong> (SocketTCP &amp;Connected, IPAddress *Address=NULL)</td>
<td>Wait for a connection (must be listening to a port).</td>
</tr>
<tr>
<td><strong>Socket::Status Send</strong> (const char *Data, std::size_t Size)</td>
<td>Send an array of bytes to the host (must be connected first).</td>
</tr>
<tr>
<td><strong>Socket::Status Receive</strong> (char *Data, std::size_t MaxSize, std::size_t &amp;SizeReceived)</td>
<td>Receive an array of bytes from the host (must be connected first).</td>
</tr>
<tr>
<td><strong>Socket::Status Send</strong> (Packet &amp;PacketToSend)</td>
<td>Send a packet of data to the host (must be connected first).</td>
</tr>
<tr>
<td><strong>Socket::Status Receive</strong> (Packet &amp;PacketToReceive)</td>
<td>Receive a packet from the host (must be connected first).</td>
</tr>
<tr>
<td>bool <strong>Close</strong> ()</td>
<td>Close the socket.</td>
</tr>
<tr>
<td>bool <strong>IsValid</strong> () const</td>
<td>Check if the socket is in a valid state; this function can be called any time to check if the socket is OK.</td>
</tr>
<tr>
<td>bool <strong>operator==</strong> (const SocketTCP &amp;Other) const</td>
<td>Comparison operator ==.</td>
</tr>
<tr>
<td>bool <strong>operator!=</strong> (const SocketTCP &amp;Other) const</td>
<td>Comparison operator !=.</td>
</tr>
<tr>
<td>bool <strong>operator&lt;</strong> (const SocketTCP &amp;Other) const</td>
<td>Comparison operator &lt;.</td>
</tr>
</tbody>
</table>
Friends

class Selector<SocketTCP>
Detailed Description

**SocketTCP** wraps a socket using TCP protocol to send data safely (but a bit slower).

Definition at line 45 of file *SocketTCP.hpp*. 
sf::SocketTCP::SocketTCP()

Default constructor.

Definition at line 47 of file SocketTCP.cpp.
Member Function Documentation

Socket::Status sf::SocketTCP::Accept (SocketTCP & Connected, IPAddress * Address = NULL)

Wait for a connection (must be listening to a port).
This function will block if the socket is blocking

Parameters:
- **Connected**: Socket containing the connection with the connected client
- **Address**: Pointer to an address to fill with client infos (NULL by default)

Returns:
- Status code
This function will block if the socket is blocking
Definition at line 206 of file SocketTCP.cpp.

bool sf::SocketTCP::Close()

Close the socket.

Returns:
- True if operation has been successful
Definition at line 406 of file SocketTCP.cpp.

Socket::Status sf::SocketTCP::Connect (unsigned short Port,
Connect to another computer on a specified port.

**Parameters:**
- **Port**: Port to use for transfers (warning: ports < 1024 are reserved)
- **HostAddress**: IP Address of the host to connect to
- **Timeout**: Maximum time to wait, in seconds (0 by default: no timeout) (this parameter is ignored for non-blocking sockets)

**Returns:**
True if operation has been successful

Definition at line 70 of file SocketTCP.cpp.

```cpp
bool sf::SocketTCP::IsValid () const
```

Check if the socket is in a valid state; this function can be called any time to check if the socket is OK.

Check if the socket is in a valid state; this function can be called any time to check if the socket is OK.

**Returns:**
True if the socket is valid

Definition at line 429 of file SocketTCP.cpp.

```cpp
bool sf::SocketTCP::Listen (unsigned short Port )
```
Listen to a specified port for incoming data or connections.

**Parameters:**

- **Port**: Port to listen to

**Returns:**

True if operation has been successful

Definition at line 169 of file `SocketTCP.cpp`.

```cpp
bool sf::SocketTCP::operator!= (const SocketTCP & Other) const
```

Comparison operator !=.  

**Parameters:**

- **Other**: Socket to compare

**Returns:**

True if *this != Other

Definition at line 447 of file `SocketTCP.cpp`.

```cpp
bool sf::SocketTCP::operator< (const SocketTCP & Other) const
```

Comparison operator <.

Provided for compatibility with standard containers, as comparing two sockets doesn't make much sense...

**Parameters:**

- **Other**: Socket to compare

**Returns:**
True if *this < Other

Provided for compatibility with standard containers, as comparing two sockets doesn't make much sense...

Definition at line 458 of file SocketTCP.cpp.

```cpp
bool sf::SocketTCP::operator==( const SocketTCP & Other ) const
```

Comparison operator ==.

**Parameters:**
- *Other* : Socket to compare

**Returns:**
True if *this == Other

Definition at line 438 of file SocketTCP.cpp.

```cpp
Socket::Status sf::SocketTCP::Receive ( Packet & PacketToReceive )
```

Receive a packet from the host (must be connected first).

This function will block if the socket is blocking

**Parameters:**
- *PacketToReceive* : Packet to fill with received data

**Returns:**
Status code

This function will block if the socket is blocking
Socket::Status sf::SocketTCP::Receive (char * Data, std::size_t MaxSize, std::size_t & SizeReceived)

Receive an array of bytes from the host (must be connected first).

This function will block if the socket is blocking

Parameters:
- **Data**: Pointer to a byte array to fill (make sure it is big enough)
- **MaxSize**: Maximum number of bytes to read
- **SizeReceived**: Number of bytes received

Returns:
- Status code

This function will block if the socket is blocking

Definition at line 272 of file SocketTCP.cpp.

Socket::Status sf::SocketTCP::Send (Packet & PacketToSend)

Send a packet of data to the host (must be connected first).

Parameters:
- **PacketToSend**: Packet to send

Returns:
- Status code

Definition at line 340 of file SocketTCP.cpp.
Socket::Status sf::SocketTCP::Send(const char * Data, std::size_t Size)

Send an array of bytes to the host (must be connected first).

**Parameters:**

- *Data*: Pointer to the bytes to send
- *Size*: Number of bytes to send

**Returns:**

Status code

Definition at line 314 of file SocketTCP.cpp.

void sf::SocketTCP::SetBlocking(bool Blocking)

Change the blocking state of the socket.

The default behaviour of a socket is blocking.

**Parameters:**

- *Blocking*: Pass true to set the socket as blocking, or false for non-blocking

Definition at line 235 of file SocketTCP.cpp.

---

The documentation for this class was generated from the following files:
• SocketTCP.hpp
• SocketTCP.cpp

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
sf::SocketUDP
sf::SocketUDP Class Reference

_socketUDP_ wraps a socket using UDP protocol to send data fastly (but with less safety). More...

#include <SocketUDP.hpp>

List of all members.
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>SocketUDP()</code></td>
<td>Default constructor.</td>
</tr>
<tr>
<td><code>SetBlocking(bool Blocking)</code></td>
<td>Change the blocking state of the socket.</td>
</tr>
<tr>
<td><code>Bind(unsigned short Port)</code></td>
<td>Bind the socket to a specific port.</td>
</tr>
<tr>
<td><code>Unbind()</code></td>
<td>Unbind the socket from its previous port, if any.</td>
</tr>
<tr>
<td><code>Send(char *Data, std::size_t Size, const IPAddress &amp;Address, unsigned short Port)</code></td>
<td>Send an array of bytes.</td>
</tr>
<tr>
<td><code>Receive(char *Data, std::size_t MaxSize, std::size_t &amp;SizeReceived, IPAddress &amp;Address, unsigned short &amp;Port)</code></td>
<td>Receive an array of bytes.</td>
</tr>
<tr>
<td><code>Send(Packet &amp;PacketToSend, const IPAddress &amp;Address, unsigned short Port)</code></td>
<td>Send a packet of data.</td>
</tr>
<tr>
<td><code>Receive(Packet &amp;PacketToReceive, IPAddress &amp;Address, unsigned short &amp;Port)</code></td>
<td>Receive a packet.</td>
</tr>
<tr>
<td><code>Close()</code></td>
<td>Close the socket.</td>
</tr>
<tr>
<td><code>IsValid()</code> const</td>
<td>Check if the socket is in a valid state; this function can be called any time to check if the socket is OK.</td>
</tr>
<tr>
<td><code>GetPort()</code> const</td>
<td>Get the port the socket is currently bound to.</td>
</tr>
<tr>
<td><code>operator==(const SocketUDP &amp;Other)</code> const</td>
<td>Comparison operator ==.</td>
</tr>
<tr>
<td><code>operator!=(const SocketUDP &amp;Other)</code> const</td>
<td>Comparison operator !=.</td>
</tr>
<tr>
<td><code>operator&lt;(const SocketUDP &amp;Other)</code> const</td>
<td>Comparison operator &lt;.</td>
</tr>
</tbody>
</table>
Friends

```java
class Selector<SocketUDP>
```
Detailed Description

**SocketUDP** wraps a socket using UDP protocol to send data fastly (but with less safety).

Definition at line 45 of file *SocketUDP.hpp*. 
## Constructor & Destructor Documentation

<table>
<thead>
<tr>
<th>sf::SocketUDP::SocketUDP ( )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default constructor.</td>
</tr>
<tr>
<td>Definition at line 41 of file SocketUDP.cpp.</td>
</tr>
</tbody>
</table>
**Member Function Documentation**

### bool sf::SocketUDP::Bind ( unsigned short Port )

Bind the socket to a specific port.

**Parameters:**
- Port : Port to bind the socket to

**Returns:**
- True if operation has been successful

Definition at line 64 of file `SocketUDP.cpp`.

### bool sf::SocketUDP::Close ( )

Close the socket.

**Returns:**
- True if operation has been successful

Definition at line 316 of file `SocketUDP.cpp`.

### unsigned short sf::SocketUDP::GetPort ( ) const

Get the port the socket is currently bound to.

**Returns:**
- Current port (0 means the socket is not bound)

Definition at line 349 of file `SocketUDP.cpp`. 
bool sf::SocketUDP::IsValid ( ) const

Check if the socket is in a valid state; this function can be called any time to check if the socket is OK.

Check if the socket is in a valid state; this function can be called any time to check if the socket is OK.

**Returns:**
True if the socket is valid

Definition at line 340 of file SocketUDP.cpp.

bool sf::SocketUDP::operator!= ( const SocketUDP & Other ) const

Comparison operator !.=.

**Parameters:**
Other : Socket to compare

**Returns:**
True if *this != Other

Definition at line 367 of file SocketUDP.cpp.

bool sf::SocketUDP::operator< ( const SocketUDP & Other ) const

Comparison operator <.

Provided for compatibility with standard containers, as comparing two sockets doesn't make much sense...

**Parameters:**
Other : Socket to compare

Returns:
    True if *this < Other

Provided for compatibility with standard containers, as comparing two sockets doesn't make much sense...

Definition at line 378 of file SocketUDP.cpp.

| bool sf::SocketUDP::operator==( const SocketUDP & Other ) const |

Comparison operator ==.

Parameters:
    Other : Socket to compare

Returns:
    True if *this == Other

Definition at line 358 of file SocketUDP.cpp.

| Socket::Status sf::SocketUDP::Receive ( Packet & PacketToReceive, IPAddress & Address, unsigned short & Port ) |

Receive a packet.

This function will block if the socket is blocking

Parameters:
    PacketToReceive : Packet to fill with received data
Receive an array of bytes.

This function will block if the socket is blocking

**Parameters:**

- `Data` : Pointer to a byte array to fill (make sure it is big enough)
- `MaxSize` : Maximum number of bytes to read
- `SizeReceived` : Number of bytes received
- `Address` : Address of the computer which sent the data
- `Port` : Port on which the remote computer sent the data

**Returns:**

Status code

This function will block if the socket is blocking

Definition at line 245 of file `SocketUDP.cpp`.

Receive an array of bytes.

This function will block if the socket is blocking

**Parameters:**

- `Data` : Pointer to a byte array to fill (make sure it is big enough)
- `MaxSize` : Maximum number of bytes to read
- `SizeReceived` : Number of bytes received
- `Address` : Address of the computer which sent the data
- `Port` : Port on which the remote computer sent the data

**Returns:**

Status code

This function will block if the socket is blocking

Definition at line 162 of file `SocketUDP.cpp`.

**Address** : Address of the computer which sent the packet

**Port** : Port on which the remote computer sent the data

**Returns:**

Status code

This function will block if the socket is blocking

Definition at line 245 of file `SocketUDP.cpp`.
Send a packet of data.

**Parameters:**
- `PacketToSend` : Packet to send
- `Address` : Address of the computer to send the packet to
- `Port` : Port to send the data to

**Returns:**
Status code

Definition at line 219 of file `SocketUDP.cpp`.

Send an array of bytes.

**Parameters:**
- `Data` : Pointer to the bytes to send
- `Size` : Number of bytes to send
- `Address` : Address of the computer to send the packet to
- `Port` : Port to send the data to

**Returns:**
Status code
### `sf::SocketUDP::SetBlocking` (bool `Blocking`)

Change the blocking state of the socket.

The default behaviour of a socket is blocking

**Parameters:**
- `Blocking`: Pass true to set the socket as blocking, or false for non-blocking

### `sf::SocketUDP::Unbind`()

Unbind the socket from its previous port, if any.

Unbind the socket to its previous port.

**Returns:**
- True if operation has been successful

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

- `SocketUDP.hpp`
- `SocketUDP.cpp`
sf::Sound
sf::Sound Class Reference

**Sound** defines the properties of a sound such as position, volume, pitch, etc. [More...](#)

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

Inheritance diagram for sf::Sound:

![Inheritance diagram for sf::Sound](image)

[List of all members.](#)
### Public Types

| enum     | Status {       |
|          | Stopped,      |
|          | Paused,       |
|          | Playing       |
|          }     |

*Enumeration of the sound states. More...*
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sound ()</strong></td>
<td><em>Default constructor.</em></td>
</tr>
<tr>
<td><strong>Sound (const SoundBuffer &amp;Buffer, bool Loop=false, float Pitch=1.f, float Volume=100.f, const Vector3f &amp;Position=Vector3f(0, 0, 0))</strong></td>
<td><em>Construct the sound from its parameters.</em></td>
</tr>
<tr>
<td><strong>Sound (const Sound &amp;Copy)</strong></td>
<td><em>Copy constructor.</em></td>
</tr>
<tr>
<td><strong>~Sound ()</strong></td>
<td><em>Destructor.</em></td>
</tr>
<tr>
<td><strong>void Play ()</strong></td>
<td><em>Play the sound.</em></td>
</tr>
<tr>
<td><strong>void Pause ()</strong></td>
<td><em>Pause the sound.</em></td>
</tr>
<tr>
<td><strong>void Stop ()</strong></td>
<td><em>Stop the sound.</em></td>
</tr>
<tr>
<td><strong>void SetBuffer (const SoundBuffer &amp;Buffer)</strong></td>
<td><em>Set the source buffer.</em></td>
</tr>
<tr>
<td><strong>void SetLoop (bool Loop)</strong></td>
<td><em>Set the sound loop state.</em></td>
</tr>
<tr>
<td><strong>void SetPitch (float Pitch)</strong></td>
<td><em>Set the sound pitch.</em></td>
</tr>
<tr>
<td><strong>void SetVolume (float Volume)</strong></td>
<td><em>Set the sound volume.</em></td>
</tr>
<tr>
<td><strong>void SetPosition (float X, float Y, float Z)</strong></td>
<td><em>Set the sound position (take 3 values).</em></td>
</tr>
<tr>
<td><strong>void SetPosition (const Vector3f &amp;Position)</strong></td>
<td><em>Set the sound position (take a 3D vector).</em></td>
</tr>
<tr>
<td><strong>void SetRelativeToListener (bool Relative)</strong></td>
<td><em>Make the sound's position relative to the listener's position, or absolute.</em></td>
</tr>
<tr>
<td><strong>void SetMinDistance (float MinDistance)</strong></td>
<td><em>Set the minimum distance - closer than this distance, the listener will hear the sound at its maximum volume.</em></td>
</tr>
<tr>
<td><strong>void SetAttenuation (float Attenuation)</strong></td>
<td><em>Set the attenuation factor - the higher the attenuation, the more the sound will be attenuated with distance from listener.</em></td>
</tr>
<tr>
<td><strong>void SetPlayingOffset (float TimeOffset)</strong></td>
<td><em>Set the current playing position of the sound.</em></td>
</tr>
<tr>
<td>**const SoundBuffer **</td>
<td><strong>GetBuffer () const</strong></td>
</tr>
<tr>
<td><strong>bool GetLoop () const</strong></td>
<td><em>Tell whether or not the sound is looping.</em></td>
</tr>
<tr>
<td>Type</td>
<td>Function</td>
</tr>
<tr>
<td>-----------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>float</td>
<td><code>GetPitch()</code> const</td>
</tr>
<tr>
<td>float</td>
<td><code>GetVolume()</code> const</td>
</tr>
<tr>
<td>Vector3f</td>
<td><code>GetPosition()</code> const</td>
</tr>
<tr>
<td>bool</td>
<td><code>IsRelativeToListener()</code> const</td>
</tr>
<tr>
<td>float</td>
<td><code>GetMinDistance()</code> const</td>
</tr>
<tr>
<td>float</td>
<td><code>GetAttenuation()</code> const</td>
</tr>
<tr>
<td>Status</td>
<td><code>GetStatus()</code> const</td>
</tr>
<tr>
<td>float</td>
<td><code>GetPlayingOffset()</code> const</td>
</tr>
<tr>
<td>Sound &amp;</td>
<td><code>operator=</code> (const Sound &amp; Other)</td>
</tr>
<tr>
<td>void</td>
<td><code>ResetBuffer()</code></td>
</tr>
</tbody>
</table>
Friends

class SoundStream
Detailed Description

**Sound** defines the properties of a sound such as position, volume, pitch, etc.

Definition at line 45 of file *Sound.hpp*.
### Member Enumeration Documentation

<table>
<thead>
<tr>
<th>enum sf::Sound::Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enumeration of the sound states.</td>
</tr>
</tbody>
</table>

**Enumerator:**

- *Stopped*  **Sound** is not playing.
- *Paused*  **Sound** is paused.
- *Playing*  **Sound** is playing.

Definition at line 52 of file `Sound.hpp`. 
Constructor & Destructor Documentation

sf::Sound::Sound ( )

Default constructor.

Definition at line 38 of file Sound.cpp.

sf::Sound::Sound ( const SoundBuffer & Buffer,
  bool Loop = false,
  float Pitch = 1.f,
  float Volume = 100.f,
  const Vector3f & Position = Vector3f(0, 0, 0) )

[explicit]

Construct the sound from its parameters.

Parameters:

  Buffer : Sound buffer to play (NULL by default)
  Loop   : Loop flag (false by default)
  Pitch  : Value of the pitch (1 by default)
  Volume : Volume (100 by default)
  Position : Position (0, 0, 0 by default)

Definition at line 48 of file Sound.cpp.

sf::Sound::Sound ( const Sound & Copy )

Copy constructor.

Parameters:

  Copy : Instance to copy
sf::Sound::~Sound()

Destructor.

Definition at line 85 of file Sound.cpp.
Member Function Documentation

**float sf::Sound::GetAttenuation() const**

Get the attenuation factor.

**Returns:**
Attenuation factor of the sound

Definition at line 319 of file Sound.cpp.

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

Get the source buffer.

**Returns:**
Sound buffer bound to the sound (can be NULL)

Definition at line 237 of file Sound.cpp.

**bool sf::Sound::GetLoop() const**

Tell whether or not the sound is looping.

**Returns:**
True if the sound is looping, false otherwise

Reimplemented in sf::SoundStream.

Definition at line 246 of file Sound.cpp.
**float sf::Sound::GetMinDistance ( ) const**

Get the minimum distance.

**Returns:**
Minimum distance for the sound

Definition at line 307 of file Sound.cpp.

---

**float sf::Sound::GetPitch ( ) const**

Get the pitch.

**Returns:**
Pitch value

Definition at line 258 of file Sound.cpp.

---

**float sf::Sound::GetPlayingOffset ( ) const**

Get the current playing position of the sound.

**Returns:**
Current playing position, expressed in seconds

Reimplemented in sf::SoundStream.

Definition at line 331 of file Sound.cpp.

---

**Vector3f sf::Sound::GetPosition ( ) const**

Get the sound position.
### Returns:
Position of the sound in the world

Definition at line 282 of file `Sound.cpp`.

### `Sound::Status sf::Sound::GetStatus ( ) const`

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

**Returns:**
Current status of the sound

Reimplemented in `sf::SoundStream`.

Definition at line 343 of file `Sound.cpp`.

### `float sf::Sound::GetVolume ( ) const`

Get the volume.

**Returns:**
Volume value (in range [1, 100])

Definition at line 270 of file `Sound.cpp`.

### `bool sf::Sound::IsRelativeToListener ( ) const`

Tell if the sound's position is relative to the listener's position, or if it's absolute.

Tell if the sound's position is relative to the listener's position, or if it's absolute.
**Returns:**
True if the position is relative, false if it's absolute

Definition at line 295 of file Sound.cpp.

**Sound & sf::Sound::operator= ( const Sound & Other )**

Assignment operator.

**Parameters:**
*Other* : Instance to assign

**Returns:**
Reference to the sound

Definition at line 363 of file Sound.cpp.

**void sf::Sound::Pause ( )**

Pause the sound.

Definition at line 112 of file Sound.cpp.

**void sf::Sound::Play ( )**

Play the sound.

Reimplemented in sf::SoundStream.

Definition at line 103 of file Sound.cpp.

**void sf::Sound::ResetBuffer ( )**
Reset the internal buffer.

This function is for internal use only, you don't have to use it.

Definition at line 394 of file Sound.cpp.

```cpp
void sf::Sound::SetAttenuation ( float Attenuation )
```

Set the attenuation factor - the higher the attenuation, the more the sound will be attenuated with distance from listener.

Set the attenuation factor - the higher the attenuation, the more the sound will be attenuated with distance from listener.

The default attenuation factor 1.0

**Parameters:**

- `Attenuation` : New attenuation factor for the sound

The default attenuation factor 1.0

Definition at line 219 of file Sound.cpp.

```cpp
void sf::Sound::SetBuffer ( const SoundBuffer & Buffer )
```

Set the source buffer.

**Parameters:**

- `Buffer` : New sound buffer to bind to the sound

Definition at line 130 of file Sound.cpp.
void sf::Sound::SetLoop ( bool Loop )

Set the sound loop state.

This parameter is disabled by default

Parameters:

  Loop : True to play in loop, false to play once

Reimplemented in sf::SoundStream.

Definition at line 149 of file Sound.cpp.

void sf::Sound::SetMinDistance ( float MinDistance )

Set the minimum distance - closer than this distance, the listener will hear the sound at its maximum volume.

Set the minimum distance - closer than this distance, the listener will hear the sound at its maximum volume.

The default minimum distance is 1.0

Parameters:

  MinDistance : New minimum distance for the sound

The default minimum distance is 1.0

Definition at line 208 of file Sound.cpp.

void sf::Sound::SetPitch ( float Pitch )

Set the sound pitch.
The default pitch is 1

**Parameters:**
*Pitch*: New pitch

Definition at line 158 of file *Sound.cpp*.

```cpp
void sf::Sound::SetPlayingOffset ( float TimeOffset )
```

Set the current playing position of the sound.

**Parameters:**
*TimeOffset*: New playing position, expressed in seconds

Definition at line 228 of file *Sound.cpp*.

```cpp
void sf::Sound::SetPosition ( const Vector3f & Position )
```

Set the sound position (take a 3D vector).

The default position is (0, 0, 0)

**Parameters:**
*Position*: Position of the sound in the world

The default position is (0, 0, 0)

Definition at line 186 of file *Sound.cpp*.

```cpp
void sf::Sound::SetPosition ( float X, float Y, float Z )
```
Set the sound position (take 3 values).

The default position is (0, 0, 0)

**Parameters:**

\(X, Y, Z\) : Position of the sound in the world

The default position is (0, 0, 0)

Definition at line 176 of file Sound.cpp.

```cpp
void sf::Sound::SetRelativeToListener ( bool Relative )
```

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

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

The default value is false (absolute)

**Parameters:**

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

The default value is false (absolute)

Definition at line 197 of file Sound.cpp.

```cpp
void sf::Sound::SetVolume ( float Volume )
```

Set the sound volume.
The default volume is 100

**Parameters:**

*Volume* : Volume (in range [0, 100])

Definition at line 167 of file `Sound.cpp`.

```cpp
void sf::Sound::Stop()
```

Stop the sound.

Reimplemented in `sf::SoundStream`.

Definition at line 121 of file `Sound.cpp`.

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

- `Sound.hpp`
- `Sound.cpp`

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
sf::SoundBuffer
sf::SoundBuffer Class Reference

**SoundBuffer** is the low-level for loading and manipulating sound buffers. [More...](#) 

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

Inheritance diagram for sf::SoundBuffer:

```
sf::AudioResource  sf::Resource<SoundBuffer>  sf::SoundBuffer
```

[More...](#) 

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SoundBuffer ()</strong></td>
<td>Default constructor.</td>
</tr>
<tr>
<td><strong>SoundBuffer (const SoundBuffer &amp;Copy)</strong></td>
<td>Copy constructor.</td>
</tr>
<tr>
<td><strong>~SoundBuffer ()</strong></td>
<td>Destructor.</td>
</tr>
<tr>
<td>bool <strong>LoadFromFile</strong> (const std::string &amp;Filename)</td>
<td>Load the sound buffer from a file.</td>
</tr>
<tr>
<td>bool <strong>LoadFromMemory</strong> (const char *Data, std::size_t SizeInBytes)</td>
<td>Load the sound buffer from a file in memory.</td>
</tr>
<tr>
<td>bool <strong>LoadFromSamples</strong> (const Int16 *Samples, std::size_t SamplesCount, unsigned int ChannelsCount, unsigned int SampleRate)</td>
<td>Load the sound buffer from an array of samples - assumed format for samples is 16 bits signed integer.</td>
</tr>
<tr>
<td>bool <strong>SaveToFile</strong> (const std::string &amp;Filename) const</td>
<td>Save the sound buffer to a file.</td>
</tr>
<tr>
<td>const Int16 * <strong>GetSamples</strong> () const</td>
<td>Return the sound samples.</td>
</tr>
<tr>
<td>std::size_t <strong>GetSamplesCount</strong> () const</td>
<td>Return the samples count.</td>
</tr>
<tr>
<td>unsigned int <strong>GetSampleRate</strong> () const</td>
<td>Get the sample rate.</td>
</tr>
<tr>
<td>unsigned int <strong>GetChannelsCount</strong> () const</td>
<td>Return the number of channels (1 = mono, 2 = stereo, .</td>
</tr>
<tr>
<td>float <strong>GetDuration</strong> () const</td>
<td>Get the sound duration.</td>
</tr>
<tr>
<td><strong>SoundBuffer &amp; operator= (const SoundBuffer &amp;Other)</strong></td>
<td>Assignment operator.</td>
</tr>
</tbody>
</table>
Friends

class Sound
Detailed Description

**SoundBuffer** is the low-level for loading and manipulating sound buffers.

Definition at line 46 of file *SoundBuffer.hpp*. 
Constructor & Destructor Documentation

**sf::SoundBuffer::SoundBuffer ( )**

Default constructor.

Definition at line 42 of file SoundBuffer.cpp.

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

Copy constructor.

**Parameters:**

Copy : Instance to copy

Definition at line 54 of file SoundBuffer.cpp.

**sf::SoundBuffer::~SoundBuffer ( )**

Destructor.

Definition at line 73 of file SoundBuffer.cpp.
Member Function Documentation

### unsigned int sf::SoundBuffer::GetChannelsCount ( ) const

Return the number of channels (1 = mono, 2 = stereo, ..)

**Returns:**
Number of channels

..)

Definition at line 253 of file `SoundBuffer.cpp`.

### float sf::SoundBuffer::GetDuration ( ) const

Get the sound duration.

**Returns:**
Sound duration, in seconds

Definition at line 265 of file `SoundBuffer.cpp`.

### unsigned int sf::SoundBuffer::GetSampleRate ( ) const

Get the sample rate.

**Returns:**
Sound frequency (number of samples per second)

Definition at line 241 of file `SoundBuffer.cpp`. 
const Int16 * sf::SoundBuffer::GetSamples() const

Return the sound samples.

**Returns:**
Pointer to the array of sound samples, in 16 bits signed integer format.

Definition at line 223 of file SoundBuffer.cpp.

std::size_t sf::SoundBuffer::GetSamplesCount() const

Return the samples count.

**Returns:**
Number of samples.

Definition at line 232 of file SoundBuffer.cpp.

bool sf::SoundBuffer::LoadFromFile( const std::string & Filename )

Load the sound buffer from a file.

**Parameters:**
  *Filename*: Path of the sound file to load

**Returns:**
True if loading has been successful.

Definition at line 88 of file SoundBuffer.cpp.


```cpp
bool sf::SoundBuffer::LoadFromMemory ( const char * Data,
                                        std::size_t SizeInBytes )
```

Load the sound buffer from a file in memory.

**Parameters:**
- *Data*: Pointer to the file data in memory
- *SizeInBytes*: Size of the data to load, in bytes

**Returns:**
- True if loading has been successful

Definition at line 129 of file SoundBuffer.cpp.

```cpp
bool sf::SoundBuffer::LoadFromSamples ( const Int16 * Samples,
                                        std::size_t SamplesCount,
                                        unsigned int ChannelsCount,
                                        unsigned int SampleRate )
```

Load the sound buffer from an array of samples - assumed format for samples is 16 bits signed integer.

Load the sound buffer from an array of samples - assumed format for samples is 16 bits signed integer.

**Parameters:**
- *Samples*: Pointer to the samples in memory
- *SamplesCount*: Number of samples pointed by Samples
- *ChannelsCount*: Number of channels (1 = mono, 2 = stereo, ...)
- *SampleRate*: Frequency (number of samples to play per second)

**Returns:**
True if loading has been successful

Definition at line 171 of file SoundBuffer.cpp.

\texttt{SoundBuffer} & sf::SoundBuffer::operator= ( const SoundBuffer & \textit{Other} )

Assignment operator.

\textbf{Parameters:}
\begin{itemize}
  \item \textit{Other} : Instance to assign
\end{itemize}

\textbf{Returns:}
Reference to the sound buffer

Definition at line 274 of file SoundBuffer.cpp.

\texttt{bool} sf::SoundBuffer::SaveToFile ( const std::string & \textit{Filename} ) const

Save the sound buffer to a file.

\textbf{Parameters:}
\begin{itemize}
  \item \textit{Filename} : Path of the sound file to write
\end{itemize}

\textbf{Returns:}
True if saving has been successful

Definition at line 199 of file SoundBuffer.cpp.

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

- SoundBuffer.hpp
sf::SoundBufferRecorder
sf::SoundBufferRecorder Class Reference

Specialized SoundRecorder which saves the captured audio data into a sound buffer. More...

#include <SoundBufferRecorder.hpp>

Inheritance diagram for sf::SoundBufferRecorder:

List of all members.
**Public Member Functions**

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>const SoundBuffer &amp;</td>
<td>GetBuffer () const</td>
<td>Get the sound buffer containing the captured audio data.</td>
</tr>
<tr>
<td>void</td>
<td>Start (unsigned int SampleRate=44100)</td>
<td>Start the capture.</td>
</tr>
<tr>
<td>void</td>
<td>Stop ()</td>
<td>Stop the capture.</td>
</tr>
<tr>
<td>unsigned int</td>
<td>GetSampleRate () const</td>
<td>Get the sample rate.</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static bool</th>
<th>CanCapture ()</th>
</tr>
</thead>
</table>
|Tell if the system supports sound capture.
Detailed Description

Specialized `SoundRecorder` which saves the captured audio data into a sound buffer.

Definition at line 42 of file `SoundBufferRecorder.hpp`. 
### Member Function Documentation

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bool sf::SoundRecorder::CanCapture( )</code></td>
<td>Tell if the system supports sound capture. If not, this class won't be usable</td>
</tr>
<tr>
<td></td>
<td><strong>Returns:</strong> True if audio capture is supported. If not, this class won't be usable</td>
</tr>
<tr>
<td></td>
<td>Definition at line 136 of file <code>SoundRecorder.cpp</code>.</td>
</tr>
<tr>
<td><code>const SoundBuffer &amp; sf::SoundBufferRecorder::GetBuffer( )</code></td>
<td>Get the sound buffer containing the captured audio data.</td>
</tr>
<tr>
<td></td>
<td><strong>Returns:</strong> Constant reference to the sound buffer</td>
</tr>
<tr>
<td></td>
<td>Definition at line 70 of file <code>SoundBufferRecorder.cpp</code>.</td>
</tr>
<tr>
<td><code>unsigned int sf::SoundRecorder::GetSampleRate( )</code></td>
<td>Get the sample rate.</td>
</tr>
<tr>
<td></td>
<td><strong>Returns:</strong> Frequency, in samples per second</td>
</tr>
<tr>
<td></td>
<td>Definition at line 126 of file <code>SoundRecorder.cpp</code>.</td>
</tr>
</tbody>
</table>
void sf::SoundRecorder::Start ( unsigned int SampleRate = 44100 ) [inherited]

Start the capture.

Warning : only one capture can happen at the same time

Parameters:

`SampleRate` : Sound frequency (the more samples, the higher the quality) (44100 by default = CD quality)

Warning : only one capture can happen at the same time

Definition at line 69 of file SoundRecorder.cpp.

void sf::SoundRecorder::Stop ( ) [inherited]

Stop the capture.

Definition at line 115 of file SoundRecorder.cpp.

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

- SoundBufferRecorder.hpp
- SoundBufferRecorder.cpp
sf::SoundRecorder
sf::SoundRecorder Class Reference

**SoundRecorder** is an interface for capturing sound data, it is meant to be used as a base class. [More...](#)

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

Inheritance diagram for sf::SoundRecorder:

![Inheritance Diagram](image)

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

<table>
<thead>
<tr>
<th>virtual</th>
<th><code>~SoundRecorder()</code></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Virtual destructor.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><code>Start(unsigned int SampleRate=44100)</code></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Start the capture.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><code>Stop()</code></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Stop the capture.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>unsigned int</th>
<th><code>GetSampleRate()</code> const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Get the sample rate.</td>
</tr>
</tbody>
</table>
**Static Public Member Functions**

<table>
<thead>
<tr>
<th>static bool CanCapture()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Tell if the system supports sound capture.</em></td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>SoundRecorder()</code></td>
<td>Default constructor.</td>
</tr>
</tbody>
</table>
Private Types

typedef void(* FuncType )(void *)
**Private Member Functions**

<table>
<thead>
<tr>
<th>void</th>
<th>Launch ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>Create and run the thread.</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>Wait ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>Wait until the thread finishes.</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>Terminate ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>Terminate the thread Terminating a thread with this function is not safe, you should rather try to make the thread function terminate by itself.</em></td>
</tr>
</tbody>
</table>
**Detailed Description**

*SoundRecorder* is an interface for capturing sound data, it is meant to be used as a base class.

Definition at line 41 of file *SoundRecorder.hpp*. 
## Constructor & Destructor Documentation

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sf::SoundRecorder::~SoundRecorder ( ) [virtual]</code></td>
<td>Virtual destructor. Definition at line 59 of file <code>SoundRecorder.cpp</code>.</td>
</tr>
<tr>
<td><code>sf::SoundRecorder::SoundRecorder ( ) [protected]</code></td>
<td>Default constructor. Definition at line 48 of file <code>SoundRecorder.cpp</code>.</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
bool sf::SoundRecorder::CanCapture ( ) [static]
Tell if the system supports sound capture.
If not, this class won't be usable

**Returns:**
True if audio capture is supported
If not, this class won't be usable
Definition at line 136 of file SoundRecorder.cpp.
```

```cpp
unsigned int sf::SoundRecorder::GetSampleRate ( ) const
Get the sample rate.

**Returns:**
Frequency, in samples per second
Definition at line 126 of file SoundRecorder.cpp.
```

```cpp
void sf::SoundRecorder::Start ( unsigned int SampleRate = 44100 )
Start the capture.
Warning : only one capture can happen at the same time

**Parameters:**
: **Sound** frequency (the more samples, the higher the
```
Warning: only one capture can happen at the same time

Definition at line 69 of file SoundRecorder.cpp.

void sf::SoundRecorder::Stop ( )

Stop the capture.

Definition at line 115 of file SoundRecorder.cpp.

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

- SoundRecorder.hpp
- SoundRecorder.cpp

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
sf::SoundStream
sf::SoundStream Class Reference

**SoundStream** is a streamed sound, ie samples are acquired while the sound is playing. More...

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

Inheritance diagram for sf::SoundStream:

![Inheritance diagram](image)

List of all members.
## Classes

<table>
<thead>
<tr>
<th>struct</th>
<th>Chunk</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Structure defining a chunk of audio data to stream. More...</td>
</tr>
</tbody>
</table>
Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Enumeration of the sound states. More...</td>
</tr>
</tbody>
</table>
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
</table>
| virtual 
~SoundStream () | Virtual destructor.                              |
| void Play ()      | Start playing the audio stream.                  |
| void Stop ()      | Stop playing the audio stream.                   |
| unsigned int GetChannelsCount () const | Return the number of channels (1 = mono, 2 = stereo). |
| unsigned int GetSampleRate () const | Get the stream sample rate.                      |
| Status GetStatus () const | Get the status of the stream (stopped, paused, playing). |
| float GetPlayingOffset () const | Get the current playing position of the stream. |
| void SetLoop (bool Loop) | Set the stream loop state.                       |
| bool GetLoop () const | Tell whether or not the stream is looping.        |
Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SoundStream</strong> ()</td>
<td>Default constructor.</td>
</tr>
<tr>
<td><strong>Initialize</strong> (unsigned int ChannelsCount, unsigned int SampleRate)</td>
<td>Set the audio stream parameters, you must call it before <strong>Play</strong>().</td>
</tr>
</tbody>
</table>
Private Types

typedef void(* FuncType )(void *)
# Private Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Launch()</code></td>
<td>Create and run the thread.</td>
</tr>
<tr>
<td><code>Wait()</code></td>
<td>Wait until the thread finishes.</td>
</tr>
<tr>
<td><code>Terminate()</code></td>
<td>Terminate the thread. Terminating a thread with this function is not safe, you should rather try to make the thread function terminate by itself.</td>
</tr>
<tr>
<td><code>Pause()</code></td>
<td>Pause the sound.</td>
</tr>
<tr>
<td><code>SetBuffer(const SoundBuffer &amp;Buffer)</code></td>
<td>Set the source buffer.</td>
</tr>
<tr>
<td><code>SetPitch(float Pitch)</code></td>
<td>Set the sound pitch.</td>
</tr>
<tr>
<td><code>SetVolume(float Volume)</code></td>
<td>Set the sound volume.</td>
</tr>
<tr>
<td><code>GetPosition(float X, float Y, float Z)</code></td>
<td>Set the sound position (take 3 values).</td>
</tr>
<tr>
<td><code>GetPosition(const Vector3f &amp;Position)</code></td>
<td>Set the sound position (take a 3D vector).</td>
</tr>
<tr>
<td><code>SetRelativeToListener(bool Relative)</code></td>
<td>Make the sound's position relative to the listener's position, or absolute.</td>
</tr>
<tr>
<td><code>SetMinDistance(float MinDistance)</code></td>
<td>Set the minimum distance - closer than this distance, the listener will hear the sound at its maximum volume.</td>
</tr>
<tr>
<td><code>SetAttenuation(float Attenuation)</code></td>
<td>Set the attenuation factor - the higher the attenuation, the more the sound will be attenuated with distance from listener.</td>
</tr>
<tr>
<td><code>SetPlayingOffset(float TimeOffset)</code></td>
<td>Set the current playing position of the sound.</td>
</tr>
<tr>
<td><code>GetBuffer()</code></td>
<td>Get the source buffer.</td>
</tr>
<tr>
<td><code>GetPitch()</code></td>
<td>Get the pitch.</td>
</tr>
<tr>
<td><code>GetVolume()</code></td>
<td>Get the volume.</td>
</tr>
<tr>
<td><code>GetPosition()</code></td>
<td>Get the sound position.</td>
</tr>
<tr>
<td><code>IsRelativeToListener()</code></td>
<td>Tell if the sound's position is relative to the listener's position, or if it's absolute.</td>
</tr>
<tr>
<td><code>GetMinDistance()</code></td>
<td>Get the minimum distance.</td>
</tr>
<tr>
<td>Type</td>
<td>Function</td>
</tr>
<tr>
<td>--------</td>
<td>------------------</td>
</tr>
<tr>
<td>float</td>
<td><code>GetAttenuation()</code> const</td>
</tr>
<tr>
<td>void</td>
<td><code>ResetBuffer()</code></td>
</tr>
</tbody>
</table>
Detailed Description

**SoundStream** is a streamed sound, ie samples are acquired while the sound is playing.

Use it for big sounds that would require hundreds of MB in memory (see *Music*), or for streaming sound from the network

Definition at line 44 of file *SoundStream.hpp*. 
**Member Enumeration Documentation**

<table>
<thead>
<tr>
<th>enum sf::Sound::Status</th>
</tr>
</thead>
</table>

Enumeration of the sound states.

Definition at line 52 of file `Sound.hpp`. 
Constructor & Destructor Documentation

#### sf::SoundStream::~SoundStream ( ) [virtual]

Virtual destructor.

Definition at line 54 of file SoundStream.cpp.

#### sf::SoundStream::SoundStream ( ) [protected]

Default constructor.

Definition at line 39 of file SoundStream.cpp.
## Member Function Documentation

### `unsigned int sf::SoundStream::GetChannelsCount()` const

Return the number of channels (1 = mono, 2 = stereo).
Return the number of channels (1 = mono, 2 = stereo, .

**Returns:**
Number of channels

Definition at line 126 of file `SoundStream.cpp`.

### `bool sf::SoundStream::GetLoop()` const

Tell whether or not the stream is looping.
Tell whether or not the music is looping.

**Returns:**
True if the music is looping, false otherwise

Reimplemented from `sf::Sound`.

Definition at line 180 of file `SoundStream.cpp`.

### `float sf::SoundStream::GetPlayingOffset()` const

Get the current playing position of the stream.
Returns: Current playing position, expressed in seconds

Current playing position, expressed in seconds

Reimplemented from **sf::Sound**.

Definition at line 162 of file **SoundStream.cpp**.

---

**unsigned int sf::SoundStream::GetSampleRate( ) const**

Get the stream sample rate.

Get the sound frequency (sample rate).

**Returns:** Stream frequency (number of samples per second)

Definition at line 135 of file **SoundStream.cpp**.

---

**Sound::Status sf::SoundStream::GetStatus( ) const**

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

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

**Returns:** Current status of the sound

Reimplemented from **sf::Sound**.

Definition at line 144 of file **SoundStream.cpp**.
### void sf::SoundStream::Initialize(unsigned int ChannelsCount, unsigned int SampleRate)

Set the audio stream parameters, you must call it before `Play()`.

**Parameters:**
- `ChannelsCount` : Number of channels
- `SampleRate` : Sample rate

Definition at line 64 of file `SoundStream.cpp`.

### void sf::SoundStream::Play()

Start playing the audio stream.

Reimplemented from `sf::Sound`.

Definition at line 85 of file `SoundStream.cpp`.

### void sf::SoundStream::SetLoop(bool Loop)

Set the stream loop state.

Set the music loop state.

This parameter is disabled by default

**Parameters:**
- `Loop` : True to play in loop, false to play once

Reimplemented from `sf::Sound`.
void sf::SoundStream::Stop ( )

Stop playing the audio stream.

Reimplemented from sf::Sound.

Definition at line 115 of file SoundStream.cpp.

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

- SoundStream.hpp
- SoundStream.cpp

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
sf::SoundStream::Chunk
sf::SoundStream::SoundStream::Chunk

Struct Reference

Structure defining a chunk of audio data to stream. More...

#include <SoundStream.hpp>

List of all members.
### Public Attributes

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>const Int16 *</td>
<td>Samples</td>
<td>Pointer to the audio samples.</td>
</tr>
<tr>
<td>std::size_t</td>
<td>NbSamples</td>
<td>Number of samples pointed by Samples.</td>
</tr>
</tbody>
</table>
Detailed Description

Structure defining a chunk of audio data to stream.

Definition at line 69 of file `SoundStream.hpp`.
### Member Data Documentation

<table>
<thead>
<tr>
<th>std::size_t sf::SoundStream::SoundStream::Chunk::NbSamples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of samples pointed by Samples.</td>
</tr>
<tr>
<td>Definition at line 72 of file SoundStream.hpp.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>const Int16* sf::SoundStream::SoundStream::Chunk::Samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pointer to the audio samples.</td>
</tr>
<tr>
<td>Definition at line 71 of file SoundStream.hpp.</td>
</tr>
</tbody>
</table>

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

- **SoundStream.hpp**

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
sf::Sprite
sf::Sprite Class Reference

**Sprite** defines a sprite: texture, transformations, color, and draw on screen. [More...]

```cpp
#include <Sprite.hpp>
```

Inheritance diagram for sf::Sprite:

```
sf::Drawable

sf::Sprite
```

[List of all members.]

## Public Member Functions

**Sprite** ()
*Default constructor.*

**Sprite** (const **Image** &Img, const **Vector2f** &Position=**Vector2f**(0, 0), const **Vector2f** &Scale=**Vector2f**(1, 1), float Rotation=0.1, const **Color** &Col=**Color**(255, 255, 255, 255))
*Construct the sprite from a source image.*

void **SetImage** (const **Image** &Img)
*Change the image of the sprite.*

void **SetSubRect** (const **IntRect** &SubRect)
*Set the sub-rectangle of the sprite inside the source image.*

void **Resize** (float Width, float Height)
*Resize the sprite (by changing its scale factors) (take 2 values).*

void **Resize** (const **Vector2f** &Size)
*Resize the sprite (by changing its scale factors) (take a 2D vector).*

void **FlipX** (bool Flipped)
*Flip the sprite horizontally.*

void **FlipY** (bool Flipped)
*Flip the sprite vertically.*

const **Image** * **GetImage** () const
*Get the source image of the sprite.*

const **IntRect** & **GetSubRect** () const
*Get the sub-rectangle of the sprite inside the source image.*

**Vector2f** **GetSize** () const
*Get the sprite size.*

**Color** **GetPixel** (unsigned int X, unsigned int Y) const
*Get the color of a given pixel in the sprite (point is in local coordinates).*

void **SetPosition** (float X, float Y)
*Set the position of the object (take 2 values).*

void **SetPosition** (const **Vector2f** &Position)
*Set the position of the object (take a 2D vector).*

void **SetX** (float X)
*Set the X position of the object.*

void **SetY** (float Y)
*Set the Y position of the object.*

void **SetScale** (float ScaleX, float ScaleY)
*Set the scale of the object (take 2 values).*

void **SetScale** (const **Vector2f** &Scale)
*Set the scale of the object (take a 2D vector).*

void **SetScaleX** (float FactorX)
*Set the X scale factor of the object.*

void **SetScaleY** (float FactorY)
*Set the Y scale factor of the object.*
| void SetCenter(float CenterX, float CenterY) | Set the center of the object, in coordinates relative to the top-left of the object (take 2 values). |
| void SetCenter(const Vector2f &Center) | Set the center of the object, in coordinates relative to the top-left of the object (take a 2D vector). |
| void SetRotation(float Rotation) | Set the orientation of the object. |
| void SetColor(const Color &Col) | Set the color of the object. |
| void SetBlendMode(Blend::Mode Mode) | Set the blending mode for the object. |
| const Vector2f &GetPosition() const | Get the position of the object. |
| const Vector2f &GetScale() const | Get the current scale of the object. |
| const Vector2f &GetCenter() const | Get the center of the object. |
| float GetRotation() const | Get the orientation of the object. |
| const Color &GetColor() const | Get the color of the object. |
| Blend::Mode GetBlendMode() const | Get the current blending mode. |
| void Move(float OffsetX, float OffsetY) | Move the object of a given offset (take 2 values). |
| void Move(const Vector2f &Offset) | Move the object of a given offset (take a 2D vector). |
| void Scale(float FactorX, float FactorY) | Scale the object (take 2 values). |
| void Scale(const Vector2f &Factor) | Scale the object (take a 2D vector). |
| void Rotate(float Angle) | Rotate the object. |
| sf::Vector2f TransformToLocal(const sf::Vector2f &Point) const | Transform a point from global coordinates into local coordinates (ie it applies the inverse of object's center, translation, rotation and scale to the point). |
| sf::Vector2f TransformToGlobal(const sf::Vector2f &Point) const | Transform a point from local coordinates into global coordinates (ie it applies the object's center, translation, rotation and scale to the point). |
## Protected Member Functions

<table>
<thead>
<tr>
<th>virtual void Render(RenderTarget &amp;Target) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>/see Drawable::Render</td>
</tr>
<tr>
<td>const Matrix3 &amp; GetMatrix() const</td>
</tr>
<tr>
<td>Get the transform matrix of the drawable.</td>
</tr>
<tr>
<td>const Matrix3 &amp; GetInverseMatrix() const</td>
</tr>
<tr>
<td>Get the inverse transform matrix of the drawable.</td>
</tr>
</tbody>
</table>
Detailed Description

**Sprite** defines a sprite: texture, transformations, color, and draw on screen.

Definition at line 44 of file **Sprite.hpp**.
## Constructor & Destructor Documentation

### sf::Sprite::Sprite ( )

Default constructor.

Definition at line 38 of file `Sprite.cpp`.

```cpp
sf::Sprite::Sprite ( const Image & Img,
 const Vector2f & Position = Vector2f(0, 0),
 const Vector2f & Scale = Vector2f(1, 1),
 float Rotation = 0.f,
 const Color & Col = Color(255, 255, 255, 255) ) [explicit]
```

Construct the sprite from a source image.

### Parameters:

- **Img**: Image of the sprite
- **Position**: Position of the sprite (0, 0 by default)
- **Scale**: Scale factor (1, 1 by default)
- **Rotation**: Orientation, in degrees (0 by default)
- **Col**: Color of the sprite (white by default)

Definition at line 50 of file `Sprite.cpp`. 
Member Function Documentation

void sf::Sprite::FlipX ( bool Flipped )

Flip the sprite horizontally.

Parameters:
   Flipped : True to flip the sprite

Definition at line 112 of file Sprite.cpp.

void sf::Sprite::FlipY ( bool Flipped )

Flip the sprite vertically.

Parameters:
   Flipped : True to flip the sprite

Definition at line 121 of file Sprite.cpp.

Blend::Mode sf::Drawable::GetBlendMode ( ) const [inherited]

Get the current blending mode.

Returns:
   Current blending mode

Definition at line 258 of file Drawable.cpp.

const Vector2f & sf::Drawable::GetCenter ( ) const [inherited]
Get the center of the object.

**Returns:**
Current position of the center

Definition at line 231 of file `Drawable.cpp`.

```cpp
const Color & sf::Drawable::GetColor() const [inherited]
```

Get the color of the object.

**Returns:**
Current color

Definition at line 249 of file `Drawable.cpp`.

```cpp
const Image * sf::Sprite::GetImage() const
```

Get the source image of the sprite.

**Returns:**
Pointer to the image (can be NULL)

Definition at line 130 of file `Sprite.cpp`.

```cpp
const Matrix3 & sf::Drawable::GetInverseMatrix() const [protected, inherited]
```

Get the inverse transform matrix of the drawable.

**Returns:**
Inverse transform matrix
const Matrix3 & sf::Drawable::GetMatrix() const [protected, inherited]

Get the transform matrix of the drawable.

Returns:
Transform matrix

Definition at line 350 of file Drawable.cpp.

Color sf::Sprite::GetPixel(unsigned int X, unsigned int Y) const

Get the color of a given pixel in the sprite (point is in local coordinates).

Parameters:
X : X coordinate of the pixel to get
Y : Y coordinate of the pixel to get

Returns:
Color of pixel (X, Y)

Definition at line 334 of file Drawable.cpp.

const Vector2f & sf::Drawable::GetPosition() const [inherited]

Definition at line 158 of file Sprite.cpp.
Get the position of the object.

**Returns:**
Current position

Definition at line 213 of file `Drawable.cpp`.

```cpp
float sf::Drawable::GetRotation() const [inherited]
```

Get the orientation of the object.

Rotation is always in the range [0, 360]

**Returns:**
Current rotation, in degrees

Definition at line 240 of file `Drawable.cpp`.

```cpp
const Vector2f & sf::Drawable::GetScale() const [inherited]
```

Get the current scale of the object.

**Returns:**
Current scale factor (always positive)

Definition at line 222 of file `Drawable.cpp`.

```cpp
Vector2f sf::Sprite::GetSize() const
```

Get the sprite size.

**Returns:**
Size of the sprite

Definition at line 148 of file Sprite.cpp.

```cpp
const IntRect & sf::Sprite::GetSubRect () const
```

Get the sub-rectangle of the sprite inside the source image.

**Returns:**
Sub-rectangle

Definition at line 139 of file Sprite.cpp.

```cpp
void sf::Drawable::Move ( const Vector2f & Offset ) [inherited]
```

Move the object of a given offset (take a 2D vector).

**Parameters:**
Offset : Amount of units to move the object of

Definition at line 278 of file Drawable.cpp.

```cpp
void sf::Drawable::Move ( float OffsetX, float OffsetY ) [inherited]
```

Move the object of a given offset (take 2 values).

**Parameters:**
OffsetX : X offset
OffsetY : Y offset
### sf::Sprite::Render

```cpp
definition of the file Drawable.cpp.

```void sf::Sprite::Render (RenderTarget & Target) const [protected, virtual]

/see Drawable::Render

/see sfDrawable::Render

Implements sf::Drawable.

Definition at line 180 of file Sprite.cpp.
```

### sf::Sprite::Resize

```cpp
void sf::Sprite::Resize (const Vector2f & Size)

Resize the sprite (by changing its scale factors) (take a 2D vector).

Resize the object (by changing its scale factors) (take a 2D vector) The default size is defined by the subrect.

The default size is defined by the subrect

**Parameters:**

- `Size`: New size (both coordinates must be strictly positive)

Definition at line 103 of file Sprite.cpp.
```

### sf::Sprite::Resize

```cpp
void sf::Sprite::Resize (float Width, float Height)

Resize the sprite (by changing its scale factors) (take 2
```
The default size is defined by the subrect

**Parameters:**

- **Width**: New width (must be strictly positive)
- **Height**: New height (must be strictly positive)

Definition at line 89 of file `Sprite.cpp`.

---

```cpp
void sf::Drawable::Rotate(float Angle) [inherited]
```

Rotate the object.

**Parameters:**

- **Angle**: Angle of rotation, in degrees

Definition at line 306 of file `Drawable.cpp`.

---

```cpp
void sf::Drawable::Scale(const Vector2f & Factor) [inherited]
```

Scale the object (take a 2D vector).

**Parameters:**

- **Factor**: Scaling factors (both values must be strictly positive)

Definition at line 297 of file `Drawable.cpp`.

---

```cpp
void sf::Drawable::Scale(float FactorX, float FactorY) [inherited]
```
Scale the object (take 2 values).

**Parameters:**
- `FactorX`: Scaling factor on X (must be strictly positive)
- `FactorY`: Scaling factor on Y (must be strictly positive)

Definition at line 287 of file `Drawable.cpp`.

```cpp
void sf::Drawable::SetBlendMode ( Blend::Mode Mode ) [inherited]
```

Set the blending mode for the object.

The default blend mode is `Blend::Alpha`

**Parameters:**
- `Mode`: New blending mode

The default blend mode is `Blend::Alpha`

Definition at line 204 of file `Drawable.cpp`.

```cpp
void sf::Drawable::SetCenter ( const Vector2f & Center ) [inherited]
```

Set the center of the object, in coordinates relative to the top-left of the object (take a 2D vector).

Set the center of the object, in coordinates relative to the top-left of the object (take a 2D vector).

The default center is (0, 0)

**Parameters:**

The default center is (0, 0)

Definition at line 171 of file \texttt{Drawable.cpp}.

\begin{verbatim}
void sf::Drawable::SetCenter(float centerX, float centerY)
    \[\text{inherited}\]
\end{verbatim}

Set the center of the object, in coordinates relative to the top-left of the object (take 2 values).

The default center is (0, 0)

\textbf{Parameters:}

- \textit{CenterX} : \textit{X} coordinate of the center
- \textit{CenterY} : \textit{Y} coordinate of the center

Definition at line 157 of file \texttt{Drawable.cpp}.

\begin{verbatim}
void sf::Drawable::SetColor(const Color & Col)
    \[\text{inherited}\]
\end{verbatim}

Set the color of the object.

The default color is white

\textbf{Parameters:}

- \textit{Col} : New color
The default color is white

Definition at line 194 of file Drawable.cpp.

```cpp
void sf::Sprite::SetImage (const Image & Img )
```

Change the image of the sprite.

Set the image of the sprite.

**Parameters:**

- `Img`: New image

Definition at line 63 of file Sprite.cpp.

```cpp
void sf::Drawable::SetPosition (const Vector2f & Position ) [inherited]
```

Set the position of the object (take a 2D vector).

**Parameters:**

- `Position`: New position

Definition at line 75 of file Drawable.cpp.

```cpp
void sf::Drawable::SetPosition ( float X,
                                 float Y ) [inherited]
```

Set the position of the object (take 2 values).

**Parameters:**

- `X`: New X coordinate
Y : New Y coordinate

Definition at line 65 of file Drawable.cpp.

```cpp
void sf::Drawable::SetRotation ( float Rotation ) [inherited]
```

Set the orientation of the object.

**Parameters:**

*Rotation* : Angle of rotation, in degrees

Definition at line 180 of file Drawable.cpp.

```cpp
void sf::Drawable::SetScale ( const Vector2f & Scale ) [inherited]
```

Set the scale of the object (take a 2D vector).

**Parameters:**

*Scale* : New scale (both values must be strictly positive)

Definition at line 117 of file Drawable.cpp.

```cpp
void sf::Drawable::SetScale ( float ScaleX, float ScaleY ) [inherited]
```

Set the scale of the object (take 2 values).

**Parameters:**

*ScaleX* : New horizontal scale (must be strictly positive)
*ScaleY* : New vertical scale (must be strictly positive)
void sf::Drawable::SetScaleX(float FactorX) [inherited]

Set the X scale factor of the object.

**Parameters:**
- X: New X scale factor

Definition at line 107 of file Drawable.cpp.

void sf::Drawable::SetScaleY(float FactorY) [inherited]

Set the Y scale factor of the object.

**Parameters:**
- Y: New Y scale factor

Definition at line 127 of file Drawable.cpp.

void sf::Sprite::SetSubRect(const IntRect & SubRect)

Set the sub-rectangle of the sprite inside the source image.

By default, the subrect covers the entire source image

**Parameters:**
- SubRect: New sub-rectangle

Definition at line 79 of file Sprite.cpp.

void sf::Drawable::SetX(float X) [inherited]
Set the X position of the object.

**Parameters:**
- \( X \) : New X coordinate

Definition at line 85 of file *Drawable.cpp*.

```cpp
void sf::Drawable::SetY(float Y) [inherited]
```

Set the Y position of the object.

**Parameters:**
- \( Y \) : New Y coordinate

Definition at line 96 of file *Drawable.cpp*.

```cpp
sf::Vector2f sf::Drawable::TransformToGlobal(const sf::Vector2f & Point)
```

Transform a point from local coordinates into global coordinates (ie it applies the object's center, translation, rotation and scale to the point).

Transform a point from local coordinates into global coordinates (ie it applies the object's center, translation, rotation and scale to the point).

**Parameters:**
- \( Point \) : Point to transform

**Returns:**
- Transformed point
Transform a point from global coordinates into local coordinates (ie it applies the inverse of object's center, translation, rotation and scale to the point).

Parameters:
- **Point**: Point to transform

Returns:
- Transformed point

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

- `Sprite.hpp`
- `Sprite.cpp`

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
sf::String
sf::String Class Reference

**String** defines a graphical 2D text, that can be drawn on screen.  
[More...](#)

```cpp
#include <String.hpp>
```

Inheritance diagram for sf::String:

```
sf::Drawable
    `------------`
    |            |
    v            v
sf::String
```

[List of all members.](#)
## Public Types

```
enum Style {
    Regular = 0,
    Bold = 1 << 0,
    Italic = 1 << 1,
    Underlined = 1 << 2
}
```

Enumerate the string drawing styles. [More...](#)
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>String()</code></td>
<td>Default constructor.</td>
</tr>
<tr>
<td><code>String(const Unicode::Text &amp;Text, const Font &amp;CharFont=Font::GetDefaultFont(), float Size=30.f)</code></td>
<td>Construct the string from any kind of text.</td>
</tr>
<tr>
<td><code>void SetText(const Unicode::Text &amp;Text)</code></td>
<td>Set the text (from any kind of string).</td>
</tr>
<tr>
<td><code>void SetFont(const Font &amp;CharFont)</code></td>
<td>Set the font of the string.</td>
</tr>
<tr>
<td><code>void SetSize(float Size)</code></td>
<td>Set the size of the string The default size is 30.</td>
</tr>
<tr>
<td><code>void SetStyle(unsigned long TextStyle)</code></td>
<td>Set the style of the text The default style is Regular.</td>
</tr>
<tr>
<td><code>const Unicode::Text &amp; GetText()</code> const</td>
<td>Get the text (the returned text can be converted implicitly to any kind of string).</td>
</tr>
<tr>
<td><code>const Font &amp; GetFont()</code> const</td>
<td>Get the font used by the string.</td>
</tr>
<tr>
<td><code>float GetSize()</code> const</td>
<td>Get the size of the characters.</td>
</tr>
<tr>
<td><code>unsigned long GetStyle()</code> const</td>
<td>Get the style of the text.</td>
</tr>
<tr>
<td><code>sf::Vector2f GetCharacterPos(std::size_t Index)</code> const</td>
<td>Return the visual position of the Index-th character of the string, in coordinates relative to the string (note: translation, center, rotation and scale are not applied).</td>
</tr>
<tr>
<td><code>FloatRect GetRect()</code> const</td>
<td>Get the string rectangle on screen.</td>
</tr>
<tr>
<td><code>void SetPosition(float X, float Y)</code></td>
<td>Set the position of the object (take 2 values).</td>
</tr>
<tr>
<td><code>void SetPosition(const Vector2f &amp;Position)</code></td>
<td>Set the position of the object (take a 2D vector).</td>
</tr>
<tr>
<td><code>void SetX(float X)</code></td>
<td>Set the X position of the object.</td>
</tr>
<tr>
<td><code>void SetY(float Y)</code></td>
<td>Set the Y position of the object.</td>
</tr>
<tr>
<td><code>void SetScale(float ScaleX, float ScaleY)</code></td>
<td>Set the scale of the object (take 2 values).</td>
</tr>
<tr>
<td><code>void SetScale(const Vector2f &amp;Scale)</code></td>
<td>Set the scale of the object (take a 2D vector).</td>
</tr>
<tr>
<td><code>void SetScaleX(float FactorX)</code></td>
<td>Set the X scale factor of the object.</td>
</tr>
<tr>
<td><code>void SetScaleY(float FactorY)</code></td>
<td></td>
</tr>
</tbody>
</table>
Set the Y scale factor of the object.

void **SetCenter** (float CenterX, float CenterY)
Set the center of the object, in coordinates relative to the top-left of the object (take 2 values).

void **SetCenter** (const Vector2f &Center)
Set the center of the object, in coordinates relative to the top-left of the object (take a 2D vector).

void **SetRotation** (float Rotation)
Set the orientation of the object.

void **SetColor** (const Color &Col)
Set the color of the object.

void **SetBlendMode** (Blend::Mode Mode)
Set the blending mode for the object.

const Vector2f & **GetPosition** () const
Get the position of the object.

const Vector2f & **GetScale** () const
Get the current scale of the object.

const Vector2f & **GetCenter** () const
Get the center of the object.

float **GetRotation** () const
Get the orientation of the object.

const Color & **GetColor** () const
Get the color of the object.

Blend::Mode **GetBlendMode** () const
Get the current blending mode.

void **Move** (float OffsetX, float OffsetY)
Move the object of a given offset (take 2 values).

void **Move** (const Vector2f &Offset)
Move the object of a given offset (take a 2D vector).

void **Scale** (float FactorX, float FactorY)
Scale the object (take 2 values).

void **Scale** (const Vector2f &Factor)
Scale the object (take a 2D vector).

void **Rotate** (float Angle)
Rotate the object.

sf::Vector2f **TransformToLocal** (const sf::Vector2f &Point) const
Transform a point from global coordinates into local coordinates (ie it applies the inverse of object's center, translation, rotation and scale to the point).

sf::Vector2f **TransformToGlobal** (const sf::Vector2f &Point) const
Transform a point from local coordinates into global coordinates (ie it applies the object's center, translation, rotation and scale to the point).
## Protected Member Functions

<table>
<thead>
<tr>
<th>virtual void</th>
<th>Render (RenderTarget &amp;Target) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>/see Drawable::Render</td>
<td></td>
</tr>
<tr>
<td>const Matrix3 &amp;</td>
<td>GetMatrix () const</td>
</tr>
<tr>
<td>Get the transform matrix of the drawable.</td>
<td></td>
</tr>
<tr>
<td>const Matrix3 &amp;</td>
<td>GetInverseMatrix () const</td>
</tr>
<tr>
<td>Get the inverse transform matrix of the drawable.</td>
<td></td>
</tr>
</tbody>
</table>
**Detailed Description**

*String* defines a graphical 2D text, that can be drawn on screen.

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

-----------------------------
**Member Enumeration Documentation**

<table>
<thead>
<tr>
<th><strong>enum sf::String::Style</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Enumerate the string drawing styles.</td>
</tr>
</tbody>
</table>

**Enumerator:**

- **Regular**  
  Regular characters, no style.
- **Bold**  
  Characters are bold.
- **Italic**  
  Characters are in italic.
- **Underlined**  
  Characters are underlined.

Definition at line 51 of file `String.hpp`. 
Constructor & Destructor Documentation

sf::String::String ( )

Default constructor.

Definition at line 39 of file String.cpp.

sf::String::String ( const Unicode::Text & Text, const Font & CharFont = Font::GetDefaultFont(), float Size = 30.f )

Construct the string from any kind of text.

Parameters:
    Text : Text assigned to the string
    Font : Font used to draw the string (SFML built-in font by default)
    Size : Characters size (30 by default)

Definition at line 52 of file String.cpp.
### Member Function Documentation

<table>
<thead>
<tr>
<th>Function</th>
<th>Documentation</th>
</tr>
</thead>
</table>
| **Blend::Mode** sf::Drawable::GetBlendMode ( ) const [inherited] | Get the current blending mode.  
**Returns:**  
Current blending mode  
Definition at line 258 of file **Drawable.cpp**. |
| **const Vector2f & sf::Drawable::GetCenter ( ) const [inherited]** | Get the center of the object.  
**Returns:**  
Current position of the center  
Definition at line 231 of file **Drawable.cpp**. |
| **sf::Vector2f sf::String::GetCharacterPos ( std::size_t Index ) const** | Return the visual position of the Index-th character of the string, in coordinates relative to the string (note : translation, center, rotation and scale are not applied).  
Return the visual position of the Index-th character of the string, in coordinates relative to the string (note : translation, center, rotation and scale are not applied).  
**Parameters:**  
*Index* : Index of the character |
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Definition at line</th>
<th>File</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sf::Drawable::GetColor()</code></td>
<td>Get the color of the object.</td>
<td>153</td>
<td>String.cpp</td>
</tr>
<tr>
<td><code>sf::String::GetFont()</code></td>
<td>Get the font used by the string.</td>
<td>249</td>
<td>Drawable.cpp</td>
</tr>
<tr>
<td><code>sf::Drawable::GetInverseMatrix()</code></td>
<td>Get the inverse transform matrix of the drawable.</td>
<td>350</td>
<td>Drawable.cpp</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>Returns</td>
<td>Definition</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------------------------------------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>const Matrix3 &amp; sf::Drawable::GetMatrix()</code></td>
<td>Get the transform matrix of the drawable.</td>
<td>Transform matrix</td>
<td>Definition at line 334 of file <code>Drawable.cpp</code>.</td>
</tr>
<tr>
<td><code>const Vector2f &amp; sf::Drawable::GetPosition()</code></td>
<td>Get the position of the object.</td>
<td>Current position</td>
<td>Definition at line 213 of file <code>Drawable.cpp</code>.</td>
</tr>
<tr>
<td><code>FloatRect sf::String::GetRect()</code></td>
<td>Get the string rectangle on screen.</td>
<td>Rectangle containing the string in screen coordinates</td>
<td>Definition at line 195 of file <code>String.cpp</code>.</td>
</tr>
<tr>
<td><code>float sf::Drawable::GetRotation()</code></td>
<td>Get the orientation of the object.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Rotation is always in the range $[0, 360]$

**Returns:**
Current rotation, in degrees

Definition at line **240** of file *Drawable.cpp*.

<table>
<thead>
<tr>
<th>const Vector2f &amp; sf::Drawable::GetScale ( ) const [inherited]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get the current scale of the object.</td>
</tr>
</tbody>
</table>

**Returns:**
Current scale factor (always positive)

Definition at line **222** of file *Drawable.cpp*.

<table>
<thead>
<tr>
<th>float sf::String::GetSize ( ) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get the size of the characters.</td>
</tr>
</tbody>
</table>

**Returns:**
Size of the characters

Definition at line **133** of file *String.cpp*.

<table>
<thead>
<tr>
<th>unsigned long sf::String::GetStyle ( ) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get the style of the text.</td>
</tr>
</tbody>
</table>

**Returns:**
Current string style (combination of *Style* enum values)
**const Unicode::Text & sf::String::GetText ( ) const**

Get the text (the returned text can be converted implicitly to any kind of string).

**Returns:**
String's text

Definition at line 142 of file String.cpp.

**void sf::Drawable::Move ( const Vector2f & Offset ) [inherited]**

Move the object of a given offset (take a 2D vector).

**Parameters:**
Offset : Amount of units to move the object of

Definition at line 115 of file String.cpp.

**void sf::Drawable::Move ( float OffsetX, float OffsetY ) [inherited]**

Move the object of a given offset (take 2 values).

**Parameters:**
OffsetX : X offset
OffsetY : Y offset

Definition at line 278 of file Drawable.cpp.

**void sf::Drawable::Move ( float OffsetX, float OffsetY ) [inherited]**

Move the object of a given offset (take 2 values).

**Parameters:**
OffsetX : X offset
OffsetY : Y offset

Definition at line 268 of file Drawable.cpp.
void sf::String::Render (RenderTarget & Target) const [protected, virtual]

/see Drawable::Render

/see sfDrawable::Render

Implements sf::Drawable.

Definition at line 213 of file String.cpp.

void sf::Drawable::Rotate (float Angle) [inherited]

Rotate the object.

Parameters:

Angle : Angle of rotation, in degrees

Definition at line 306 of file Drawable.cpp.

void sf::Drawable::Scale (const Vector2f & Factor) [inherited]

Scale the object (take a 2D vector).

Parameters:

Factor : Scaling factors (both values must be strictly positive)

Definition at line 297 of file Drawable.cpp.

void sf::Drawable::Scale (float FactorX, float FactorY) [inherited]
Scale the object (take 2 values).

**Parameters:**

*FactorX* : Scaling factor on X (must be strictly positive)

*FactorY* : Scaling factor on Y (must be strictly positive)

Definition at line 287 of file **Drawable.cpp**.

```cpp
void sf::Drawable::SetBlendMode ( Blend::Mode Mode ) [inherited]
```

Set the blending mode for the object.

The default blend mode is **Blend::Alpha**

**Parameters:**

*Mode* : New blending mode

The default blend mode is **Blend::Alpha**

Definition at line 204 of file **Drawable.cpp**.

```cpp
void sf::Drawable::SetCenter ( const Vector2f & Center ) [inherited]
```

Set the center of the object, in coordinates relative to the top-left of the object (take a 2D vector).

Set the center of the object, in coordinates relative to the top-left of the object (take a 2D vector).

The default center is (0, 0)

**Parameters:**

*Center* : New center
The default center is (0, 0)

Definition at line 171 of file Drawable.cpp.

```cpp
class Drawable
{
public:
    void SetCenter(float CenterX, float CenterY) [inherited]

    Set the center of the object, in coordinates relative to the top-left of the object (take 2 values).

    The default center is (0, 0)

    Parameters:
        CenterX : X coordinate of the center
        CenterY : Y coordinate of the center

    Definition at line 157 of file Drawable.cpp.

    void SetColor(const Color & Col) [inherited]

    Set the color of the object.

    The default color is white

    Parameters:
        Col : New color
```
The default color is white

Definition at line 194 of file Drawable.cpp.

```cpp
void sf::String::SetFont(const Font & CharFont)
```

Set the font of the string.

**Parameters:**
- `Font`: Font to use

Definition at line 75 of file String.cpp.

```cpp
void sf::Drawable::SetPosition(const Vector2f & Position)
```

Set the position of the object (take a 2D vector).

**Parameters:**
- `Position`: New position

Definition at line 75 of file Drawable.cpp.

```cpp
void sf::Drawable::SetPosition(float X, float Y)
```

Set the position of the object (take 2 values).

**Parameters:**
- `X`: New X coordinate
- `Y`: New Y coordinate
### Definition at line 65 of file `Drawable.cpp`

**void sf::Drawable::SetRotation ( float Rotation ) [inherited]**

Set the orientation of the object.

**Parameters:**
- *Rotation*: Angle of rotation, in degrees

**Definition at line 180 of file `Drawable.cpp`**

### Definition at line 117 of file `Drawable.cpp`

**void sf::Drawable::SetScale ( const Vector2f & Scale ) [inherited]**

Set the scale of the object (take a 2D vector).

**Parameters:**
- *Scale*: New scale (both values must be strictly positive)

**Definition at line 117 of file `Drawable.cpp`**

### Definition at line 107 of file `Drawable.cpp`

**void sf::Drawable::SetScale ( float ScaleX, float ScaleY ) [inherited]**

Set the scale of the object (take 2 values).

**Parameters:**
- *ScaleX*: New horizontal scale (must be strictly positive)
- *ScaleY*: New vertical scale (must be strictly positive)

**Definition at line 107 of file `Drawable.cpp`**
void sf::Drawable::SetScaleX (float \textit{FactorX} ) [inherited]

Set the X scale factor of the object.

\textbf{Parameters:}

\begin{itemize}
\item [$X$] : New X scale factor
\end{itemize}

Definition at line 127 of file Drawable.cpp.

---

void sf::Drawable::SetScaleY (float \textit{FactorY} ) [inherited]

Set the Y scale factor of the object.

\textbf{Parameters:}

\begin{itemize}
\item [$Y$] : New Y scale factor
\end{itemize}

Definition at line 141 of file Drawable.cpp.

---

void sf::String::SetSize (float \textit{Size} )

Set the size of the string The default size is 30.

Set the size of the string.

\textbf{Parameters:}

\begin{itemize}
\item [$Size$] : New size, in pixels
\end{itemize}

Definition at line 88 of file String.cpp.

---

void sf::String::SetStyle (unsigned long \textit{TextStyle} )

Set the style of the text The default style is Regular.
Set the style of the text The default style is Regular.

**Parameters:**

- `TextStyle` : New text style, (combination of `Style` enum values)

Definition at line 102 of file `String.cpp`.

```cpp
void sf::String::SetText ( const Unicode::Text & Text )
```

Set the text (from any kind of string).

**Parameters:**

- `Text` : New text

Definition at line 65 of file `String.cpp`.

```cpp
void sf::Drawable::SetX ( float X ) [inherited]
```

Set the X position of the object.

**Parameters:**

- `X` : New X coordinate

Definition at line 85 of file `Drawable.cpp`.

```cpp
void sf::Drawable::SetY ( float Y ) [inherited]
```

Set the Y position of the object.

**Parameters:**

- `Y` : New Y coordinate
sf::Vector2f sf::Drawable::TransformToGlobal ( const sf::Vector2f & Point )

Transform a point from local coordinates into global coordinates (ie it applies the object's center, translation, rotation and scale to the point).

Parameters:
   Point : Point to transform

Returns:
   Transformed point

Definition at line 325 of file Drawable.cpp.

sf::Vector2f sf::Drawable::TransformToLocal ( const sf::Vector2f & Point ) const

Transform a point from global coordinates into local coordinates (ie it applies the inverse of object's center, translation, rotation and scale to the point).

Parameters:
   Point : Point to transform
Returns:
Transformed point

Definition at line 316 of file Drawable.cpp.

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

- String.hpp
- String.cpp
sf::Thread
**sf::Thread Class Reference**

**Thread** defines an easy way to manipulate a thread. [More...](#)

```cpp
#include <Thread.hpp>
```

Inheritance diagram for sf::Thread:

![Inheritance Diagram](image)

[List of all members.](#)
Public Types

typedef void(* FuncType )(void *)
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Thread</strong> (FuncType Function, void *UserData=NULL)</td>
<td>Construct the thread from a function pointer.</td>
</tr>
<tr>
<td><code>virtual</code> <code>~Thread</code> ()</td>
<td>Virtual destructor.</td>
</tr>
<tr>
<td><code>void</code> <code>Launch</code> ()</td>
<td>Create and run the thread.</td>
</tr>
<tr>
<td><code>void</code> <code>Wait</code> ()</td>
<td>Wait until the thread finishes.</td>
</tr>
<tr>
<td><code>void</code> <code>Terminate</code> ()</td>
<td>Terminate the thread Terminating a thread with this function is not safe, you should rather try to make the thread function terminate by itself.</td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Thread ()</strong></td>
<td>Default constructor.</td>
</tr>
</tbody>
</table>
Detailed Description

Thread defines an easy way to manipulate a thread.

There are two ways to use Thread:

- Inherit from it and override the Run() virtual function
- Construct a Thread instance and pass it a function pointer to call

Definition at line 44 of file Win32/Thread.hpp.
Constructor & Destructor Documentation

sf::Thread::Thread ( Thread::FuncType  
     Function,  
     void *  
     UserData = NULL  
 )

Construct the thread from a function pointer.

Parameters:

Function : Entry point of the thread
UserData  : Data to pass to the thread function (NULL by default)

Definition at line 50 of file Thread.cpp.

sf::Thread::~Thread ( ) [virtual]

Virtual destructor.

Definition at line 62 of file Thread.cpp.

sf::Thread::Thread ( ) [protected]

Default constructor.

Definition at line 38 of file Thread.cpp.
Member Function Documentation

void sf::Thread::Launch ( )

Create and run the thread.

Definition at line 72 of file Thread.cpp.

void sf::Thread::Terminate ( )

Terminate the thread Terminating a thread with this function is not safe, you should rather try to make the thread function terminate by itself.

Terminate the thread Terminating a thread with this function is not safe, you should rather try to make the thread function terminate by itself.

Definition at line 109 of file Thread.cpp.

void sf::Thread::Wait ( )

Wait until the thread finishes.

Definition at line 89 of file Thread.cpp.

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

- Win32/Thread.hpp
sf::Unicode
sf::Unicode Class Reference

Provides utility functions to convert from and to any unicode and ASCII encoding. More...

#include <Unicode.hpp>

List of all members.
### Classes

<table>
<thead>
<tr>
<th>class</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>This class is an abstract definition of a unicode text, it can be converted from and to any kind of string and encoding.</em> <a href="#">More...</a></td>
</tr>
</tbody>
</table>
## Public Types

<table>
<thead>
<tr>
<th>Type Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>typedef std::basic_string&lt;Uint8&gt;</code> UTF8String</td>
<td>Define a string type for each encoding Warning: in UTF8 and UTF16 strings, one element doesn't necessarily maps to one character; only an UTF32 element is wide enough to hold all possible unicode values.</td>
</tr>
<tr>
<td><code>typedef std::basic_string&lt;Uint16&gt;</code> UTF16String</td>
<td></td>
</tr>
<tr>
<td><code>typedef std::basic_string&lt;Uint32&gt;</code> UTF32String</td>
<td></td>
</tr>
</tbody>
</table>
## Public Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>template&lt;typename In , typename Out &gt;</td>
<td>UTF32ToANSI (In Begin, In End, Out Output, char Replacement, const std::locale &amp;Locale)</td>
<td>Generic function to convert an UTF-32 characters range to an ANSI characters range, using the given locale.</td>
</tr>
<tr>
<td>template&lt;typename In , typename Out &gt;</td>
<td>ANSIToUTF32 (In Begin, In End, Out Output, const std::locale &amp;Locale)</td>
<td>Generic function to convert an ANSI characters range to an UTF-32 characters range, using the given locale.</td>
</tr>
<tr>
<td>template&lt;typename In , typename Out &gt;</td>
<td>UTF8ToUTF16 (In Begin, In End, Out Output, Uint16 Replacement)</td>
<td>Generic function to convert an UTF-8 characters range to an UTF-16 characters range, using the given locale.</td>
</tr>
<tr>
<td>template&lt;typename In , typename Out &gt;</td>
<td>UTF8ToUTF32 (In Begin, In End, Out Output, Uint32 Replacement)</td>
<td>Generic function to convert an UTF-8 characters range to an UTF-32 characters range, using the given locale.</td>
</tr>
<tr>
<td>template&lt;typename In , typename Out &gt;</td>
<td>UTF16ToUTF8 (In Begin, In End, Out Output, Uint8 Replacement)</td>
<td>Generic function to convert an UTF-16 characters range to an UTF-8 characters range, using the given locale.</td>
</tr>
<tr>
<td>template&lt;typename In , typename Out &gt;</td>
<td>UTF16ToUTF32 (In Begin, In End, Out Output, Uint32 Replacement)</td>
<td>Generic function to convert an UTF-16 characters range to an UTF-32 characters range, using the given locale.</td>
</tr>
<tr>
<td>template&lt;typename In , typename Out &gt;</td>
<td>UTF32ToUTF8 (In Begin, In End, Out Output, Uint8 Replacement)</td>
<td>Generic function to convert an UTF-32 characters range to an UTF-8 characters range, using the given locale.</td>
</tr>
<tr>
<td>template&lt;typename In &gt;</td>
<td>std::size_t GetUTF32Length (In Begin, In End)</td>
<td>Get the number of characters composing an UTF-32 string.</td>
</tr>
<tr>
<td>template&lt;typename In &gt;</td>
<td>std::size_t GetUTF16Length (In Begin, In End)</td>
<td>Get the number of characters composing an UTF-16 string.</td>
</tr>
<tr>
<td>template&lt;typename In &gt;</td>
<td>std::size_t GetUTF32Length (In Begin, In End)</td>
<td>Get the number of characters composing an UTF-32 string.</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

```cpp
template<typename In , typename Out >
static Out UTF32ToANSI (In Begin, In End, Out Output, char Replacement= '?', const std::locale &Locale= GetDefaultLocale())

Generic function to convert an UTF-32 characters range to an ANSI characters range, using the given locale.

template<typename In , typename Out >
static Out UTF32ToUTF16 (In Begin, In End, Out Output, char Replacement= '?', const std::locale &Locale= GetDefaultLocale())

Generic function to convert an UTF-32 characters range to an UTF-16 characters range, using the given locale.

template<typename In , typename Out >
static Out UTF8ToUTF32 (In Begin, In End, Out Output, char Replacement= '?', const std::locale &Locale= GetDefaultLocale())

Generic function to convert an UTF-8 characters range to an UTF-32 characters range, using the given locale.

template<typename In , typename Out >
static Out UTF8ToUTF16 (In Begin, In End, Out Output, char Replacement= '?', const std::locale &Locale= GetDefaultLocale())

Generic function to convert an UTF-8 characters range to an UTF-16 characters range, using the given locale.

template<typename In , typename Out >
static Out UTF8ToUTF32 (In Begin, In End, Out Output, Uint16 Replacement= '?')

Generic function to convert an UTF-8 characters range to an UTF-32 characters range, using the given locale.

template<typename In , typename Out >
static Out UTF8ToUTF16 (In Begin, In End, Out Output, Uint16 Replacement= '?')

Generic function to convert an UTF-8 characters range to an UTF-16 characters range, using the given locale.

template<typename In , typename Out >
static Out UTF16ToUTF8 (In Begin, In End, Out Output, Uint8 Replacement= '?')

Generic function to convert an UTF-16 characters range to an UTF-8 characters range, using the given locale.

template<typename In , typename Out >
static Out UTF16ToUTF32 (In Begin, In End, Out Output, Uint16 Replacement= '?')

Generic function to convert an UTF-16 characters range to an UTF-32 characters range, using the given locale.

template<typename In , typename Out >
static Out UTF16ToUTF8 (In Begin, In End, Out Output, Uint8 Replacement= '?')

Generic function to convert an UTF-16 characters range to an UTF-8 characters range, using the given locale.

template<typename In , typename Out >
static Out UTF32ToUTF16 (In Begin, In End, Out Output, Uint16 Replacement= '?')

Generic function to convert an UTF-32 characters range to an UTF-16 characters range, using the given locale.

template<typename In >
static std::size_t GetUTF8Length (In Begin, In End)

Get the number of characters composing an UTF-8 string.

template<typename In >
static std::size_t GetUTF16Length (In Begin, In End)

Get the number of characters composing an UTF-16 string.

template<typename In >
static std::size_t GetUTF32Length (In Begin, In End)

Get the number of characters composing an UTF-32 string.
Detailed Description

Provides utility functions to convert from and to any unicode and ASCII encoding.

Definition at line 44 of file Unicode.hpp.
### Member Typedef Documentation

<table>
<thead>
<tr>
<th>typedef std::basic_string&lt;Uint8&gt;</th>
<th>sf::Unicode::UTF8String</th>
</tr>
</thead>
</table>

Define a string type for each encoding. Warning: in UTF8 and UTF16 strings, one element doesn't necessarily maps to one character; only an UTF32 element is wide enough to hold all possible unicode values.

Definition at line 54 of file Unicode.hpp.
Member Function Documentation

```cpp
template<typename In , typename Out >
Out sf::Unicode::ANSIToUTF32 ( In Begin , In End , Out Output , const std::locale & Locale )
```

Generic function to convert an ANSI characters range to an UTF-32 characters range, using the given locale.

Definition at line 67 of file Unicode.inl.

```cpp
template<typename In , typename Out >
static Out sf::Unicode::ANSIToUTF32 ( In Begin , In End , Out Output , const std::locale & Locale = GetDefaultLocale() )
```

Generic function to convert an ANSI characters range to an UTF-32 characters range, using the given locale.

**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 (uses the current one by default)

**Returns:**
- Iterator to the end of the output sequence which has been written
Get the number of characters composing an UTF-16 string.

Definition at line 442 of file Unicode.inl.

Get the number of characters composing an UTF-16 string.

Parameters:
  begin : Iterator pointing to the beginning of the input sequence
  end : Iterator pointing to the end of the input sequence

Returns:
  Count of the characters in the string

Get the number of characters composing an UTF-32 string.

Definition at line 471 of file Unicode.inl.
Get the number of characters composing an UTF-32 string.

**Parameters:**
- *Begin*: Iterator pointing to the beginning of the input sequence
- *End*: Iterator pointing to the end of the input sequence

**Returns:**
Count of the characters in the string

Get the number of characters composing an UTF-8 string.

Definition at line 422 of file `Unicode.inl`.

Get the number of characters composing an UTF-8 string.

**Parameters:**
- *Begin*: Iterator pointing to the beginning of the input sequence
- *End*: Iterator pointing to the end of the input sequence

**Returns:**
Count of the characters in the string

```
template<typename In , typename Out >
Out sf::Unicode::UTF16ToUTF32 ( In Begin, In End, Out Output,
Uint32 Replacement ) [inline]
```

Generic function to convert an UTF-16 characters range to an UTF-32 characters range, using the given locale.

Definition at line 280 of file Unicode.inl.

```
template<typename In , typename Out >
static Out sf::Unicode::UTF16ToUTF32 ( In Begin, In End, Out Output,
Uint32 Replacement = '?') [inline, static]
```

Generic function to convert an UTF-16 characters range to an UTF-32 characters range, using the given locale.

**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 character for characters not convertible to
  output encoding ("?" by default -- use 0 to use no replacement character)

**Returns:**
Iterator to the end of the output sequence which has been
template<typename In , typename Out >
Out sf::Unicode::UTF16ToUTF8 ( In Begin,
In End,
Out Output,
Uint8 Replacement 
 ) [inline]

Generic function to convert an UTF-16 characters range to an UTF-8 characters range, using the given locale.

Definition at line 207 of file Unicode.inl.

template<typename In , typename Out >
static Out sf::Unicode::UTF16ToUTF8 ( In Begin,
In End,
Out Output,
Uint8 Replacement = '?'
 ) [inline, static]

Generic function to convert an UTF-16 characters range to an UTF-8 characters range, using the given locale.

**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 character for characters not convertible to output encoding ("?" by default -- use 0 to use no replacement character)

**Returns:**
Iterator to the end of the output sequence which has been
template<typename In , typename Out >
Out sf::Unicode::UTF32ToANSI ( In Begin,
In End,
Out Output,
char Replacement,
const std::locale & Locale )

[inline]

Generic function to convert an UTF-32 characters range to an ANSI characters range, using the given locale.

Definition at line 31 of file Unicode.inl.

template<typename In , typename Out >
static Out sf::Unicode::UTF32ToANSI ( In Begin,
In End,
Out Output,
char Replacement = '?',
const std::locale & Locale = GetDefaultLocale() )

Generic function to convert an UTF-32 characters range to an ANSI characters range, using the given locale.

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 character for characters not convertible to output encoding ('?' by default -- use 0 to use no replacement character)
  * **Locale**: Locale to use for conversion (uses the current one by default)
Returns:
Iterator to the end of the output sequence which has been written

Generic function to convert an UTF-32 characters range to an UTF-16 characters range, using the given locale.

Definition at line 379 of file Unicode.inl.

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 character for characters not convertible to
Replacement output encoding ("?" by default -- use 0 to use no replacement character)

Returns:
Iterator to the end of the output sequence which has been written

template<typename In , typename Out >
Out sf::Unicode::UTF32ToUTF8 ( In Begin,
In End,
Out Output,
Uint8 Replacement )

[inline]

Generic function to convert an UTF-32 characters range to an UTF-8 characters range, using the given locale.

Definition at line 326 of file Unicode.inl.

template<typename In , typename Out >
static Out sf::Unicode::UTF32ToUTF8 ( In Begin,
In End,
Out Output,
Uint8 Replacement = '?'
)

[inline, static]

Generic function to convert an UTF-32 characters range to an UTF-8 characters range, using the given locale.

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 character for characters not convertible to
**Replacement** output encoding (‘?’ by default -- use 0 to use no replacement character)

**Returns:**
Iterator to the end of the output sequence which has been written

```cpp
template<typename In , typename Out >
Out sf::Unicode::UTF8ToUTF16 ( In Begin, In End, Out Output, Uint16 Replacement )
```

Generic function to convert an UTF-8 characters range to an UTF-16 characters range, using the given locale.

Definition at line 102 of file Unicode.inl.

```cpp
template<typename In , typename Out >
static Out sf::Unicode::UTF8ToUTF16 ( In Begin, In End, Out Output, Uint16 Replacement = '?' )
```

Generic function to convert an UTF-8 characters range to an UTF-16 characters range, using the given locale.

**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 character for characters not convertible to
Replacement output encoding ('?' by default -- use 0 to use no replacement character)

Returns:
Iterator to the end of the output sequence which has been written

```plaintext
template<typename In , typename Out >
Out sf::Unicode::UTF8ToUTF32 ( In Begin, In End, Out Output, Uint32 Replacement )
```

Generic function to convert an UTF-8 characters range to an UTF-32 characters range, using the given locale.

Definition at line 163 of file Unicode.inl.

```plaintext
template<typename In , typename Out >
static Out sf::Unicode::UTF8ToUTF32 ( In Begin, In End, Out Output, Uint32 Replacement = '?' )
```

Generic function to convert an UTF-8 characters range to an UTF-32 characters range, using the given locale.

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 character for characters not convertible to
Replacement output encoding (‘?’ by default -- use 0 to use no replacement character)

Returns:
Iterator to the end of the output sequence which has been written

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

- Unicode.hpp
- Unicode.inl
- Unicode.cpp

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
sf::Unicode::Unicode::Text Class Reference

This class is an abstract definition of a unicode text, it can be converted from and to any kind of string and encoding. More...

#include <Unicode.hpp>

List of all members.
Public Member Functions

- **Text ()**
  Default constructor (empty text).

- **Text (const char *Str)**
  Construct the unicode text from any type of string.

- **Text (const wchar_t *Str)**

- **Text (const Uint8 *Str)**

- **Text (const Uint16 *Str)**

- **Text (const Uint32 *Str)**

- **Text (const std::string &Str)**

- **Text (const std::wstring &Str)**

- **Text (const Unicode::UTF8String &Str)**

- **Text (const Unicode::UTF16String &Str)**

- **Text (const Unicode::UTF32String &Str)**

- **operator std::string () const**
  Operator to cast the text to any type of string.

- **operator std::wstring () const**

- **operator Unicode::UTF8String () const**

- **operator Unicode::UTF16String () const**

- **operator const Unicode::UTF32String & () const**
Detailed Description

This class is an abstract definition of a unicode text, it can be converted from and to any kind of string and encoding.

Definition at line 63 of file Unicode.hpp.
## Constructor & Destructor Documentation

**sf::Unicode::Unicode::Text::Text ( )**

Default constructor (empty text).

Definition at line 105 of file `Unicode.cpp`.

**sf::Unicode::Unicode::Text::Text ( const char * Str )**

Construct the unicode text from any type of string.

**Parameters:**

- *Str* : String to convert

Definition at line 114 of file `Unicode.cpp`.
sf::Unicode::Unicode::Text::operator std::string ( ) const

Operator to cast the text to any type of string.

Returns:
   Converted string

Definition at line 227 of file Unicode.cpp.

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

- Unicode.hpp
- Unicode.cpp
sf::Vector2
**sf::Vector2< T > Class Template Reference**

**Vector2** is an utility class for manipulating 2 dimensional vectors. [More...](#)

#include `<Vector2.hpp>`

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Vector2 ()</strong></td>
<td>Default constructor.</td>
</tr>
<tr>
<td><strong>Vector2 (T X, T Y)</strong></td>
<td>Construct the vector from its coordinates.</td>
</tr>
</tbody>
</table>
## Public Attributes

<table>
<thead>
<tr>
<th>T</th>
<th>x</th>
<th>X coordinate of the vector.</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>y</td>
<td>Y coordinate of the vector.</td>
</tr>
</tbody>
</table>
Detailed Description

template<typename T>
class sf::Vector2< T >

**Vector2** is an utility class for manipulating 2 dimensional vectors.

Template parameter defines the type of coordinates (integer, float, ...)

Definition at line 37 of file Vector2.hpp.
### Constructor & Destructor Documentation

**template<typename T>**

```cpp
ts::Vector2<T>::Vector2()
```

Default constructor.

**template<typename T>**

```cpp
ts::Vector2<T>::Vector2(T X, T Y)
```

Construct the vector from its coordinates.

**Parameters:**
- **X**: X coordinate
- **Y**: Y coordinate
### Member Data Documentation

<table>
<thead>
<tr>
<th>Template&lt;typename T&gt;</th>
<th>T sf::Vector2&lt;T&gt;::x</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X coordinate of the vector.</td>
</tr>
<tr>
<td></td>
<td>Definition at line 59 of file Vector2.hpp.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Template&lt;typename T&gt;</th>
<th>T sf::Vector2&lt;T&gt;::y</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Y coordinate of the vector.</td>
</tr>
<tr>
<td></td>
<td>Definition at line 60 of file Vector2.hpp.</td>
</tr>
</tbody>
</table>

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

- **Vector2.hpp**

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
sf::Vector3
sf::Vector3< T > Class Template Reference

**Vector3** is an utility class for manipulating 3 dimensional vectors. [More...](#)

```
#include <Vector3.hpp>
```

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vector3 ()</td>
<td>Default constructor.</td>
</tr>
<tr>
<td>Vector3 (T X, T Y, T Z)</td>
<td>Construct the vector from its coordinates.</td>
</tr>
</tbody>
</table>
### Public Attributes

<table>
<thead>
<tr>
<th>T</th>
<th>x</th>
<th>X coordinate of the vector.</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>y</td>
<td>Y coordinate of the vector.</td>
</tr>
<tr>
<td>T</td>
<td>z</td>
<td>Z coordinate of the vector.</td>
</tr>
</tbody>
</table>
Detailed Description

template<typename T>
class sf::Vector3< T >

**Vector3** is an utility class for manipulating 3 dimensional vectors.

Template parameter defines the type of coordinates (integer, float, ...)

Definition at line 37 of file Vector3.hpp.
 Constructor & Destructor Documentation

**Default constructor.**

```
template<typename T>
sf::Vector3 T>::Vector3 ( )
```

Construct the vector from its coordinates.

**Parameters:**
- $X$: $X$ coordinate
- $Y$: $Y$ coordinate
- $Z$: $Z$ coordinate
## Member Data Documentation

```cpp
template<typename T>
T sf::Vector3<T>::x
```

X coordinate of the vector.

Definition at line 60 of file `Vector3.hpp`.

```cpp
template<typename T>
T sf::Vector3<T>::y
```

Y coordinate of the vector.

Definition at line 61 of file `Vector3.hpp`.

```cpp
template<typename T>
T sf::Vector3<T>::z
```

Z coordinate of the vector.

Definition at line 62 of file `Vector3.hpp`.

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

- `Vector3.hpp`

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
sf::VideoMode
sf::VideoMode Class Reference

**VideoMode** defines a video mode (width, height, bpp, frequency) and provides static functions for getting modes supported by the display device. [More...](#)

```cpp
#include <VideoMode.hpp>
```

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>VideoMode()</code></td>
<td>Default constructor.</td>
</tr>
<tr>
<td><code>VideoMode(unsigned int ModeWidth, unsigned int ModeHeight, unsigned int ModeBpp=32)</code></td>
<td>Construct the video mode with its attributes.</td>
</tr>
<tr>
<td><code>bool IsValid()</code></td>
<td>Tell whether or not the video mode is supported.</td>
</tr>
<tr>
<td><code>bool operator==(const VideoMode &amp;Other)</code></td>
<td>Comparison operator overload -- tell if two video modes are equal.</td>
</tr>
<tr>
<td><code>bool operator!=(const VideoMode &amp;Other)</code></td>
<td>Comparison operator overload -- tell if two video modes are different.</td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static VideoMode</td>
<td>GetDesktopMode ()</td>
<td>Get the current desktop video mode.</td>
</tr>
<tr>
<td>static VideoMode</td>
<td>GetMode (std::size_t Index)</td>
<td>Get a valid video mode. Index must be in range [0, GetModesCount()]; Modes are sorted from best to worst.</td>
</tr>
<tr>
<td>static std::size_t</td>
<td>GetModesCount ()</td>
<td>Get valid video modes count.</td>
</tr>
</tbody>
</table>
## Public Attributes

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unsigned int</td>
<td><strong>Width</strong></td>
<td>Video mode width, in pixels.</td>
</tr>
<tr>
<td>unsigned int</td>
<td><strong>Height</strong></td>
<td>Video mode height, in pixels.</td>
</tr>
<tr>
<td>unsigned int</td>
<td><strong>BitsPerPixel</strong></td>
<td>Video mode pixel depth, in bits per pixels.</td>
</tr>
</tbody>
</table>
Detailed Description

**VideoMode** defines a video mode (width, height, bpp, frequency) and provides static functions for getting modes supported by the display device.

Definition at line 42 of file **VideoMode.hpp**.
### Constructor & Destructor Documentation

<table>
<thead>
<tr>
<th>sf::VideoMode::VideoMode ( )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default constructor.</td>
</tr>
<tr>
<td>Definition at line 67 of file VideoMode.cpp.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sf::VideoMode::VideoMode ( unsigned int ModeWidth, unsigned int ModeHeight, unsigned int ModeBpp = 32 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Construct the video mode with its attributes.</td>
</tr>
<tr>
<td><strong>Parameters:</strong></td>
</tr>
<tr>
<td>* ModeWidth : Width in pixels</td>
</tr>
<tr>
<td>* ModeHeight : Height in pixels</td>
</tr>
<tr>
<td>* ModeBpp : Pixel depths in bits per pixel (32 by default)</td>
</tr>
<tr>
<td>Definition at line 79 of file VideoMode.cpp.</td>
</tr>
</tbody>
</table>
### Member Function Documentation

<table>
<thead>
<tr>
<th>VideoMode sf::VideoMode::GetDesktopMode( ) [static]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get the current desktop video mode.</td>
</tr>
<tr>
<td><strong>Returns:</strong></td>
</tr>
<tr>
<td>Current desktop video mode</td>
</tr>
<tr>
<td>Definition at line 91 of file VideoMode.cpp.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VideoMode sf::VideoMode::GetMode( std::size_t Index ) [static]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get a valid video mode Index must be in range [0, GetModesCount()] Modes are sorted from best to worst.</td>
</tr>
<tr>
<td>Get a valid video mode Index must be in range [0, GetModesCount()].</td>
</tr>
<tr>
<td><strong>Parameters:</strong></td>
</tr>
<tr>
<td>Index : Index of video mode to get</td>
</tr>
<tr>
<td><strong>Returns:</strong></td>
</tr>
<tr>
<td>Corresponding video mode (invalid mode if index is out of range)</td>
</tr>
<tr>
<td>Definition at line 102 of file VideoMode.cpp.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>std::size_t sf::VideoMode::GetModesCount( ) [static]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get valid video modes count.</td>
</tr>
</tbody>
</table>
Returns:
        Number of valid video modes available

Definition at line 117 of file VideoMode.cpp.

```cpp
bool sf::VideoMode::IsValid() const
```

Tell whether or not the video mode is supported.

Returns:
        True if video mode is supported, false otherwise

Definition at line 129 of file VideoMode.cpp.

```cpp
bool sf::VideoMode::operator!=(const VideoMode & Other) const
```

Comparison operator overload -- tell if two video modes are different.

**Parameters:**
        Other : Video mode to compare

**Returns:**
        True if modes are different

Definition at line 152 of file VideoMode.cpp.

```cpp
bool sf::VideoMode::operator==(const VideoMode & Other) const
```

Comparison operator overload -- tell if two video modes are equal.
Parameters:
   Other : Video mode to compare

Returns:
   True if modes are equal

Definition at line 141 of file VideoMode.cpp.
# Member Data Documentation

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
<th>Definition Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>unsigned int sf::VideoMode::BitsPerPixel</td>
<td>Video mode pixel depth, in bits per pixels.</td>
<td>Definition at line 123 of file VideoMode.hpp.</td>
</tr>
<tr>
<td>unsigned int sf::VideoMode::Height</td>
<td>Video mode height, in pixels.</td>
<td>Definition at line 122 of file VideoMode.hpp.</td>
</tr>
<tr>
<td>unsigned int sf::VideoMode::Width</td>
<td>Video mode width, in pixels.</td>
<td>Definition at line 121 of file VideoMode.hpp.</td>
</tr>
</tbody>
</table>

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

- VideoMode.hpp
- VideoMode.cpp

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
sf::View
sf::View Class Reference

This class defines a view (position, size, etc. More...

#include <View.hpp>

List of all members.
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>View</strong> (const FloatRect &amp;ViewRect=FloatRect(0, 0, 1000, 1000))</td>
<td>Construct the view from a rectangle.</td>
</tr>
<tr>
<td><strong>View</strong> (const sf::Vector2f &amp;Center, const sf::Vector2f &amp;HalfSize)</td>
<td>Construct the view from its center and half-size.</td>
</tr>
<tr>
<td>void SetCenter (float X, float Y)</td>
<td>Change the center of the view (take 2 values).</td>
</tr>
<tr>
<td>void SetCenter (const sf::Vector2f &amp;Center)</td>
<td>Change the center of the view (take a vector).</td>
</tr>
<tr>
<td>void SetHalfSize (float HalfWidth, float HalfHeight)</td>
<td>Change the half-size of the view (take 2 values).</td>
</tr>
<tr>
<td>void SetHalfSize (const sf::Vector2f &amp;HalfSize)</td>
<td>Change the half-size of the view (take a vector).</td>
</tr>
<tr>
<td>void SetFromRect (const FloatRect &amp;ViewRect)</td>
<td>Rebuild the view from a rectangle.</td>
</tr>
<tr>
<td>const sf::Vector2f &amp; GetCenter () const</td>
<td>Get the center of the view.</td>
</tr>
<tr>
<td>const sf::Vector2f &amp; GetHalfSize () const</td>
<td>Get the half-size of the view.</td>
</tr>
<tr>
<td>const sf::FloatRect &amp; GetRect () const</td>
<td>Get the bounding rectangle of the view.</td>
</tr>
<tr>
<td>void Move (float OffsetX, float OffsetY)</td>
<td>Move the view (take 2 values).</td>
</tr>
<tr>
<td>void Move (const sf::Vector2f &amp;Offset)</td>
<td>Move the view (take a vector).</td>
</tr>
<tr>
<td>void Zoom (float Factor)</td>
<td>Resize the view rectangle to simulate a zoom / unzoom effect.</td>
</tr>
<tr>
<td>class</td>
<td>RenderTarget</td>
</tr>
</tbody>
</table>
Detailed Description

This class defines a view (position, size, etc.) ; you can consider it as a 2D camera

Definition at line 45 of file View.hpp.
sf::View::View ( const FloatRect & ViewRect = FloatRect(0, 0, 1000, 1000) )

Construct the view from a rectangle.

**Parameters:**
- ViewRect: Rectangle defining the position and size of the view (1000x1000 by default)

Definition at line 37 of file View.cpp.

sf::View::View ( const sf::Vector2f & Center,
    const sf::Vector2f & HalfSize
)

Construct the view from its center and half-size.

**Parameters:**
- Center: Center of the view
- HalfSize: Half-size of the view (from center to corner)

Definition at line 46 of file View.cpp.
### Member Function Documentation

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
<th>Returns</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sf::Vector2f &amp; sf::View::GetCenter()</code></td>
<td>Get the center of the view.</td>
<td>Center of the view</td>
<td>Line 108 of <code>View.cpp</code></td>
</tr>
<tr>
<td><code>sf::Vector2f &amp; sf::View::GetHalfSize()</code></td>
<td>Get the half-size of the view.</td>
<td>Half-size of the view</td>
<td>Line 117 of <code>View.cpp</code></td>
</tr>
<tr>
<td><code>sf::FloatRect &amp; sf::View::GetRect()</code></td>
<td>Get the bounding rectangle of the view.</td>
<td>Bounding rectangle of the view</td>
<td>Line 126 of <code>View.cpp</code></td>
</tr>
<tr>
<td><code>void sf::View::Move(const sf::Vector2f &amp; Offset)</code></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```cpp
const sf::Vector2f & sf::View::GetCenter () const
Get the center of the view.

Returns:
    Center of the view

Definition at line 108 of file View.cpp.

const sf::Vector2f & sf::View::GetHalfSize () const
Get the half-size of the view.

Returns:
    Half-size of the view

Definition at line 117 of file View.cpp.

const sf::FloatRect & sf::View::GetRect () const
Get the bounding rectangle of the view.

Returns:
    Bounding rectangle of the view

Definition at line 126 of file View.cpp.
```
Move the view (take a vector).

**Parameters:**
- \( \text{Offset} \) : Offset to move the view

Definition at line 150 of file `View.cpp`.

```cpp
void sf::View::Move(float \text{OffsetX},
                     float \text{OffsetY})
```

Move the view (take 2 values).

**Parameters:**
- \( \text{OffsetX} \) : Offset to move the view, on X axis
- \( \text{OffsetY} \) : Offset to move the view, on Y axis

Definition at line 139 of file `View.cpp`.

```cpp
void sf::View::SetCenter(const sf::Vector2f & \text{Center})
```

Change the center of the view (take a vector).

**Parameters:**
- \( \text{Center} \) : New center

Definition at line 69 of file `View.cpp`.

```cpp
void sf::View::SetCenter(float \text{X},
                          float \text{Y})
```
Change the center of the view (take 2 values).

**Parameters:**

- \( X \) : X coordinate of the new center
- \( Y \) : Y coordinate of the new center

Definition at line 58 of file View.cpp.

```cpp
def void sf::View::SetFromRect (const FloatRect & ViewRect)
```

Rebuild the view from a rectangle.

**Parameters:**

- `ViewRect` : Rectangle defining the position and size of the view

Definition at line 98 of file View.cpp.

```cpp
def void sf::View::SetHalfSize (const sf::Vector2f & HalfSize)
```

Change the half-size of the view (take a vector).

**Parameters:**

- `HalfSize` : New half-size

Definition at line 89 of file View.cpp.

```cpp
def void sf::View::SetHalfSize (float HalfWidth,
                                float HalfHeight)
```

Change the half-size of the view (take 2 values).
Parameters:

- **HalfWidth**: New half-width
- **HalfHeight**: New half-height

Definition at line 78 of file View.cpp.

```cpp
void sf::View::Zoom ( float Factor )
```

Resize the view rectangle to simulate a zoom / unzoom effect.

Parameters:

- **Factor**: Zoom factor to apply, relative to the current zoom

Definition at line 159 of file View.cpp.

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

- **View.hpp**
- **View.cpp**

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
sf::Window
sf::Window Class Reference

**Window** is a rendering window; it can create a new window or connect to an existing one. [More...](#)

```cpp
#include <Window.hpp>
```

Inheritance diagram for sf::Window:

[Diagram showing the inheritance structure of sf::Window]

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Window ()</strong></td>
<td>Default constructor.</td>
</tr>
<tr>
<td>**Window (VideoMode Mode, const std::string &amp;Title, unsigned long WindowStyle=Style::Resize</td>
<td>Style::Close, const WindowSettings &amp;Params=WindowSettings())**</td>
</tr>
<tr>
<td><strong>Window (WindowHandle Handle, const WindowSettings &amp;Params=WindowSettings())</strong></td>
<td>Construct the window from an existing control.</td>
</tr>
<tr>
<td><strong>virtual ~Window ()</strong></td>
<td>Destructor.</td>
</tr>
<tr>
<td>**void Create (VideoMode Mode, const std::string &amp;Title, unsigned long WindowStyle=Style::Resize</td>
<td>Style::Close, const WindowSettings &amp;Params=WindowSettings())**</td>
</tr>
<tr>
<td><strong>void Create (WindowHandle Handle, const WindowSettings &amp;Params=WindowSettings())</strong></td>
<td>Create (or recreate) the window from an existing control.</td>
</tr>
<tr>
<td><strong>void Close ()</strong></td>
<td>Close (destroy) the window.</td>
</tr>
<tr>
<td><strong>bool IsOpened () const</strong></td>
<td>Tell whether or not the window is opened (ie.</td>
</tr>
<tr>
<td><strong>unsigned int GetWidth () const</strong></td>
<td>Get the width of the rendering region of the window.</td>
</tr>
<tr>
<td><strong>unsigned int GetHeight () const</strong></td>
<td>Get the height of the rendering region of the window.</td>
</tr>
<tr>
<td><strong>const WindowSettings &amp; GetSettings () const</strong></td>
<td>Get the creation settings of the window.</td>
</tr>
<tr>
<td><strong>bool GetEvent (Event &amp;EventReceived)</strong></td>
<td>Get the event on top of events stack, if any, and pop it.</td>
</tr>
<tr>
<td><strong>void UseVerticalSync (bool Enabled)</strong></td>
<td>Enable / disable vertical synchronization.</td>
</tr>
<tr>
<td><strong>void ShowMouseCursor (bool Show)</strong></td>
<td>Show or hide the mouse cursor.</td>
</tr>
<tr>
<td><strong>void SetCursorPosition (unsigned int Left, unsigned int Top)</strong></td>
<td>Change the position of the mouse cursor.</td>
</tr>
<tr>
<td><strong>void SetPosition (int Left, int Top)</strong></td>
<td>Change the position of the window on screen.</td>
</tr>
<tr>
<td><strong>void SetSize (unsigned int Width, unsigned int Height)</strong></td>
<td>Change the size of the rendering region of the window.</td>
</tr>
<tr>
<td><strong>void Show (bool State)</strong></td>
<td>Show or hide the window.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>void EnableKeyRepeat(bool Enabled)</code></td>
<td>Enable or disable automatic key-repeat.</td>
</tr>
<tr>
<td><code>void SetIcon(unsigned int Width, unsigned int Height, const Uint8* Pixels)</code></td>
<td>Change the window's icon.</td>
</tr>
<tr>
<td><code>bool SetActive(bool Active=true) const</code></td>
<td>Activate or deactivate the window as the current target for rendering.</td>
</tr>
<tr>
<td><code>void Display()</code></td>
<td>Display the window on screen.</td>
</tr>
<tr>
<td><code>const Input&amp; GetInput() const</code></td>
<td>Get the input manager of the window.</td>
</tr>
<tr>
<td><code>void SetFramerateLimit(unsigned int Limit)</code></td>
<td>Limit the framerate to a maximum fixed frequency.</td>
</tr>
<tr>
<td><code>float GetFrameTime() const</code></td>
<td>Get time elapsed since last frame.</td>
</tr>
<tr>
<td><code>void SetJoystickThreshold(float Threshold)</code></td>
<td>Change the joystick threshold, ie.</td>
</tr>
</tbody>
</table>
Detailed Description

Window is a rendering window; it can create a new window or connect to an existing one.

Definition at line 55 of file Window/Window.hpp.
Constructor & Destructor Documentation

**sf::Window::Window ( )**

Default constructor.

Definition at line 49 of file `Window.cpp`.

**sf::Window::Window ( VideoMode Mode, const std::string & Title, unsigned long WindowStyle = Style::Resize | Style::Close const WindowSettings & Params = WindowSettings() )**

Construct a new window.

### Parameters:
- **Mode**: Video mode to use
- **Title**: Title of the window
- **WindowStyle**: `Window` style, see `sf::Style` (Resize | Close by default)
- **Params**: Creation parameters (see default constructor for default values)

Definition at line 64 of file `Window.cpp`.

**sf::Window::Window ( WindowHandle Handle, const WindowSettings & Params = WindowSettings() )**

Construct the window from an existing control.

### Parameters:
**Handle**  : Platform-specific handle of the control

**Params**  : Creation parameters (see default constructor for default values)

Definition at line 79 of file `Window.cpp`.

```
sf::Window::~Window( ) [virtual]
```

Destructor.

Definition at line 94 of file `Window.cpp`. 
**Member Function Documentation**

### void sf::Window::Close ( )

Close (destroy) the window.

The *sf::Window* instance remains valid and you can call Create to recreate the window.

The *sf::Window* instance remains valid and you can call Create to recreate the window.

Definition at line 165 of file *Window.cpp*.

### void sf::Window::Create ( WindowHandle Handle,
const WindowSettings & Params = WindowSettings )

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

Create the window from an existing control.

**Parameters:**
- *Handle* : Platform-specific handle of the control
- *Params* : Creation parameters (see default constructor for default values)

Definition at line 147 of file *Window.cpp*.

### void sf::Window::Create ( VideoMode Mode,
const std::string & Title,
unsigned long WindowStyle = Style::Resize | Style::Close,
const WindowSettings & Params = WindowSettings )
Create (or recreate) the window.

Create the window.

**Parameters:**

- **Mode**: Video mode to use
- **Title**: Title of the window
- **WindowStyle**: Window style, see `sf::Style` (Resize | Close by default)
- **Params**: Creation parameters (see default constructor for default values)

Definition at line **104** of file `Window.cpp`.

```cpp
void sf::Window::Display (
)
```

Display the window on screen.

Definition at line **353** of file `Window.cpp`.

```cpp
void sf::Window::EnableKeyRepeat ( bool Enabled )
```

Enable or disable automatic key-repeat.

Automatic key-repeat is enabled by default

**Parameters:**

- **Enabled**: True to enable, false to disable

Automatic key-repeat is enabled by default

Definition at line **317** of file `Window.cpp`.
### sf::Window::GetEvent (Event & EventReceived)

Get the event on top of events stack, if any, and pop it.

Get the event on top of events stack, if any.

**Parameters:**

* EventReceived : Event to fill, if any

**Returns:**

True if an event was returned, false if events stack was empty

Definition at line 218 of file `Window.cpp`.

### sf::Window::GetFrameTime () const

Get time elapsed since last frame.

**Returns:**

Time elapsed, in seconds

Definition at line 394 of file `Window.cpp`.

### sf::Window::GetHeight () const

Get the height of the rendering region of the window.

**Returns:**

Height in pixels

Reimplemented in `sf::RenderWindow`.
**const Input & sf::Window::GetInput() const**

Get the input manager of the window.

**Returns:**
Reference to the input

Definition at line 200 of file `Window.cpp`.

**const WindowSettings & sf::Window::GetSettings() const**

Get the creation settings of the window.

**Returns:**
Structure containing the creation settings

Definition at line 376 of file `Window.cpp`.

**unsigned int sf::Window::GetWidth() const**

Get the width of the rendering region of the window.

**Returns:**
Width in pixels

Reimplemented in `sf::RenderWindow`.

Definition at line 209 of file `Window.cpp`.

**bool sf::Window::IsOpened() const**
Tell whether or not the window is opened (ie.
has been created). Note that a hidden window (Show(false))
will still return true

**Returns:**
True if the window is opened

has been created). Note that a hidden window (Show(false))
will still return true

Definition at line 182 of file Window.cpp.

```cpp
bool sf::Window::SetActive ( bool Active = true ) const
```

Activate of deactivate the window as the current target for
rendering.

Activate of deactivate the window as the current target for
rendering.

**Parameters:**

*Active* : True to activate, false to deactivate (true by default)

**Returns:**
True if operation was successful, false otherwise

Definition at line 338 of file Window.cpp.

```cpp
void sf::Window::SetCursorPosition ( unsigned int Left, 
unigned int Top )
```
Change the position of the mouse cursor.

**Parameters:**
- *Left*: Left coordinate of the cursor, relative to the window
- *Top*: Top coordinate of the cursor, relative to the window

Definition at line **260** of file *Window.cpp*.

```cpp
void sf::Window::SetFramerateLimit (unsigned int Limit)
```

Limit the framerate to a maximum fixed frequency.

Set the framerate at a fixed frequency.

**Parameters:**
- *Limit*: Framerate limit, in frames per seconds (use 0 to disable limit)

Definition at line **385** of file *Window.cpp*.

```cpp
void sf::Window::SetIcon (unsigned int Width,
                          unsigned int Height,
                          const Uint8 *Pixels)
```

Change the window's icon.

**Parameters:**
- *Width*: Icon's width, in pixels
- *Height*: Icon's height, in pixels
- *Pixels*: Pointer to the pixels in memory, format must be RGBA 32 bits

Definition at line **327** of file *Window.cpp*. 
void sf::Window::SetJoystickThreshold ( float Threshold )

Change the joystick threshold, ie.
the value below which no move event will be generated

Parameters:
  Threshold : New threshold, in range [0, 100]
the value below which no move event will be generated
Definition at line 404 of file Window.cpp.

void sf::Window::SetPosition ( int Left, int Top )

Change the position of the window on screen.
Only works for top-level windows

Parameters:
  Left : Left position
  Top : Top position
Definition at line 276 of file Window.cpp.

void sf::Window::SetSize ( unsigned int Width, unsigned int Height )

Change the size of the rendering region of the window.
**Parameters:**

- *Width*: New width
- *Height*: New height

Definition at line 293 of file `Window.cpp`.

```cpp
void sf::Window::Show ( bool State )
```

Show or hide the window.

**Parameters:**

- *State*: True to show, false to hide

Definition at line 303 of file `Window.cpp`.

```cpp
void sf::Window::ShowMouseCursor ( bool Show )
```

Show or hide the mouse cursor.

**Parameters:**

- *Show*: True to show, false to hide

Definition at line 250 of file `Window.cpp`.

```cpp
void sf::Window::UseVerticalSync ( bool Enabled )
```

Enable / disable vertical synchronization.

**Parameters:**

- *Enabled*: True to enable v-sync, false to deactivate

Definition at line 240 of file `Window.cpp`. 
The documentation for this class was generated from the following files:

- Window/Window.hpp
- Window.cpp
sf::WindowListener
sf::WindowListener Class Reference

Base class for classes that want to receive events from a window (for internal use only). More...

#include <WindowListener.hpp>

Inheritance diagram for sf::WindowListener:

List of all members.
## Public Member Functions

<table>
<thead>
<tr>
<th>virtual void</th>
<th><code>OnEvent</code> (const <code>Event</code> &amp;EventReceived)=0</th>
</tr>
</thead>
</table>

*Called each time an event is received from attached window.*
Protected Member Functions

virtual ~WindowListener ()

Destructor.
Detailed Description

Base class for classes that want to receive events from a window (for internal use only).

Definition at line 42 of file WindowListener.hpp.
## Constructor & Destructor Documentation

**virtual sf::WindowListener::~WindowListener ( )** [inline, protected, virtual]

Destructor.

Definition at line **60** of file `WindowListener.hpp`. 

<table>
<thead>
<tr>
<th>Constructor &amp; Destructor Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>virtual sf::WindowListener::~WindowListener ( )</strong> [inline, protected, virtual]</td>
</tr>
<tr>
<td>Destructor.</td>
</tr>
<tr>
<td>Definition at line <strong>60</strong> of file <code>WindowListener.hpp</code>.</td>
</tr>
</tbody>
</table>
virtual void sf::WindowListener::OnEvent ( const Event & EventReceived )

Called each time an event is received from attached window.

Parameters:

EventReceived : Event received

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

- **WindowListener.hpp**
sf::WindowSettings
sf::WindowSettings Struct Reference

Structure defining the creation settings of windows. More...

#include <WindowSettings.hpp>

List of all members.
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>WindowSettings</code></td>
<td>(unsigned int Depth=24, unsigned int Stencil=8, unsigned int Antialiasing=0)</td>
</tr>
</tbody>
</table>

*Default constructor.*
## Public Attributes

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unsigned int</td>
<td>DepthBits</td>
<td>Bits of the depth buffer.</td>
</tr>
<tr>
<td>unsigned int</td>
<td>StencilBits</td>
<td>Bits of the stencil buffer.</td>
</tr>
<tr>
<td>unsigned int</td>
<td>AntialiasingLevel</td>
<td>Level of antialiasing.</td>
</tr>
</tbody>
</table>
Detailed Description

Structure defining the creation settings of windows.

Definition at line 34 of file WindowSettings.hpp.
Constructor & Destructor Documentation

sf::WindowSettings::WindowSettings ( unsigned int \textit{Depth} = 24, 
    unsigned int \textit{Stencil} = 8, 
    unsigned int \textit{Antialiasing} = 0
)

Default constructor.

\textbf{Parameters:}

\begin{itemize}
    \item \textit{Depth} : Depth buffer bits (24 by default)
    \item \textit{Stencil} : Stencil buffer bits (8 by default)
    \item \textit{Antialiasing} : Antialiasing level (0 by default)
\end{itemize}

Definition at line \textbf{44} of file \texttt{WindowSettings.hpp}.
Member Data Documentation

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Definition Location</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>unsigned int sf::WindowSettings::AntialiasingLevel</code></td>
<td>Level of antialiasing.</td>
<td>Line 56 of WindowSettings.hpp</td>
</tr>
<tr>
<td><code>unsigned int sf::WindowSettings::DepthBits</code></td>
<td>Bits of the depth buffer.</td>
<td>Line 54 of WindowSettings.hpp</td>
</tr>
<tr>
<td><code>unsigned int sf::WindowSettings::StencilBits</code></td>
<td>Bits of the stencil buffer.</td>
<td>Line 55 of WindowSettings.hpp</td>
</tr>
</tbody>
</table>

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

- `WindowSettings.hpp`

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::

Documentation generated by doxygen 1.5.2 ::
Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

- sf::AudioResource
  - sf::Sound
    - sf::SoundStream
    - sf::Music
  - sf::SoundBuffer
- sf::Clock
- sf::Color
- sf::Drawable
  - sf::PostFX
  - sf::Shape
  - sf::Sprite
  - sf::String
- sf::Event
- sf::Event::Event::JoyButtonEvent
- sf::Event::Event::JoyMoveEvent
- sf::Event::Event::KeyEvent
- sf::Event::Event::MouseButtonEvent
- sf::Event::Event::MouseMoveEvent
- sf::Event::Event::MouseWheelEvent
- sf::Event::Event::SizeEvent
- sf::Event::Event::TextEvent
- sf::Ftp::Ftp::Response
  - sf::Ftp::Ftp::DirectoryResponse
  - sf::Ftp::Ftp::ListingResponse
- sf::Glyph
- sf::Http::Http::Request
- sf::Http::Http::Response
- sf::IPAddress
- `sf::Listener`
- `sf::Matrix3`
- `sf::NonCopyable`
  - `sf::Context`
  - `sf::Ftp`
  - `sf::Http`
  - `sf::Input`
  - `sf::Lock`
  - `sf::Mutex`
  - `sf::Thread`
    - `sf::SoundRecorder`
      - `sf::SoundBufferRecorder`
    - `sf::SoundStream`
  - `sf::Window`
    - `sf::RenderWindow`
- `sf::Packet`
- `sf::Randomizer`
- `sf::Rect<T>`
- `sf::RenderTarget`
  - `sf::RenderWindow`
- `sf::Resource<T>`
- `sf::Resource<Font>`
  - `sf::Font`
- `sf::Resource<Image>`
  - `sf::Image`
- `sf::Resource<SoundBuffer>`
  - `sf::SoundBuffer`
- `sf::ResourcePtr<T>`
- `sf::SelectorBase`
  - `sf::Selector<Type>`
- `sf::SocketHelper`
- `sf::SocketTCP`
- `sf::SocketUDP`
- sf::SoundStream::SoundStream::Chunk
- sf::Unicode
- sf::Unicode::Unicode::Text
- sf::Vector2< T >
- sf::Vector3< T >
- sf::VideoMode
- sf::View
- sf::WindowListener
  - sf::Input
  - sf::Window
- sf::WindowSettings

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
Here is a list of all documented class members with links to the class documentation for each member:

- a -

- a : sf::Color
- Accept() : sf::SocketTCP
- Accepted : sf::Http::Http::Response
- Add() : sf::Selector< Type >, sf::SelectorBase
- AddPoint() : sf::Shape
- Advance : sf::Glyph
- ANSIToUTF32() : sf::Unicode
- AntialiasingLevel : sf::WindowSettings
- Append() : sf::Packet
- Ascii : sf::Ftp
- AudioResource() : sf::AudioResource
<table>
<thead>
<tr>
<th>Main Page</th>
<th>Namespaces</th>
<th>Classes</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Namespace List</strong></td>
<td><strong>Namespace Members</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Namespace List

Here is a list of all documented namespaces with brief descriptions:

<table>
<thead>
<tr>
<th>Namespace</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sf::Blend</td>
<td>Enumerate the blending modes for drawable objects</td>
</tr>
<tr>
<td>sf::Joy</td>
<td>Definition of joystick axis for joystick events</td>
</tr>
<tr>
<td>sf::Key</td>
<td>Definition of key codes for keyboard events</td>
</tr>
<tr>
<td>sf::Mouse</td>
<td>Definition of button codes for mouse events</td>
</tr>
<tr>
<td>sf::Style</td>
<td>Enumeration of window creation styles</td>
</tr>
</tbody>
</table>

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::

Documentation generated by doxygen 1.5.2 ::
sf::Blend
sf::Blend Namespace Reference

Enumerate the blending modes for drawable objects. More...
### Enumerations

<table>
<thead>
<tr>
<th></th>
<th>Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>enum</td>
<td>Alpha, Add, Multiply, None</td>
</tr>
</tbody>
</table>
Detailed Description

Enumerate the blending modes for drawable objects.
### Enumeration Type Documentation

<table>
<thead>
<tr>
<th>Enumerators</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alpha</td>
<td>Pixel = Src * a + Dest * (1 - a).</td>
</tr>
<tr>
<td>Add</td>
<td>Pixel = Src + Dest.</td>
</tr>
<tr>
<td>Multiply</td>
<td>Pixel = Src * Dest.</td>
</tr>
<tr>
<td>None</td>
<td>No blending.</td>
</tr>
</tbody>
</table>

Definition at line 45 of file `Drawable.hpp`.

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
sf::Joy Namespace Reference

Definition of joystick axis for joystick events. More...
## Enumerations

<table>
<thead>
<tr>
<th>enum</th>
<th>Axis { AxisX, AxisY, AxisZ, AxisR, AxisU, AxisV, AxisPOV, AxisCount }</th>
</tr>
</thead>
<tbody>
<tr>
<td>enum</td>
<td>{ Count = 4, ButtonCount = 32 } }</td>
</tr>
</tbody>
</table>
Detailed Description

Definition of joystick axis for joystick events.
# Enumeration Type Documentation

<table>
<thead>
<tr>
<th>anonymous enum</th>
</tr>
</thead>
</table>

**Enumerator:**

- **Count**: Total number of supported joysticks.
- **ButtonCount**: Total number of supported joystick buttons.

Definition at line 186 of file `Event.hpp`.

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::

Documentation generated by doxygen 1.5.2 ::
sf::Key
sf::Key Namespace Reference

Definition of key codes for keyboard events. More...
Enumerations

```c
enum Code {
    A = 'a',
    B = 'b',
    C = 'c',
    D = 'd',
    E = 'e',
    F = 'f',
    G = 'g',
    H = 'h',
    I = 'i',
    J = 'j',
    K = 'k',
    L = 'l',
    M = 'm',
    N = 'n',
    O = 'o',
    P = 'p',
    Q = 'q',
    R = 'r',
    S = 's',
    T = 't',
    U = 'u',
    V = 'v',
    W = 'w',
    X = 'x',
    Y = 'y',
    Z = 'z',
    Num0 = '0',
    Num1 = '1',
    Num2 = '2',
    Num3 = '3',
    Num4 = '4',
    Num5 = '5',
    Num6 = '6',
    Num7 = '7',
    Num8 = '8',
    Num9 = '9',
    Escape = 256,
    LControl,
    LShift,
    LAlt,
    LSystem,
    RControl,
    RShift,
    RAlt,
    RSystem,
    Menu,
    LBracket,
    RBracket,
    SemiColon,
    Comma,
    Period,
    Quote,
    Slash,
};
```
BackSlash, Tilde, Equal, Dash, Space, Return, Back, 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, Count
Detailed Description

Definition of key codes for keyboard events.
## Enumeration Type Documentation

**enum sf::Key::Code**

### Enumerator:
- **LSystem**: OS specific key (left side): windows (Win and Linux), apple (MacOS), ...
- **RSystem**: OS specific key (right side): windows (Win and Linux), apple (MacOS), ...
- **LBracket**: [
- **RBracket**: ]
- **SemiColon**: ;
- **Comma**: ,
- **Period**: .
- **Quote**: '
- **Slash**: /
- **Tilde**: ~
- **Equal**: =
- **Dash**: ⋅
- **Add**: +
- **Subtract**: −
- **Multiply**: *
- **Divide**: /
- **Left**: Left arrow.
- **Right**: Right arrow.
- **Up**: Up arrow.
- **Down**: Down arrow.

Definition at line 41 of file Event.hpp.
sf::Mouse
sf::Mouse Namespace Reference

Definition of button codes for mouse events. More...
Enumerations

enum Button {
    Left,
    Right,
    Middle,
    XButton1,
    XButton2,
    ButtonCount
}
Detailed Description

Definition of button codes for mouse events.
sf::Style
sf::Style Namespace Reference

Enumeration of window creation styles. More...
Enumerations

```cpp
enum {
    None = 0,
    Titlebar = 1 << 0,
    Resize = 1 << 1,
    Close = 1 << 2,
    Fullscreen = 1 << 3
}
```
Detailed Description

Enumeration of window creation styles.
## Enumeration Type Documentation

### anonymous enum

<table>
<thead>
<tr>
<th>Enumerator:</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>None</strong></td>
<td>No border / title bar (this flag and all others are mutually exclusive).</td>
</tr>
<tr>
<td><strong>Titlebar</strong></td>
<td>Title bar + fixed border.</td>
</tr>
<tr>
<td><strong>Resize</strong></td>
<td>Titlebar + resizable border + maximize button.</td>
</tr>
<tr>
<td><strong>Close</strong></td>
<td>Titlebar + close button.</td>
</tr>
<tr>
<td><strong>Fullscreen</strong></td>
<td>Fullscreen mode (this flag and all others are mutually exclusive).</td>
</tr>
</tbody>
</table>

Definition at line **37** of file *WindowStyle.hpp*. 

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
Here is a list of all documented namespace members with links to the namespaces they belong to:

- **a** -
  - Add : sf::Blend, sf::Key
  - Alpha : sf::Blend

- **b** -
  - ButtonCount : sf::Joy

- **c** -
  - Close : sf::Style
  - Code : sf::Key
  - Comma : sf::Key
  - Count : sf::Joy

- **d** -
• Dash : sf::Key
• Divide : sf::Key
• Down : sf::Key

- e -
• Equal : sf::Key

- f -
• Fullscreen : sf::Style

- l -
• LBracket : sf::Key
• Left : sf::Key
• LSystem : sf::Key

- m -
• Mode : sf::Blend
• Multiply : sf::Key, sf::Blend

- n -
• None : sf::Blend, sf::Style

- p -
• Period : sf::Key

- q -
• Quote : sf::Key
- r -
  - RBracket : sf::Key
  - Resize : sf::Style
  - Right : sf::Key
  - RSystem : sf::Key

- s -
  - SemiColon : sf::Key
  - Slash : sf::Key
  - Subtract : sf::Key

- t -
  - Tilde : sf::Key
  - Titlebar : sf::Style

- u -
  - Up : sf::Key
<table>
<thead>
<tr>
<th>Main Page</th>
<th>Namespaces</th>
<th>Classes</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>File List</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Here is a list of all documented files with brief descriptions:

<table>
<thead>
<tr>
<th>File</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Arial.hpp</td>
<td>[code]</td>
</tr>
<tr>
<td>Audio.hpp</td>
<td>[code]</td>
</tr>
<tr>
<td>AudioDevice.cpp</td>
<td>[code]</td>
</tr>
<tr>
<td>AudioDevice.hpp</td>
<td>[code]</td>
</tr>
<tr>
<td>AudioResource.cpp</td>
<td>[code]</td>
</tr>
<tr>
<td>AudioResource.hpp</td>
<td>[code]</td>
</tr>
<tr>
<td>Clock.cpp</td>
<td>[code]</td>
</tr>
<tr>
<td>Clock.hpp</td>
<td>[code]</td>
</tr>
<tr>
<td>Color.cpp</td>
<td>[code]</td>
</tr>
<tr>
<td>Color.hpp</td>
<td>[code]</td>
</tr>
<tr>
<td>Config.hpp</td>
<td>[code]</td>
</tr>
<tr>
<td>Context.cpp</td>
<td>[code]</td>
</tr>
<tr>
<td>Context.hpp</td>
<td>[code]</td>
</tr>
<tr>
<td>Doxygen.hpp</td>
<td>[code]</td>
</tr>
<tr>
<td>Drawable.cpp</td>
<td>[code]</td>
</tr>
<tr>
<td>Drawable.hpp</td>
<td>[code]</td>
</tr>
<tr>
<td>Event.hpp</td>
<td>[code]</td>
</tr>
<tr>
<td>Font.cpp</td>
<td>[code]</td>
</tr>
<tr>
<td>Font.hpp</td>
<td>[code]</td>
</tr>
<tr>
<td>FontLoader.cpp</td>
<td>[code]</td>
</tr>
<tr>
<td>FontLoader.hpp</td>
<td>[code]</td>
</tr>
<tr>
<td>Ftp.cpp</td>
<td>[code]</td>
</tr>
<tr>
<td>Ftp.hpp</td>
<td>[code]</td>
</tr>
<tr>
<td>Glyph.hpp</td>
<td>[code]</td>
</tr>
<tr>
<td>Graphics.hpp</td>
<td>[code]</td>
</tr>
<tr>
<td>GraphicsContext.cpp</td>
<td>[code]</td>
</tr>
<tr>
<td>GraphicsContext.hpp</td>
<td>[code]</td>
</tr>
<tr>
<td>Http.cpp</td>
<td>[code]</td>
</tr>
<tr>
<td>File Name</td>
<td>Link</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>OpenGL.hpp</td>
<td>code</td>
</tr>
<tr>
<td>Packet.cpp</td>
<td>code</td>
</tr>
<tr>
<td>Packet.hpp</td>
<td>code</td>
</tr>
<tr>
<td>Platform.cpp</td>
<td>code</td>
</tr>
<tr>
<td>Platform.hpp</td>
<td>code</td>
</tr>
<tr>
<td>Win32/Platform.hpp</td>
<td>code</td>
</tr>
<tr>
<td>PostFX.cpp</td>
<td>code</td>
</tr>
<tr>
<td>PostFX.hpp</td>
<td>code</td>
</tr>
<tr>
<td>Randomizer.cpp</td>
<td>code</td>
</tr>
<tr>
<td>Randomizer.hpp</td>
<td>code</td>
</tr>
<tr>
<td>Rect.hpp</td>
<td>code</td>
</tr>
<tr>
<td>Rect.inl</td>
<td>code</td>
</tr>
<tr>
<td>RenderTarget.cpp</td>
<td>code</td>
</tr>
<tr>
<td>RenderTarget.hpp</td>
<td>code</td>
</tr>
<tr>
<td>RenderWindow.cpp</td>
<td>code</td>
</tr>
<tr>
<td>RenderWindow.hpp</td>
<td>code</td>
</tr>
<tr>
<td>Resource.hpp</td>
<td>code</td>
</tr>
<tr>
<td>Resource.inl</td>
<td>code</td>
</tr>
<tr>
<td>ResourcePtr.inl</td>
<td>code</td>
</tr>
<tr>
<td>Selector.hpp</td>
<td>code</td>
</tr>
<tr>
<td>Selector.inl</td>
<td>code</td>
</tr>
<tr>
<td>SelectorBase.cpp</td>
<td>code</td>
</tr>
<tr>
<td>SelectorBase.hpp</td>
<td>code</td>
</tr>
<tr>
<td>SFML_Main.cpp</td>
<td>code</td>
</tr>
<tr>
<td>Shape.cpp</td>
<td>code</td>
</tr>
<tr>
<td>Shape.hpp</td>
<td>code</td>
</tr>
<tr>
<td>Sleep.cpp</td>
<td>code</td>
</tr>
<tr>
<td>Sleep.hpp</td>
<td>code</td>
</tr>
<tr>
<td>SocketHelper.cpp</td>
<td>code</td>
</tr>
<tr>
<td>SocketHelper.hpp</td>
<td>code</td>
</tr>
<tr>
<td>Win32/SocketHelper.hpp</td>
<td>code</td>
</tr>
<tr>
<td>File Name</td>
<td>Status</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>Unicode.inl</td>
<td>[code]</td>
</tr>
<tr>
<td>Vector2.hpp</td>
<td>[code]</td>
</tr>
<tr>
<td>Vector2.inl</td>
<td>[code]</td>
</tr>
<tr>
<td>Vector3.hpp</td>
<td>[code]</td>
</tr>
<tr>
<td>Vector3.inl</td>
<td>[code]</td>
</tr>
<tr>
<td>VideoMode.cpp</td>
<td>[code]</td>
</tr>
<tr>
<td>VideoMode.hpp</td>
<td>[code]</td>
</tr>
<tr>
<td>Cocoa/VideoModeSupport.cpp</td>
<td>[code]</td>
</tr>
<tr>
<td>Template for new ports/VideoModeSupport.cpp</td>
<td>[code]</td>
</tr>
<tr>
<td>Win32/VideoModeSupport.cpp</td>
<td>[code]</td>
</tr>
<tr>
<td>Cocoa/VideoModeSupport.hpp</td>
<td>[code]</td>
</tr>
<tr>
<td>Template for new ports/VideoModeSupport.hpp</td>
<td>[code]</td>
</tr>
<tr>
<td>VideoModeSupport.hpp</td>
<td>[code]</td>
</tr>
<tr>
<td>Win32/VideoModeSupport.hpp</td>
<td>[code]</td>
</tr>
<tr>
<td>View.cpp</td>
<td>[code]</td>
</tr>
<tr>
<td>View.hpp</td>
<td>[code]</td>
</tr>
<tr>
<td>Window.cpp</td>
<td>[code]</td>
</tr>
<tr>
<td>Window/Window.hpp</td>
<td>[code]</td>
</tr>
<tr>
<td>Window.hpp</td>
<td>[code]</td>
</tr>
<tr>
<td>WindowHandle.hpp</td>
<td>[code]</td>
</tr>
<tr>
<td>WindowImpl.cpp</td>
<td>[code]</td>
</tr>
<tr>
<td>WindowImpl.hpp</td>
<td>[code]</td>
</tr>
<tr>
<td>WindowImplCocoa.hpp</td>
<td>[code]</td>
</tr>
<tr>
<td>WindowImplWin32.cpp</td>
<td>[code]</td>
</tr>
<tr>
<td>WindowImplWin32.hpp</td>
<td>[code]</td>
</tr>
<tr>
<td>WindowImplXXX.cpp</td>
<td>[code]</td>
</tr>
<tr>
<td>WindowImplXXX.hpp</td>
<td>[code]</td>
</tr>
<tr>
<td>WindowListener.hpp</td>
<td>[code]</td>
</tr>
<tr>
<td>WindowSettings.hpp</td>
<td>[code]</td>
</tr>
<tr>
<td>WindowStyle.hpp</td>
<td>[code]</td>
</tr>
</tbody>
</table>
# Class Index

| A | C | D | E | F | G | H | I | L | M | N | P | R | S | T | U | V | W |
| AudioResource (sf) | Event::TextEvent (sf) | RenderWi |
| Clock (sf) | Font (sf) | Listener (sf) |
| Color (sf) | Ftp (sf) | Lock (sf) |
| Context (sf) | Ftp::DirectoryResponse (sf) | Resource |
| | Ftp::ListingResponse (sf) | |
| | Ftp::Response (sf) | |
| | Matrix3 (sf) | Selector |
| | Music (sf) | SelectorBase |
| | Mutex (sf) | Shape |
| | NonCopyable (sf) | SocketHelper |
| | SocketTCP (sf) | SoundBuffer |
| | SocketUDP (sf) | SoundBufferRecorder |
| | Packet (sf) | SoundBuffer |
| | PostFX (sf) | SoundBuffer |
| | Randomizer (sf) | SoundBuffer |
| |_Rect (sf) | SoundStream |
| | RenderTarget SoundStream (sf) | SoundStream |
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gomila@gmail.com)

// This software is provided 'as-is', without any expressed or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

// 1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such and must not be misrepresented as being the original software.

// 3. This notice may not be removed or altered.

#ifndef SFML_AUDIORESOURCE_HPP
#define SFML_AUDIORESOURCE_HPP

// Headers
#include <SFML/Config.hpp>

namespace sf
{
    class SFML_API AudioResource
    {

protected :

AudioResource();

AudioResource(const AudioResource&);

virtual ~AudioResource();

};

} // namespace sf

#endif // SFML_AUDIORESOURCE_HPP
<table>
<thead>
<tr>
<th>Main Page</th>
<th>Namespaces</th>
<th>Classes</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class List</td>
<td>Class Index</td>
<td>Class Hierarchy</td>
<td>Class Members</td>
</tr>
</tbody>
</table>
sf::AudioResource Member List

This is the complete list of members for sf::AudioResource, including all inherited members.

<table>
<thead>
<tr>
<th>Function</th>
<th>sf::AudioResource</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>AudioResource()</td>
<td></td>
<td>protected</td>
</tr>
<tr>
<td>AudioResource(const AudioResource&amp;)</td>
<td></td>
<td>protected</td>
</tr>
<tr>
<td>~AudioResource()</td>
<td></td>
<td>protected, virtual</td>
</tr>
<tr>
<td>Main Page</td>
<td>Namespaces</td>
<td>Classes</td>
</tr>
<tr>
<td>-----------</td>
<td>------------</td>
<td>---------</td>
</tr>
<tr>
<td>File List</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any expressed or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.

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

// Headers
#include <SFML/Audio/AudioResource.hpp>
#include <SFML/Audio/AudioDevice.hpp>

namespace sf {

AudioResource::AudioResource()
{
    priv::AudioDevice::AddReference();
}
}
AudioResource::AudioResource(const AudioResource &)
{
    priv::AudioDevice::AddReference();
}

AudioResource::~AudioResource()
{
    priv::AudioDevice::RemoveReference();
}

// namespace sf
<table>
<thead>
<tr>
<th>Main Page</th>
<th>Namespaces</th>
<th>Classes</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>File List</td>
</tr>
</tbody>
</table>

**SFML**

Simple and Fast Multimedia Library
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gomila@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
// you must not claim that you wrote the original software.
// If you use this software in a product, an acknowledgment
// in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
// and must not be misrepresented as being the original software.

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

#ifndef SFML_CLOCK_HPP
#define SFML_CLOCK_HPP

// Headers
#include <SFML/Config.hpp>

namespace sf {

class SFML_API Clock {

}
public :

Clock();

float GetElapsedTime() const;

void Reset();

private :

// Member data
double myStart_time;

};

}; // namespace sf

#endif // SFML_CLOCK_HPP
**sf::Clock Member List**

This is the complete list of members for *sf::Clock*, including all inherited members.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock()</td>
<td>sf::Clock</td>
</tr>
<tr>
<td>GetElapsedTime()</td>
<td>const sf::Clock</td>
</tr>
<tr>
<td>Reset()</td>
<td>sf::Clock</td>
</tr>
</tbody>
</table>

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved  ::
Documentation generated by doxygen 1.5.2  ::
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
// you must not claim that you wrote the original software.
// If you use this software in a product, an acknowledgment
// in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
// and must not be misrepresented as being the original software.

// 3. This notice may not be removed or altered.

// Headers
#include <SFML/System/Clock.hpp>
#include <SFML/System/Platform.hpp>

namespace sf
{
    Clock::Clock()
    {
        Reset();
    }
}
float Clock::GetElapsedTime() const
{
    return static_cast<float>(sf::priv::Platform::GetSystemTime() - myStartTime);
}

void Clock::Reset()
{
    myStartTime = sf::priv::Platform::GetSystemTime();
} // namespace sf
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.

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

#ifndef SFML_COLOR_HPP
#define SFML_COLOR_HPP

// Headers
#include <SFML/Config.hpp>

namespace sf
{
    class SFML_API Color
    {

public:
  Color();
  Color(Uint8 R, Uint8 G, Uint8 B, Uint8 A = 255);
  Color& operator +=(const Color& Other);
  Color& operator *(=)(const Color& Other);
  bool operator ==(const Color& Other) const;
  bool operator !=(const Color& Other) const;

// Static member data
static const Color Black;
static const Color White;
static const Color Red;
static const Color Green;
static const Color Blue;
static const Color Yellow;
static const Color Magenta;
static const Color Cyan;

// Member data
Uint8 r;
Uint8 g;
Uint8 b;
Uint8 a;

SFML_API Color operator +(const Color& Color1, const Color& Other);
SFML_API Color operator *(const Color& Color1, const Color& Other);
} // namespace sf
#endif // SFML_COLOR_HPP
# sf::Color Member List

This is the complete list of members for `sf::Color`, including all inherited members.

<table>
<thead>
<tr>
<th>Function</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>sf::Color</td>
</tr>
<tr>
<td>b</td>
<td>sf::Color</td>
</tr>
<tr>
<td>Black</td>
<td>sf::Color [static]</td>
</tr>
<tr>
<td>Blue</td>
<td>sf::Color [static]</td>
</tr>
<tr>
<td>Color()</td>
<td>sf::Color</td>
</tr>
<tr>
<td>Color(Uint8 R, Uint8 G, Uint8 B, Uint8 A=255)</td>
<td>sf::Color</td>
</tr>
<tr>
<td>Cyan</td>
<td>sf::Color [static]</td>
</tr>
<tr>
<td>g</td>
<td>sf::Color</td>
</tr>
<tr>
<td>Green</td>
<td>sf::Color [static]</td>
</tr>
<tr>
<td>Magenta</td>
<td>sf::Color</td>
</tr>
<tr>
<td>operator!=(const Color &amp;Other) const</td>
<td>sf::Color</td>
</tr>
<tr>
<td>operator*=(const Color &amp;Other)</td>
<td>sf::Color</td>
</tr>
<tr>
<td>operator+=(const Color &amp;Other)</td>
<td>sf::Color</td>
</tr>
<tr>
<td>operator==(const Color &amp;Other) const</td>
<td>sf::Color</td>
</tr>
<tr>
<td>r</td>
<td>sf::Color</td>
</tr>
<tr>
<td>Red</td>
<td>sf::Color [static]</td>
</tr>
<tr>
<td>White</td>
<td>sf::Color [static]</td>
</tr>
<tr>
<td>Yellow</td>
<td>sf::Color [static]</td>
</tr>
</tbody>
</table>
//SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any
// In no event will the authors be held liable for any
// Permission is granted to anyone to use this software
// including commercial applications, and to alter it
// subject to the following restrictions:
// 1. The origin of this software must not be misrepresented;
// you must not claim that you wrote the original software.
// If you use this software in a product, an
// in the product documentation would be appreciated.
// 2. Altered source versions must be plainly marked
// and must not be misrepresented as being the original
// 3. This notice may not be removed or altered

// Headers
#include <SFML/Graphics/Color.hpp>
#include <algorithm>

namespace sf
{
    // Static member data
    const Color Color::Black(0, 0, 0);
    const Color Color::White(255, 255, 255);
    const Color Color::Red(255, 0, 0);
const Color Color::Green(0, 255, 0);
const Color Color::Blue(0, 0, 255);
const Color Color::Yellow(255, 255, 0);
const Color Color::Magenta(255, 0, 255);
const Color Color::Cyan(0, 255, 255);

Color::Color() :
  r(0),
  g(0),
  b(0),
  a(255)
{
}

Color::Color(Uint8 R, Uint8 G, Uint8 B, Uint8 A) :
  r(R),
  g(G),
  b(B),
  a(A)
{
}

Color& Color::operator +=(const Color& Other) {
  r = static_cast<Uint8>(std::min(r + Other.r, 255));
  g = static_cast<Uint8>(std::min(g + Other.g, 255));
  b = static_cast<Uint8>(std::min(b + Other.b, 255));
  a = static_cast<Uint8>(std::min(a + Other.a, 255));
  return *this;
}

return *this;
Color& Color::operator *(const Color& Other) {
    r = static_cast<Uint8>(r * Other.r / 255);
    g = static_cast<Uint8>(g * Other.g / 255);
    b = static_cast<Uint8>(b * Other.b / 255);
    a = static_cast<Uint8>(a * Other.a / 255);
    return *this;
}

bool Color::operator ==(const Color& Other) const {
    return (r == Other.r) && (g == Other.g) && (b == Other.b) && (a == Other.a);
}

bool Color::operator !=(const Color& Other) const {
    return (r != Other.r) || (g != Other.g) || (b != Other.b) || (a != Other.a);
}

Color operator +(const Color& Color1, const Color& Color2) {
    Color c = Color1;
    c += Color2;
    return c;
}

Color operator *(const Color& Color1, const Color& Color2) {
    Color c = Color1;
    c *= Color2;
    return c;
}
return c;
} // namespace sf
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.

// 3. This notice may not be removed or altered.

#ifndef SFML_CONTEXT_HPP
#define SFML_CONTEXT_HPP

namespace sf
{

namespace priv
class WindowImpl;

class SFML_API Context : NonCopyable
{
public:
    Context();
    ~Context();
    void SetActive(bool Active);
    static bool IsContextActive();
    static Context& GetGlobal();
private:
    // Member data
    priv::WindowImpl* myDummyWindow;
};

// namespace sf

#endif // SFML_CONTEXT_HPP
<table>
<thead>
<tr>
<th>Main Page</th>
<th>Namespaces</th>
<th>Classes</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class List</td>
<td>Class Index</td>
<td>Class Hierarchy</td>
<td>Class Members</td>
</tr>
</tbody>
</table>
sf::Context Member List

This is the complete list of members for sf::Context, including all inherited members.

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context()</td>
<td>sf::Context</td>
<td></td>
</tr>
<tr>
<td>GetGlobal()</td>
<td>sf::Context</td>
<td>[static]</td>
</tr>
<tr>
<td>IsContextActive()</td>
<td>sf::Context</td>
<td>[static]</td>
</tr>
<tr>
<td>NonCopyable()</td>
<td>sf::NonCopyable</td>
<td>[inline, private]</td>
</tr>
<tr>
<td>.SetActive(bool Active)</td>
<td>sf::Context</td>
<td></td>
</tr>
<tr>
<td>~Context()</td>
<td>sf::Context</td>
<td></td>
</tr>
</tbody>
</table>

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.

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

// Headers

#include <SFML/Window/Context.hpp>
#include <SFML/Window/WindowImpl.hpp>

namespace
{
    // Make sure the dummy context is created at global startup
    sf::Context& Dummy = sf::Context::GetGlobal();
}

namespace sf
{
    Context::Context()
    {
        myDummyWindow = priv::WindowImpl::New();
    }

    Context::~Context()
    {
        delete myDummyWindow;
    }

    void Context::SetActive(bool Active)
    {
        myDummyWindow->SetActive(Active);
    }

    bool Context::IsContextActive()
    {
        return priv::WindowImpl::IsContextActive();
    }

    Context& Context::GetGlobal()
    {
        static Context* GlobalContext = new Context;
        return *GlobalContext;
    }

} // namespace sf
SFML - Simple and Fast Multimedia Library
Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

This software is provided 'as-is', without any express or implied warranty.
In no event will the authors be held liable for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,
including commercial applications, and to alter it and redistribute it freely,
subject to the following restrictions:

1. The origin of this software must not be misrepresented;
you must not claim that you wrote the original software.
If you use this software in a product, an
acknowledgment in the product documentation would be appreciated.

2. Altered source versions must be plainly marked as such,
and must not be misrepresented as being the original software.

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

#ifndef SFML_DRAWABLE_HPP
#define SFML_DRAWABLE_HPP

namespace sf
{

#include <SFML/System/Vector2.hpp>
#include <SFML/Graphics/Color.hpp>
#include <SFML/Graphics/Matrix3.hpp>

namespace sf
{
class RenderTarget;

namespace Blend {
    enum Mode {
        Alpha, Add, Multiply, None
    }
}

class SFML_API Drawable {
    public:
    Drawable(const Vector2f& Position = Vector2f);
    virtual ~Drawable();
    void SetPosition(float X, float Y);
    void SetPosition(const Vector2f& Position);
    void SetX(float X);
    void SetY(float Y);
    void SetScale(float ScaleX, float ScaleY);
    void SetScale(const Vector2f& Scale);
    void SetScaleX(float FactorX);
    void SetScaleY(float FactorY);
void SetCenter(float CenterX, float CenterY);
void SetCenter(const Vector2f& Center);
void SetRotation(float Rotation);
void SetColor(const Color& Col);
void SetBlendMode(Blend::Mode Mode);
const Vector2f& GetPosition() const;
const Vector2f& GetScale() const;
const Vector2f& GetCenter() const;
float GetRotation() const;
const Color& GetColor() const;
Blend::Mode GetBlendMode() const;
void Move(float OffsetX, float OffsetY);
void Move(const Vector2f& Offset);
void Scale(float FactorX, float FactorY);
void Scale(const Vector2f& Factor);
void Rotate(float Angle);
sf::Vector2f TransformToLocal(const sf::Vector2f&);
sf::Vector2f TransformToGlobal(const sf::Vector2f&);
protected:
const Matrix3& GetMatrix() const;
const Matrix3& GetInverseMatrix() const;

private:
friend class RenderTarget;

void Draw(RenderTarget& Target) const;
virtual void Render(RenderTarget& Target) const;

// Member data
Vector2f myPosition;
Vector2f myScale;
Vector2f myCenter;
float myRotation;
Color myColor;
Blend::Mode myBlendMode;
mutable bool myNeedUpdate;
mutable bool myInvNeedUpdate;
mutable Matrix3 myMatrix;
mutable Matrix3 myInvMatrix;

};

} // namespace sf

#endif // SFML_DRAWABLE_HPP
This is the complete list of members for sf::Drawable, including all inherited members.

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
<th>Access Type</th>
</tr>
</thead>
</table>
| Drawable | (const Vector2f &Position=Vector2f(0, 0),
const Vector2f &Scale=Vector2f(1, 1), float
Rotation=0.f, const Color &Col=Color(255, 255, 255, 255)) | sf::Drawable |
<p>| GetBlendMode() const | sf::Drawable |
| GetCenter() const | sf::Drawable |
| GetColor() const | sf::Drawable |
| GetInverseMatrix() const | sf::Drawable [protected] |
| GetMatrix() const | sf::Drawable [protected] |
| GetPosition() const | sf::Drawable |
| GetRotation() const | sf::Drawable |
| GetScale() const | sf::Drawable |
| Move(float OffsetX, float OffsetY) | sf::Drawable |
| Move(const Vector2f &amp;Offset) | sf::Drawable |
| RenderTarget (defined in sf::Drawable) | sf::Drawable [friend] |
| Rotate(float Angle) | sf::Drawable |
| Scale(float FactorX, float FactorY) | sf::Drawable |
| Scale(const Vector2f &amp;Factor) | sf::Drawable |
| SetBlendMode(Blend::Mode Mode) | sf::Drawable |
| SetCenter(float CenterX, float CenterY) | sf::Drawable |
| SetCenter(const Vector2f &amp;Center) | sf::Drawable |
| SetColor(const Color &amp;Col) | sf::Drawable |
| SetPosition(float X, float Y) | sf::Drawable |
| SetPosition(const Vector2f &amp;Position) | sf::Drawable |
| SetRotation(float Rotation) | sf::Drawable |
| SetScale(float ScaleX, float ScaleY) | sf::Drawable |
| SetScale(const Vector2f &amp;Scale) | sf::Drawable |
| SetScaleX(float FactorX) | sf::Drawable |
| SetScaleY(float FactorY) | sf::Drawable |
| SetX(float X) | sf::Drawable |
| SetY(float Y) | sf::Drawable |</p>
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransformToGlobal(const sf::Vector2f &amp;Point) const</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td>TransformToLocal(const sf::Vector2f &amp;Point) const</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td>~Drawable()</td>
<td>sf::Drawable [virtual]</td>
</tr>
</tbody>
</table>

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
<table>
<thead>
<tr>
<th>Main Page</th>
<th>Namespaces</th>
<th>Classes</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>File List</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SFML - Simple and Fast Multimedia Library

Copyright (C) 2007-2009 Laurent Gomila (laurent.gomila@gmail.com)

This software is provided 'as-is', without any express or implied warranty.
In no event will the authors be held liable for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.

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

Headers

#include <SFML/Graphics/Drawable.hpp>
#include <SFML/Graphics/GraphicsContext.hpp>
#include <SFML/Window/Window.hpp>
#include <math.h>

namespace sf
{
Drawable::Drawable(const Vector2f& Position, const sf::Drawable& myPosition) (Position),
myScale (Scale),
myCenter (0, 0),
myRotation (Rotation),
myColor (Col),
myBlendMode (Blend::Alpha),
myNeedUpdate (true),
myInvNeedUpdate(true)

Drawable::~Drawable()

// Nothing to do

void Drawable::SetPosition(float X, float Y)

void Drawable::SetPosition(const Vector2f& Position)

void Drawable::SetX(float X)
void Drawable::SetY(float Y) {
    myPosition.y = Y;
    myNeedUpdate = true;
    myInvNeedUpdate = true;
}

void Drawable::SetScale(float ScaleX, float ScaleY) {
    SetScaleX(ScaleX);
    SetScaleY(ScaleY);
}

void Drawable::SetScale(const Vector2f& Scale) {
    SetScaleX(Scale.x);
    SetScaleY(Scale.y);
}

void Drawable::SetScaleX(float FactorX) {
    if (FactorX > 0) {
        myScale.x = FactorX;
        myNeedUpdate = true;
        myInvNeedUpdate = true;
    }
}

void Drawable::SetScaleY(float FactorY)
if (FactorY > 0) {
    myScale.y = FactorY;
    myNeedUpdate = true;
    myInvNeedUpdate = true;
}

void Drawable::SetCenter(float CenterX, float CenterY) {
    myCenter.x = CenterX;
    myCenter.y = CenterY;
    myNeedUpdate = true;
    myInvNeedUpdate = true;
}

void Drawable::SetCenter(const Vector2f& Center) {
    SetCenter(Center.x, Center.y);
}

void Drawable::SetRotation(float Rotation) {
    myRotation = static_cast<float>(fmod(Rotation, 360.f));
    if (myRotation < 0)
        myRotation += 360.f;
    myNeedUpdate = true;
    myInvNeedUpdate = true;
}

void Drawable::SetColor(const Color& Col) {

```cpp
myColor = Col;

void Drawable::SetBlendMode(Blend::Mode Mode)
{
    myBlendMode = Mode;
}

const Vector2f& Drawable::GetPosition() const
{
    return myPosition;
}

const Vector2f& Drawable::GetScale() const
{
    return myScale;
}

const Vector2f& Drawable::GetCenter() const
{
    return myCenter;
}

float Drawable::GetRotation() const
{
    return myRotation;
}

const Color& Drawable::GetColor() const
{
    return myColor;
}
```
Blend::Mode Drawable::GetBlendMode() const
{
    return myBlendMode;
}

void Drawable::Move(float OffsetX, float OffsetY)
{
    SetX(myPosition.x + OffsetX);
    SetY(myPosition.y + OffsetY);
}

void Drawable::Move(const Vector2f& Offset)
{
    Move(Offset.x, Offset.y);
}

void Drawable::Scale(float FactorX, float FactorY)
{
    SetScaleX(myScale.x * FactorX);
    SetScaleY(myScale.y * FactorY);
}

void Drawable::Scale(const Vector2f& Factor)
{
    Scale(Factor.x, Factor.y);
}

void Drawable::Rotate(float Angle)
{
SetRotation(myRotation + Angle);

sf::Vector2f Drawable::TransformToLocal(const sf::Vector2f& Point)
{
    return GetInverseMatrix().Transform(Point);
}

sf::Vector2f Drawable::TransformToGlobal(const sf::Vector2f& Point)
{
    return GetMatrix().Transform(Point);
}

const Matrix3& Drawable::GetMatrix() const
{
    // First recomputete it if needed
    if (myNeedUpdate)
    {
        myMatrix.SetFromTransformations(myCenter, myPosition, myRotation, myScale);
        myNeedUpdate = false;
    }

    return myMatrix;
}

const Matrix3& Drawable::GetInverseMatrix() const
{
    // First recomputete it if needed
    if (myInvNeedUpdate)
    {
        myInvMatrix = GetMatrix().GetInverse();
        myInvNeedUpdate = false;
    }
}

return myInvMatrix;

void Drawable::Draw(RenderTarget& Target) const {
    // Save the current modelview matrix and set
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glPushMatrix());
    GLCheck(glMultMatrixf(GetMatrix().Get4x4Elements()));

    // Setup alpha-blending
    if (myBlendMode == Blend::None)
        { GLCheck(glDisable(GL_BLEND)); }
    else
    { GLCheck(glEnable(GL_BLEND));
        switch (myBlendMode)
        {
            case Blend::Alpha : GLCheck(glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA));
            case Blend::Add  : GLCheck(glBlendFunc(GL_SRC_ALPHA, GL_ONE));
            case Blend::Multiply : GLCheck(glBlendFunc(GL_DST_COLOR, GL_ZERO));
            default :
        }
    }

    // Set color
    GLCheck(glColor4f(myColor.r / 255.f, myColor.g / 255.f, myColor.b / 255.f, myColor.a / 255.f));

    // Let the derived class render the object
    Render(Target);

    // Restore the previous modelview matrix
    GLCheck(glMatrixMode(GL_MODELVIEW));
GLCheck(glPopMatrix());
}

// namespace sf
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)
//
// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.
// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:
// 1. The origin of this software must not be misrepresented;
// you must not claim that you wrote the original software.
// If you use this software in a product, an acknowledgment
// in the product documentation would be appreciated but is not required.
// 2. Altered source versions must be plainly marked as such,
// and must not be misrepresented as being the original software.
// 3. This notice may not be removed or altered.

#ifndef SFML_EVENT_HPP
#define SFML_EVENT_HPP

// Headers
#include <SFML/Config.hpp>

namespace sf {

namespace Key {


```c
enum Code {
    A = 'a',
    B = 'b',
    C = 'c',
    D = 'd',
    E = 'e',
    F = 'f',
    G = 'g',
    H = 'h',
    I = 'i',
    J = 'j',
    K = 'k',
    L = 'l',
    M = 'm',
    N = 'n',
    O = 'o',
    P = 'p',
    Q = 'q',
    R = 'r',
    S = 's',
    T = 't',
    U = 'u',
    V = 'v',
    W = 'w',
    X = 'x',
    Y = 'y',
    Z = 'z',
    Num0 = '0',
    Num1 = '1',
    Num2 = '2',
    Num3 = '3',
    Num4 = '4',
    Num5 = '5',
    Num6 = '6',
    Num7 = '7',
    Num8 = '8',
};
```
Num9 = '9',
Escape = 256,
LControl,
LShift,
LAlt,
LSYSTEM,
RControl,
RShift,
RAlt,
RSystem,
Menu,
LBracket,
RBracket,
Semicolon,
Comma,
Period,
Quote,
Slash,
BackSlash,
Tilde,
Equal,
Dash,
Space,
Return,
Back,
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,

Count // Keep last -- total number of keys

};
}
enum Button
{
    Left,
    Right,
    Middle,
    XButton1,
    XButton2,
    ButtonCount // Keep last -- total number
};

namespace Joy
{
    enum Axis
    {
        AxisX,
        AxisY,
        AxisZ,
        AxisR,
        AxisU,
        AxisV,
        AxisPOV,
        AxisCount // Keep last -- total number
    }

    enum
    {
        Count   = 4,
        ButtonCount = 32
    }
}

class Event
```cpp
{ public:
    struct KeyEvent
    {
        Key::Code Code;
        bool Alt;
        bool Control;
        bool Shift;
    };

    struct TextEvent
    {
        Uint32 Unicode;
    };

    struct MouseMoveEvent
    {
        int X;
        int Y;
    };

    struct MouseButtonEvent
    {
        Mouse::Button Button;
        int X;
        int Y;
    };

    struct MouseWheelEvent
    {
        int Delta;
    };

    struct JoyMoveEvent
    {
        unsigned int JoystickId;
    }
};
```
struct Joy::Axis
{
    float Position;
};

struct JoyButtonEvent
{
    unsigned int JoystickId;
    unsigned int Button;
};

struct SizeEvent
{
    unsigned int Width;
    unsigned int Height;
};

enum EventType
{
    Closed,
    Resized,
    LostFocus,
    GainedFocus,
    TextEntered,
    KeyPressed,
    KeyReleased,
    MouseWheelMoved,
    MouseButtonPressed,
    MouseButtonReleased,
    MouseMoved,
    MouseEntered,
    MouseLeft,
    JoyButtonPressed,
    JoyButtonReleased,
    JoyMoved,
    Count // Keep last -- total number of e
};
// Member data
EventType Type;
union {
    KeyEvent Key;
    TextEvent Text;
    MouseMoveEvent MouseMove;
    MouseButtonEvent MouseButton;
    MouseWheelEvent MouseWheel;
    JoyMoveEvent JoyMove;
    JoyButtonEvent JoyButton;
    SizeEvent Size;
};
} // namespace sf
#endif // SFML_EVENT_HPP
## sf::Event Member List

This is the complete list of members for `sf::Event`, including all inherited members.

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Closed</td>
<td>enum value (defined in <code>sf::Event</code>)</td>
<td><code>sf::Event</code></td>
</tr>
<tr>
<td>Count</td>
<td>enum value (defined in <code>sf::Event</code>)</td>
<td><code>sf::Event</code></td>
</tr>
<tr>
<td>EventType</td>
<td>enum name</td>
<td><code>sf::Event</code></td>
</tr>
<tr>
<td>GainedFocus</td>
<td>enum value (defined in <code>sf::Event</code>)</td>
<td><code>sf::Event</code></td>
</tr>
<tr>
<td>JoyButton</td>
<td>(defined in <code>sf::Event</code>)</td>
<td><code>sf::Event</code></td>
</tr>
<tr>
<td>JoyButtonPressed</td>
<td>enum value (defined in <code>sf::Event</code>)</td>
<td><code>sf::Event</code></td>
</tr>
<tr>
<td>JoyButtonReleased</td>
<td>enum value (defined in <code>sf::Event</code>)</td>
<td><code>sf::Event</code></td>
</tr>
<tr>
<td>JoyMove</td>
<td>(defined in <code>sf::Event</code>)</td>
<td><code>sf::Event</code></td>
</tr>
<tr>
<td>JoyMoved</td>
<td>enum value (defined in <code>sf::Event</code>)</td>
<td><code>sf::Event</code></td>
</tr>
<tr>
<td>Key</td>
<td>(defined in <code>sf::Event</code>)</td>
<td><code>sf::Event</code></td>
</tr>
<tr>
<td>KeyPressed</td>
<td>enum value (defined in <code>sf::Event</code>)</td>
<td><code>sf::Event</code></td>
</tr>
<tr>
<td>KeyReleased</td>
<td>enum value (defined in <code>sf::Event</code>)</td>
<td><code>sf::Event</code></td>
</tr>
<tr>
<td>LostFocus</td>
<td>enum value (defined in <code>sf::Event</code>)</td>
<td><code>sf::Event</code></td>
</tr>
<tr>
<td>MouseButton</td>
<td>(defined in <code>sf::Event</code>)</td>
<td><code>sf::Event</code></td>
</tr>
<tr>
<td>MouseButtonPressed</td>
<td>enum value (defined in <code>sf::Event</code>)</td>
<td><code>sf::Event</code></td>
</tr>
<tr>
<td>MouseButtonReleased</td>
<td>enum value (defined in <code>sf::Event</code>)</td>
<td><code>sf::Event</code></td>
</tr>
<tr>
<td>MouseEntered</td>
<td>enum value (defined in <code>sf::Event</code>)</td>
<td><code>sf::Event</code></td>
</tr>
<tr>
<td>MouseLeft</td>
<td>enum value (defined in <code>sf::Event</code>)</td>
<td><code>sf::Event</code></td>
</tr>
<tr>
<td>MouseMove</td>
<td>(defined in <code>sf::Event</code>)</td>
<td><code>sf::Event</code></td>
</tr>
<tr>
<td>MouseMoved</td>
<td>enum value (defined in <code>sf::Event</code>)</td>
<td><code>sf::Event</code></td>
</tr>
<tr>
<td>MouseWheel</td>
<td>(defined in <code>sf::Event</code>)</td>
<td><code>sf::Event</code></td>
</tr>
<tr>
<td>MouseWheelMoved</td>
<td>enum value (defined in <code>sf::Event</code>)</td>
<td><code>sf::Event</code></td>
</tr>
<tr>
<td>Resized</td>
<td>enum value (defined in <code>sf::Event</code>)</td>
<td><code>sf::Event</code></td>
</tr>
<tr>
<td>Size</td>
<td>(defined in <code>sf::Event</code>)</td>
<td><code>sf::Event</code></td>
</tr>
<tr>
<td>Text</td>
<td>(defined in <code>sf::Event</code>)</td>
<td><code>sf::Event</code></td>
</tr>
<tr>
<td>TextEntered</td>
<td>enum value (defined in <code>sf::Event</code>)</td>
<td><code>sf::Event</code></td>
</tr>
<tr>
<td>Type</td>
<td></td>
<td><code>sf::Event</code></td>
</tr>
</tbody>
</table>
sf::Event::Event::JoyButtonEvent

Member List

This is the complete list of members for sf::Event::Event::JoyButtonEvent, including all inherited members.

<table>
<thead>
<tr>
<th>Member</th>
<th>Definition</th>
<th>sf::Event::Event::JoyButtonEvent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Button</td>
<td>(defined in sf::Event::Event::JoyButtonEvent)</td>
<td>sf::Event::Event::JoyButtonEvent</td>
</tr>
<tr>
<td>JoystickId</td>
<td>(defined in sf::Event::Event::JoyButtonEvent)</td>
<td>sf::Event::Event::JoyButtonEvent</td>
</tr>
</tbody>
</table>

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
<table>
<thead>
<tr>
<th>Main Page</th>
<th>Namespaces</th>
<th>Classes</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class List</td>
<td>Class Index</td>
<td>Class Hierarchy</td>
<td>Class Members</td>
</tr>
</tbody>
</table>
sf::Event::Event::JoyMoveEvent Member List

This is the complete list of members for sf::Event::Event::JoyMoveEvent, including all inherited members.

<table>
<thead>
<tr>
<th>Member</th>
<th>Definition</th>
<th>sf::Event::Event::JoyMoveEvent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Axis</td>
<td>(defined in sf::Event::Event::JoyMoveEvent)</td>
<td>sf::Event::Event::JoyMoveEvent</td>
</tr>
<tr>
<td>JoystickId</td>
<td>(defined in sf::Event::Event::JoyMoveEvent)</td>
<td>sf::Event::Event::JoyMoveEvent</td>
</tr>
<tr>
<td>Position</td>
<td>(defined in sf::Event::Event::JoyMoveEvent)</td>
<td>sf::Event::Event::JoyMoveEvent</td>
</tr>
</tbody>
</table>

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
sf::Event::Event::KeyEvent Member List

This is the complete list of members for sf::Event::Event::KeyEvent, including all inherited members.

<table>
<thead>
<tr>
<th>Member</th>
<th>Defined In</th>
<th>sf::Event::Event::KeyEvent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alt</td>
<td>sf::Event::Event::KeyEvent</td>
<td>sf::Event::Event::KeyEvent</td>
</tr>
<tr>
<td>Code</td>
<td>sf::Event::Event::KeyEvent</td>
<td>sf::Event::Event::KeyEvent</td>
</tr>
<tr>
<td>Control</td>
<td>sf::Event::Event::KeyEvent</td>
<td>sf::Event::Event::KeyEvent</td>
</tr>
<tr>
<td>Shift</td>
<td>sf::Event::Event::KeyEvent</td>
<td>sf::Event::Event::KeyEvent</td>
</tr>
</tbody>
</table>

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
**sf::Event::Event::MouseButtonEvent**

Member List

This is the complete list of members for `sf::Event::Event::MouseButtonEvent`, including all inherited members.

<table>
<thead>
<tr>
<th>Member</th>
<th>Definition</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Button</td>
<td>(defined in <code>sf::Event::Event::MouseButtonEvent</code>)</td>
<td><code>sf::Event::Event::MouseButtonEvent</code></td>
</tr>
<tr>
<td>X</td>
<td>(defined in <code>sf::Event::Event::MouseButtonEvent</code>)</td>
<td><code>sf::Event::Event::MouseButtonEvent</code></td>
</tr>
<tr>
<td>Y</td>
<td>(defined in <code>sf::Event::Event::MouseButtonEvent</code>)</td>
<td><code>sf::Event::Event::MouseButtonEvent</code></td>
</tr>
</tbody>
</table>

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved  ::

Documentation generated by doxygen 1.5.2  ::
sf::Event::MouseMoveEvent

Member List

This is the complete list of members for 

**sf::Event::MouseMoveEvent**, including all inherited members.

| X (defined in sf::Event::MouseMoveEvent) | sf::Event::MouseMoveEvent |
| Y (defined in sf::Event::MouseMoveEvent) | sf::Event::MouseMoveEvent |

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::

Documentation generated by doxygen 1.5.2 ::
<table>
<thead>
<tr>
<th>Main Page</th>
<th>Namespaces</th>
<th>Classes</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class List</td>
<td>Class Index</td>
<td>Class Hierarchy</td>
<td>Class Members</td>
</tr>
</tbody>
</table>
sf::Event::Event::MouseWheelEvent

Member List

This is the complete list of members for sf::Event::Event::MouseWheelEvent, including all inherited members.

<table>
<thead>
<tr>
<th>Delta (defined in sf::Event::Event::MouseWheelEvent)</th>
<th>sf::Event::Event::MouseWheelEvent</th>
</tr>
</thead>
</table>

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
sf::Event::Event::SizeEvent Member List

This is the complete list of members for sf::Event::Event::SizeEvent, including all inherited members.

**Height** (defined in sf::Event::Event::SizeEvent) sf::Event::Event::SizeEvent

**Width** (defined in sf::Event::Event::SizeEvent) sf::Event::Event::SizeEvent
sf::Event::Event::TextEvent Member List

This is the complete list of members for sf::Event::Event::TextEvent, including all inherited members.

| Unicode (defined in sf::Event::Event::TextEvent) | sf::Event::Event::TextEvent |

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
//
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gomila@gmail.com)
//
// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it
// freely, subject to the following restrictions:
//
// 1. The origin of this software must not be misrepresented;
you must not claim that you wrote the original software.
// If you use this software in a product, an
// acknowledgment in the product documentation would be appreciated but
// is not required.
//
// 2. Altered source versions must be plainly marked as such,
// and must not be misrepresented as being the original software.
//
// 3. This notice may not be removed or altered.

#ifndef SFML_FONT_HPP
#define SFML_FONT_HPP

// Headers
#include <SFML/System/Resource.hpp>
#include <SFML/System/Vector2.hpp>
#include <SFML/System/Unicode.hpp>
#include <SFML/Graphics/Glyph.hpp>
#include <SFML/Graphics/Image.hpp>
#include <SFML/Graphics/Rect.hpp>
#include <map>

#endif // SFML_FONT_HPP
#include <string>

namespace sf
{
    class String;
}

namespace priv
{
    class FontLoader;
}

class SFML_API Font : public Resource<Font>
{
    public:
        Font();

        bool LoadFromFile(const std::string& Filename);
        bool LoadFromMemory(const char* Data, std::size_t SizeInBytes);
        unsigned int GetCharacterSize() const;
        const Glyph& GetGlyph(Uint32 CodePoint) const;
        const Image& GetImage() const;
        static const Font& GetDefaultFont();

    private:
        friend class priv::FontLoader;

    private:
        // Static member data
        static Uint32 ourDefaultCharset[];

        // Member data
Image myTexture;
unsigned int myCharSize;
std::map<Uint32, Glyph> myGlyphs;
};
#endif // SFML_FONT_HPP
<table>
<thead>
<tr>
<th>Main Page</th>
<th>Namespaces</th>
<th>Classes</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class List</td>
<td>Class Index</td>
<td>Class Hierarchy</td>
<td>Class Members</td>
</tr>
</tbody>
</table>
This is the complete list of members for `sf::Font`, including all inherited members.

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Font()</code></td>
<td><code>sf::Font</code></td>
</tr>
<tr>
<td><code>GetCharacterSize()</code> const</td>
<td><code>sf::Font</code></td>
</tr>
<tr>
<td><code>GetDefaultFont()</code></td>
<td><code>sf::Font</code></td>
</tr>
<tr>
<td><code>GetGlyph(Uint32 CodePoint) const</code></td>
<td><code>sf::Font</code></td>
</tr>
<tr>
<td><code>GetImage()</code> const</td>
<td><code>sf::Font</code></td>
</tr>
<tr>
<td><code>LoadFromFile(const std::string &amp;Filename, unsigned int CharSize=30, const Unicode::Text &amp;Charset=ourDefaultCharset)</code></td>
<td><code>sf::Font</code></td>
</tr>
<tr>
<td><code>LoadFromMemory(const char *Data, std::size_t SizeInBytes, unsigned int CharSize=30, const Unicode::Text &amp;Charset=ourDefaultCharset)</code></td>
<td><code>sf::Font</code></td>
</tr>
<tr>
<td><code>operator=(const Resource&lt; Font &gt; &amp;Other)</code></td>
<td><code>sf::Resource&lt; Font &gt;</code> [protected]</td>
</tr>
<tr>
<td><code>Resource()</code></td>
<td><code>sf::Resource&lt; Font &gt;</code> [protected]</td>
</tr>
<tr>
<td><code>Resource(const Resource&lt; Font &gt; &amp;Copy)</code></td>
<td><code>sf::Resource&lt; Font &gt;</code> [protected]</td>
</tr>
<tr>
<td><code>~Resource()</code></td>
<td><code>sf::Resource&lt; Font &gt;</code> [protected]</td>
</tr>
</tbody>
</table>
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gomila@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
// you must not claim that you wrote the original software.
// If you use this software in a product, an acknowledgment
// in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
// and must not be misrepresented as being the original software.

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

// Headers
#include <SFML/Graphics/Font.hpp>
#include <SFML/Graphics/FontLoader.hpp>
#include <iostream>

namespace sf
{

// Static member data
Uint32 Font::ourDefaultCharset[] =
{
// Printable characters in ASCII range
0x20, 0x21, 0x22, 0x23, 0x24, 0x25, 0x26, 0x27, 0x28, 0x29, 0x2A, 0x2B, 0x2C, 0x2D, 0x2E, 0x2F,
0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x38, 0x39, 0x3A, 0x3B, 0x3C, 0x3D, 0x3E, 0x3F,
0x40, 0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48, 0x49, 0x4A, 0x4B, 0x4C, 0x4D, 0x4E, 0x4F,
0x50, 0x51, 0x52, 0x53, 0x54, 0x55, 0x56, 0x57, 0x58, 0x59, 0x5A, 0x5B, 0x5C, 0x5D, 0x5E, 0x5F,
0x60, 0x61, 0x62, 0x63, 0x64, 0x65, 0x66, 0x67, 0x68, 0x69, 0x6A, 0x6B, 0x6C, 0x6D, 0x6E, 0x6F,
0x70, 0x71, 0x72, 0x73, 0x74, 0x75, 0x76, 0x77, 0x78, 0x79, 0x7A, 0x7B, 0x7C, 0x7D, 0x7E,

// Printable characters in extended ASCII range
0xA0, 0xA1, 0xA2, 0xA3, 0xA4, 0xA5, 0xA6, 0xA7, 0xA8, 0xA9, 0xAA, 0xAB, 0xAC, 0xAD, 0xAE, 0xAF,
0xB0, 0xB1, 0xB2, 0xB3, 0xB4, 0xB5, 0xB6, 0xB7, 0xB8, 0xB9, 0xBA, 0xBB, 0xBC, 0xBD, 0xBE, 0xBF,
0xC0, 0xC1, 0xC2, 0xC3, 0xC4, 0xC5, 0xC6, 0xC7, 0xC8, 0xC9, 0xCA, 0xCB, 0xCC, 0xCD, 0xCE, 0xCF,
0xD0, 0xD1, 0xD2, 0xD3, 0xD4, 0xD5, 0xD6, 0xD7, 0xD8, 0xD9, 0xDA, 0xDB, 0xDC, 0xDD, 0xDE, 0xDF,
0xE0, 0xE1, 0xE2, 0xE3, 0xE4, 0xE5, 0xE6, 0xE7, 0xE8, 0xE9, 0xEA, 0xEB, 0xEC, 0xED, 0xEE, 0xEF,
0xF0, 0xF1, 0xF2, 0xF3, 0xF4, 0xF5, 0xF6, 0xF7, 0xF8, 0xF9, 0xFA, 0xFB, 0xFC, 0xFD,

// To make it a valid string
0x00

Font::Font() :
myCharSize(0)
{
}

bool Font::LoadFromFile(const std::string& Filename)
{
    // Clear the previous character map
    myGlyphs.clear();

    // Always add these special characters
    Unicode::UTF32String UTFCharset = Charset;
    if (UTFCharset.find(L' ') != Unicode::UTF32String::npos)
        UTFCharset += L'
';
    if (UTFCharset.find(L'\n') != Unicode::UTF32String::npos)
        UTFCharset += L'
';

    return true;
if (UTFCharset.find(L'\v') != Unicode::UTF32String::npos)
    UTFCharset += L'\v';
if (UTFCharset.find(L'\t') != Unicode::UTF32String::npos)
    UTFCharset += L'\t';
return priv::FontLoader::GetInstance().LoadFontFromFile(Filename,
            CharSize,
            UTFCharset,
            *data);}

bool Font::LoadFromMemory(const char* Data, size_t SizeInBytes) {
    // Clear the previous character map
    myGlyphs.clear();
    // Check parameters
    if (!Data || (SizeInBytes == 0)) {
        std::cerr << "Failed to load font from memory,
                   no data provided"
        return false;
    }
    // Always add these special characters
    Unicode::UTF32String UTFCharset = Charset;
    if (UTFCharset.find(L'\t') != Unicode::UTF32String::npos)
        UTFCharset += L'\t';
    return priv::FontLoader::GetInstance().LoadFontFromMemory(Data,
                                                                        SizeInBytes,
                                                                        CharSize,
                                                                        UTFCharset,
                                                                        *data);}

unsigned int Font::GetCharacterSize() const {
    return myCharSize;
}

const Glyph& Font::GetGlyph(Uint32 CodePoint) const {
std::map<Uint32, Glyph>::const_iterator It = myGlyphs.find(CodePoint);

if (It != myGlyphs.end())
{
    // Valid glyph
    return It->second;
}
else
{
    // Invalid glyph -- return an invalid glyph
    static const Glyph InvalidGlyph;
    return InvalidGlyph;
}

const Image& Font::GetImage() const
{
    return myTexture;
}

const Font& Font::GetDefaultFont()
{
#if defined(SFML_SYSTEM_WINDOWS) && defined(SFML_DYNAMIC)

    // On Windows dynamic build, the default font causes a crash at global exit.
    // This is a temporary workaround that turns the crash into a memory leak.
    // Note that this bug doesn't exist anymore in SFML 2.
    static Font* DefaultFontPtr = new Font;
    Font& DefaultFont = *DefaultFontPtr;

#else

    static Font DefaultFont;

#endif

// Get the raw data of the Arial font file:
static const char DefaultFontData[] =
{
    #include <SFML/Graphics/Arial.hpp>
};

// Load the default font on first call
static bool DefaultFontLoaded = false;
if (!DefaultFontLoaded)
{
    DefaultFont.LoadFromMemory(DefaultFontData,
    DefaultFontLoaded = true;
}
return DefaultFont;
} // namespace sf

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
<table>
<thead>
<tr>
<th>Main Page</th>
<th>Namespaces</th>
<th>Classes</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>File List</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)
//
// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.
//
// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:
//
// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is not required.
//
// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.
//
// 3. This notice may not be removed or altered.

#ifndef SFML_FTP_HPP
#define SFML_FTP_HPP

#include <SFML/System/NonCopyable.hpp>
#include <SFML/Network/SocketTCP.hpp>
#include <string>
#include <vector>

namespace sf {
class IPAddress;

class SFML_API Ftp : NonCopyable
{
public:
enum TransferMode
{
    Binary,
    Ascii,
    Ebcdic
};

class SFML_API Response
{
public:
enum Status
{
    // 1xx: the requested action is being expected another reply before proceeding
    RestartMarkerReply = 110,
    ServiceReadySoon = 120,
    DataConnectionAlreadyOpened = 125,
    OpeningDataConnection = 150,

    // 2xx: the requested action has been completed
    Ok = 200,
    PointlessCommand = 202,
    SystemStatus = 211,
    DirectoryStatus = 212,
    FileStatus = 213,
    HelpMessage = 214,
    SystemType = 215,
    ServiceReady = 220,
    ClosingConnection = 221,
};
DataConnectionOpened = 225,
ClosingDataConnection = 226,
EnteringPassiveMode = 227,
LoggedIn = 230,
FileActionOk = 250,
DirectoryOk = 257,

// 3xx: the command has been accepted, but the requested action is dormant, pending receipt of further information
NeedPassword = 331,
NeedAccountToLogIn = 332,
NeedInformation = 350,

// 4xx: the command was not accepted but the error condition is temporary and the action may be requested again
ServiceUnavailable = 421,
DataConnectionUnavailable = 425,
TransferAborted = 426,
FileActionAborted = 450,
LocalError = 451,
InsufficientStorageSpace = 452,

// 5xx: the command was not accepted and the requested action did not take place
CommandUnknown = 500,
ParametersUnknown = 501,
CommandNotImplemented = 502,
BadCommandSequence = 503,
ParameterNotImplemented = 504,
NotLoggedIN = 530,
NeedAccountToStore = 532,
FileUnavailable = 550,
PageTypeUnknown = 551,
NotEnoughMemory = 552,
FilenameNotAllowed = 553,

// 10xx: SFML custom codes
InvalidResponse = 1000,
ConnectionFailed = 1001,
ConnectionClosed = 1002,
InvalidFile = 1003

};

Response(Status Code = InvalidResponse,
bool IsOk() const;
Status GetStatus() const;
const std::string& GetMessage() const;

private :

// Member data
Status myStatus;
std::string myMessage;

};

class SFML_API DirectoryResponse : public Response
{
public :

DirectoryResponse(Response Resp);
const std::string& GetDirectory() const;

private :

// Member data
std::string myDirectory;

};

class SFML_API ListingResponse : public Response
    public :
    
    public:
    
    ListingResponse(Response Resp, const std::vector<char>& Data);
    std::size_t GetCount() const;
    
    private:
    
    private:
    
    const std::string& GetFilename(std::size_t Index);
    
    // Member data
    std::vector<std::string> myFilenames;
};

~Ftp();

Response Connect(const IPAddress& Server, unsigned Response Login();

Response Login(const std::string& UserName, Response Disconnect();

Response KeepAlive();

Response ParentDirectory();

Response MakeDirectory(const std::string& Name);
Response DeleteDirectory(const std::string& Name);
Response RenameFile(const std::string& File,
Response DeleteFile(const std::string& Name);
Response Download(const std::string& DistantFile,
Response Upload(const std::string& LocalFile);
private :
Response SendCommand(const std::string& Command,
Response GetResponse();
class DataChannel;
friend class DataChannel;
// Member data
SocketTCP myCommandSocket;
};
} // namespace sf
#endif // SFML_FTP_HPP

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
This is the complete list of members for `sf::Ftp`, including all inherited members.

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ascii enum value</td>
<td>sf::Ftp</td>
</tr>
<tr>
<td>Binary enum value</td>
<td>sf::Ftp</td>
</tr>
<tr>
<td>ChangeDirectory(const std::string &amp;Directory)</td>
<td>sf::Ftp</td>
</tr>
<tr>
<td>Connect(const IPAddress &amp;Server, unsigned short Port=21, float Timeout=0.f)</td>
<td>sf::Ftp</td>
</tr>
<tr>
<td>DataChannel (defined in sf::Ftp)</td>
<td>sf::Ftp [friend]</td>
</tr>
<tr>
<td>DeleteDirectory(const std::string &amp;Name)</td>
<td>sf::Ftp</td>
</tr>
<tr>
<td>DeleteFile(const std::string &amp;Name)</td>
<td>sf::Ftp</td>
</tr>
<tr>
<td>Disconnect()</td>
<td>sf::Ftp</td>
</tr>
<tr>
<td>Download(const std::string &amp;DistantFile, const std::string &amp;DestPath, TransferMode Mode=Binary)</td>
<td>sf::Ftp</td>
</tr>
<tr>
<td>Ebcdic enum value</td>
<td>sf::Ftp</td>
</tr>
<tr>
<td>GetDirectoryListing(const std::string &amp;Directory=&quot;&quot;)</td>
<td>sf::Ftp</td>
</tr>
<tr>
<td>GetWorkingDirectory()</td>
<td>sf::Ftp</td>
</tr>
<tr>
<td>KeepAlive()</td>
<td>sf::Ftp</td>
</tr>
<tr>
<td>Login()</td>
<td>sf::Ftp</td>
</tr>
<tr>
<td>Login(const std::string &amp;UserName, const std::string &amp;Password)</td>
<td>sf::Ftp</td>
</tr>
<tr>
<td>MakeDirectory(const std::string &amp;Name)</td>
<td>sf::Ftp [inline, private]</td>
</tr>
<tr>
<td>NonCopyable()</td>
<td>sf::NonCopyable [inline, private]</td>
</tr>
<tr>
<td>ParentDirectory()</td>
<td>sf::Ftp</td>
</tr>
<tr>
<td>RenameFile(const std::string &amp;File, const std::string &amp;NewName)</td>
<td>sf::Ftp</td>
</tr>
<tr>
<td>TransferMode enum name</td>
<td>sf::Ftp</td>
</tr>
<tr>
<td>Upload(const std::string &amp;LocalFile, const std::string &amp;DestPath, TransferMode Mode=Binary)</td>
<td>sf::Ftp</td>
</tr>
<tr>
<td>~Ftp()</td>
<td>sf::Ftp</td>
</tr>
</tbody>
</table>
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gomila@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
// you must not claim that you wrote the original software.
// If you use this software in a product, an acknowledgment
// in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
// and must not be misrepresented as being the original software.

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

// Headers
#include <SFML/Network/Ftp.hpp>
#include <SFML/Network/IPAddress.hpp>
#include <algorithm>
#include <fstream>
#include <sstream>

namespace sf {


// Utility class for exchanging stuff with the server on the data channel
class Ftp::DataChannel : NonCopyable {
public:
    // Constructor
    DataChannel(Ftp& Owner);

    // Destructor
    ~DataChannel();

    // Open the data channel using the specified transfer mode and port
    Ftp::Response Open(Ftp::TransferMode Mode);

    // Send data on the data channel
    void Send(const std::vector<char>& Data);

    // Receive data on the data channel until it is closed
    void Receive(std::vector<char>& Data);

private:
    // Member data
    Ftp& myFtp;
    SocketTCP myDataSocket;
};

Ftp::Response::Response(Status Code, const std::string& Message) :
    myStatus(Code), myMessage(Message) {
};
bool Ftp::Response::isOk() const
{
    return myStatus < 400;
}

Ftp::Response::Status Ftp::Response::getStatus()
{
    return myStatus;
}

const std::string& Ftp::Response::getMessage()
{
    return myMessage;
}

Ftp::DirectoryResponse::DirectoryResponse(Ftp::Response Resp)
{
    if (IsOk())
    {
        // Extract the directory from the server response
        std::string::size_type Begin = Resp.getMessage();
        std::string::size_type End = Resp.getMessage();
        myDirectory = Resp.getMessage().substr(Begin + 1, End - Begin - 1);
    }
}

const std::string& Ftp::DirectoryResponse::getDirectory()
{
    return myDirectory;
}
Ftp::ListingResponse::ListingResponse(Ftp::Response Resp) {
  if (isOk()) {
    // Fill the array of strings
    std::string Paths(Data.begin(), Data.end);
    std::string::size_type LastPos = 0;
    for (std::string::size_type Pos = Paths.find(
      
      myFilenames.push_back(Paths.substr(LastPos, Pos - LastPos));
      LastPos = Pos + 2;
      }
  }
}

std::size_t Ftp::ListingResponse::GetCount() const {
  return myFilenames.size();
}

const std::string& Ftp::ListingResponse::GetFilename {
  return myFilenames[Index];
}

Ftp::~Ftp() {
  Disconnect();
}

Ftp::Response Ftp::Connect(const IPAddress& Server) {
Connect to the server

if (myCommandSocket.Connect(Port, Server, Timeout) != Socket::Done)
{
    return Response(Response::ConnectionFailed);
}

Get the response to the connection

return GetResponse();

Ftp::Response Ftp::Login()
{
    return _Login("anonymous", "user@sfml-dev.org");
}

Ftp::Response Ftp::Login(const std::string& UserName)
{
    Response Resp = SendCommand("USER", UserName);
    if (Resp.IsOk())
    { Resp = SendCommand("PASS", Password);
    return Resp;
    }
}

Ftp::Response Ftp::Disconnect()
{
    // Send the exit command
    Response Resp = SendCommand("QUIT");
    if (Resp.IsOk())
    { myCommandSocket.Close();
    return Resp;
    }
}

Ftp::Response Ftp::KeepAlive()
    { return SendCommand("NOOP"); }
    
Ftp::DirectoryResponse Ftp::GetWorkingDirectory()
    {
        return DirectoryResponse(SendCommand("PWD");
    }

Ftp::ListingResponse Ftp::GetDirectoryListing(const std::vector<char> &DirData; DataChannel Data(*this); Response Resp = Data.Open(Ascii); if (Resp.IsOk()) {
            // Tell the server to send us the listing
            Resp = SendCommand("NLST", Directory); if (Resp.IsOk()) {
                // Receive the listing
                Data.Receive(DirData);
                // Get the response from the server
                Resp = GetResponse();
            }
        }
        return ListingResponse(Resp, DirData); }
    
Ftp::Response Ftp::ChangeDirectory(const std::string& Directory) { /* implementation */ }
return SendCommand("CWD", Directory);
Ftp::Response Ftp::Download(const std::string& DistantFile) {
    // Open a data channel using the given transfer mode
    DataChannel Data(*this);
    Response Resp = Data.Open(Mode);
    if (Resp.IsOk())
    {
        // Tell the server to start the transfer
        Resp = SendCommand("RETR", DistantFile);
        if (Resp.IsOk())
        {
            // Receive the file data
            std::vector<char> FileData;
            Data.Receive(FileData);

            // Get the response from the server
            Resp = GetResponse();
            if (Resp.IsOk())
            {
                // Extract the filename from the file path
                std::string Filename = DistantFile;
                std::string::size_type Pos = Filename.find_last_of(
                    if (Pos != std::string::npos)
                    Filename = Filename.substr(Pos + 1);

                // Make sure the destination path ends with a slash
                std::string Path = DestPath;
                if (!Path.empty() && (Path[Path.size() - 1] != '/'))
                    Path += '/';

                // Create the file and copy the received data into it
                std::ofstream File((Path + Filename).c_str(), std::ios_base::binary);
                if (!File)
                    return Response(Response::InvalidFile);
                if (!FileData.empty())
                    File.write(&FileData[0], static_cast<std::streamsize>(FileData.size()));
Ftp::Response Ftp::Upload(const std::string& LocalFile)
{
    // Get the contents of the file to send
    std::ifstream File(LocalFile.c_str(), std::ios_base::binary);
    if (!File)
        return Response(Response::InvalidFile);
    File.seekg(0, std::ios::end);
    std::size_t Length = File.tellg();
    File.seekg(0, std::ios::beg);
    std::vector<char> FileData(Length);
    if (Length > 0)
        File.read(&FileData[0], static_cast<std::streamsize>(Length));

    // Extract the filename from the file path
    std::string Filename = LocalFile;
    std::string::size_type Pos = Filename.find_last_of('/');
    if (Pos != std::string::npos)
        Filename = Filename.substr(Pos + 1);

    // Make sure the destination path ends with a slash
    std::string Path = DestPath;
    if (!Path.empty() && (Path[Path.size() - 1] != '/'))
        Path += '/';

    // Open a data channel using the given transfer mode
    DataChannel Data(*this);
    Response Resp = Data.Open(Mode);
    if (Resp.IsOk())
    {

// Tell the server to start the transfer
Resp = SendCommand("STOR", Path + Filename);
if (Resp.IsOk())
{
    // Send the file data
    Data.Send(FileData);
    // Get the response from the server
    Resp = GetResponse();
}
return Resp;

Ftp::Response Ftp::SendCommand(const std::string& Command,
{
    // Build the command string
    std::string CommandStr;
    if (Parameter != "")
        CommandStr = Command + " " + Parameter + "\n";
    else
        CommandStr = Command + "\r\n";
    // Send it to the server
    if (myCommandSocket.Send(CommandStr.c_str(),
        return Response(Response::ConnectionClosed,
    // Get the response
    return GetResponse();
}

Ftp::Response Ftp::GetResponse()
{
    // We'll use a variable to keep track of the
It is useful in case of multi-lines responses, because the end of such a response will start by the same code.

```cpp
unsigned int LastCode = 0;
bool IsInsideMultiline = false;
std::string Message;

for (;;)
{
    // Receive the response from the server
    char Buffer[1024];
    std::size_t Length;
    if (myCommandSocket.Receive(Buffer, sizeof(Buffer), Length))
        return Response(Response::ConnectionClosed);

    // There can be several lines inside the received buffer
    std::istringstream In(std::string(Buffer, Length), std::ios_base::binary);
    while (In)
    {
        // Try to extract the code
        unsigned int Code;
        if (In >> Code)
        {
            // Extract the separator
            char Sep;
            In.get(Sep);

            // The '-' character means a multiline response
            if ((Sep == '-') && !IsInsideMultiline)
            {
                // Set the multiline flag
                IsInsideMultiline = true;

                // Keep track of the code
                if (LastCode == 0)
                    LastCode = Code;
        }

        // Extract the line
        // ...
std::getline(In, Message);

// Remove the ending '\r' (all lines are terminated by '\r\n')
Message.erase(Message.length() - 1);
Message = Sep + Message + '\r
'

else
{
    // We must make sure that the code is the same, otherwise it means
    // we haven't reached the end of the multiline response
    if ((Sep != '-') && ((Code == LastCode) || (LastCode == 0)))
    {
        // Clear the multiline flag
        IsInsideMultiline = false

        // Extract the line
        std::string Line;
        std::getline(In, Line);

        // Remove the ending '\r'
        Line.erase(Line.length() - 1);

        // Append it to the message
        if (Code == LastCode)
        {
            std::ostringstream Out;
            Out << Code << Sep << Line;
            Message += Out.str();
        }
        else
        {
            Message = Sep + Line;
        }
    }
    // Return the response code and message
    return Response(static_cast<Response::Status>(Code), Message);
}
else
{
    // The line we just read
    // only a new part of the
    // Extract the line
    std::string Line;
    std::getline(In, Line);

    if (!Line.empty())
    {
        // Remove the ending
        Line.erase(Line.length() - 1);

        // Append it to the
        std::ostringstream Out;
        Out << Code << Sep << Line;
        Message += Out.str();
    }
}
else if (LastCode != 0)
{
    // It seems we are in the middle
    // Clear the error bits of the
    In.clear();

    // Extract the line
    std::string Line;
    std::getline(In, Line);

    if (!Line.empty())
    {
        // Remove the ending '\r'
        Line.erase(Line.length() - 1);
    }
Append it to the current message:
Message += Line + "\n";
}
else
{
    // Error: cannot extract the code, and we are not in a multiline response
    return Response(Response::InvalidResponse);
}
}
}

// We never reach there
}

Ftp::DataChannel::DataChannel(Ftp& Owner) :
myFtp(Owner)
{
}
}

Ftp::Response Ftp::DataChannel::Open(Ftp::TransferMode
{
    // Open a data connection in active mode (we connect to the server)
    Ftp::Response Resp = myFtp.SendCommand("PASV");
    if (Resp.IsOk())
    {
        // Close the data socket
        myDataSocket.Close();
    }
}

Ftp::DataChannel::~DataChannel()
{
    // Close the data socket
    myDataSocket.Close();
}

Ftp::Response Ftp::DataChannel::Open(Ftp::TransferMode
{
    // Open a data connection in active mode (we connect to the server)
    Ftp::Response Resp = myFtp.SendCommand("PASV");
    if (Resp.IsOk())
    {
        // Close the data socket
        myDataSocket.Close();
    }
}
// Extract the connection address and port
std::string::size_type begin = Resp.GetMessage().find_first_of(
    if (begin != std::string::npos)
    {
        sf::Uint8 Data[6] = {0, 0, 0, 0, 0, 0};
        std::string Str = Resp.GetMessage().substr(begin);
        std::size_t Index = 0;
        for (int i = 0; i < 6; ++i)
        {
            // Extract the current number
            while (isdigit(Str[Index]))
            {
                Data[i] = Data[i] * 10 + (Str[Index] - Index++;
            }
            // Skip separator
            Index++;
        }
        // Reconstruct connection port and address
        unsigned short Port = Data[4] * 256 + Data[5];
        sf::IPAddress Address(static_cast<sf::Uint8>(Data[0]),
            static_cast<sf::Uint8>(Data[1]),
            static_cast<sf::Uint8>(Data[2]),
            static_cast<sf::Uint8>(Data[3]));
        // Connect the data channel to the server
        if (myDataSocket.Connect(Port, Address)
            {
                // Translate the transfer mode to the corresponding FTP parameter
                std::string ModeStr;
                switch (Mode)
                {
                    case Ftp::Binary : ModeStr = "BINARY";
                    case Ftp::Ascii : ModeStr = "ASCII";
                    case Ftp::EbcDIC : ModeStr = "EBCDIC";
                    case Ftp::Unknown : ModeStr = "UNKNOWN";
                }
            }
// Set the transfer mode
Resp = myFtp.SendCommand("TYPE", "I");
else
{
    // Failed to connect to the server
    Resp = Ftp::Response(Ftp::Response::ConnectionFailed);
}
return Resp;

void Ftp::DataChannel::Receive(std::vector<char>& Data)
{
    // Receive data
    Data.clear();
    char Buffer[1024];
    std::size_t Received;
    while (myDataSocket.Receive(Buffer, sizeof(Buffer), Received) == sf::Socket::Done)
    {
        std::copy(Buffer, Buffer + Received, std::back_inserter(Data));
    }
    // Close the data socket
    myDataSocket.Close();
}

void Ftp::DataChannel::Send(const std::vector<char>& Data)
{
    // Send data
    if (!Data.empty())
myDataSocket.Send(&Data[0], Data.size());

// Close the data socket
myDataSocket.Close();

} // namespace sf
**sf::Ftp::Ftp::DirectoryResponse Member List**

This is the complete list of members for `sf::Ftp::Ftp::DirectoryResponse`, including all inherited members.

<table>
<thead>
<tr>
<th>Member</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>BadCommandSequence</code> enum value</td>
<td><code>sf::Ftp::Ftp::Response</code></td>
</tr>
<tr>
<td><code>ClosingConnection</code> enum value</td>
<td><code>sf::Ftp::Ftp::Response</code></td>
</tr>
<tr>
<td><code>ClosingDataConnection</code> enum value</td>
<td><code>sf::Ftp::Ftp::Response</code></td>
</tr>
<tr>
<td><code>CommandNotImplemented</code> enum value</td>
<td><code>sf::Ftp::Ftp::Response</code></td>
</tr>
<tr>
<td><code>CommandUnknown</code> enum value</td>
<td><code>sf::Ftp::Ftp::Response</code></td>
</tr>
<tr>
<td><code>ConnectionClosed</code> enum value</td>
<td><code>sf::Ftp::Ftp::Response</code></td>
</tr>
<tr>
<td><code>ConnectionFailed</code> enum value</td>
<td><code>sf::Ftp::Ftp::Response</code></td>
</tr>
<tr>
<td><code>DataConnectionAlreadyOpened</code> enum value</td>
<td><code>sf::Ftp::Ftp::Response</code></td>
</tr>
<tr>
<td><code>DataConnectionOpened</code> enum value</td>
<td><code>sf::Ftp::Ftp::Response</code></td>
</tr>
<tr>
<td><code>DataConnectionUnavailable</code> enum value</td>
<td><code>sf::Ftp::Ftp::Response</code></td>
</tr>
<tr>
<td><code>DirectoryOk</code> enum value</td>
<td><code>sf::Ftp::Ftp::Response</code></td>
</tr>
<tr>
<td><code>DirectoryResponse</code>(Response Resp)</td>
<td><code>sf::Ftp::Ftp::DirectoryResponse</code></td>
</tr>
<tr>
<td><code>DirectoryStatus</code> enum value</td>
<td><code>sf::Ftp::Ftp::Response</code></td>
</tr>
<tr>
<td><code>EnteringPassiveMode</code> enum value</td>
<td><code>sf::Ftp::Ftp::Response</code></td>
</tr>
<tr>
<td><code>FileActionAborted</code> enum value</td>
<td><code>sf::Ftp::Ftp::Response</code></td>
</tr>
<tr>
<td><code>FileActionOk</code> enum value</td>
<td><code>sf::Ftp::Ftp::Response</code></td>
</tr>
<tr>
<td><code>FilenameNotAllowed</code> enum value</td>
<td><code>sf::Ftp::Ftp::Response</code></td>
</tr>
<tr>
<td><code>FileStatus</code> enum value</td>
<td><code>sf::Ftp::Ftp::Response</code></td>
</tr>
<tr>
<td><code>FileUnavailable</code> enum value</td>
<td><code>sf::Ftp::Ftp::Response</code></td>
</tr>
<tr>
<td><code>GetDirectory()</code> const</td>
<td><code>sf::Ftp::Ftp::DirectoryResponse</code></td>
</tr>
<tr>
<td><code>GetMessage()</code> const</td>
<td><code>sf::Ftp::Ftp::Response</code></td>
</tr>
<tr>
<td><code>GetStatus()</code> const</td>
<td><code>sf::Ftp::Ftp::Response</code></td>
</tr>
<tr>
<td><code>HelpMessage</code> enum value</td>
<td><code>sf::Ftp::Ftp::Response</code></td>
</tr>
<tr>
<td><code>InvalidFileStorageSpace</code> enum value</td>
<td><code>sf::Ftp::Ftp::Response</code></td>
</tr>
<tr>
<td><code>InvalidFile</code> enum value</td>
<td><code>sf::Ftp::Ftp::Response</code></td>
</tr>
<tr>
<td><code>InvalidResponse</code> enum value</td>
<td><code>sf::Ftp::Ftp::Response</code></td>
</tr>
<tr>
<td><code>IsOk()</code> const</td>
<td><code>sf::Ftp::Ftp::Response</code></td>
</tr>
<tr>
<td>Enum Name</td>
<td>Value Type</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>LocalError</td>
<td>enum value</td>
</tr>
<tr>
<td>LoggedIn</td>
<td>enum value</td>
</tr>
<tr>
<td>NeedAccountToLogIn</td>
<td>enum value</td>
</tr>
<tr>
<td>NeedAccountToStore</td>
<td>enum value</td>
</tr>
<tr>
<td>NeedInformation</td>
<td>enum value</td>
</tr>
<tr>
<td>NeedPassword</td>
<td>enum value</td>
</tr>
<tr>
<td>NotEnoughMemory</td>
<td>enum value</td>
</tr>
<tr>
<td>NotLoggedIn</td>
<td>enum value</td>
</tr>
<tr>
<td>Ok</td>
<td>enum value</td>
</tr>
<tr>
<td>OpeningDataConnection</td>
<td>enum value</td>
</tr>
<tr>
<td>PageTypeUnknown</td>
<td>enum value</td>
</tr>
<tr>
<td>ParameterNotImplemented</td>
<td>enum value</td>
</tr>
<tr>
<td>ParametersUnknown</td>
<td>enum value</td>
</tr>
<tr>
<td>PointlessCommand</td>
<td>enum value</td>
</tr>
<tr>
<td>Response (Status Code=InvalidResponse, const std::string &amp;Message=&quot;&quot;)</td>
<td>sf::Ftp::Ftp::Response</td>
</tr>
<tr>
<td>RestartMarkerReply</td>
<td>enum value</td>
</tr>
<tr>
<td>ServiceReady</td>
<td>enum value</td>
</tr>
<tr>
<td>ServiceReadySoon</td>
<td>enum value</td>
</tr>
<tr>
<td>ServiceUnavailable</td>
<td>enum value</td>
</tr>
<tr>
<td>Status</td>
<td>enum name</td>
</tr>
<tr>
<td>SystemStatus</td>
<td>enum value</td>
</tr>
<tr>
<td>SystemType</td>
<td>enum value</td>
</tr>
<tr>
<td>TransferAborted</td>
<td>enum value</td>
</tr>
</tbody>
</table>

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
sf::Ftp::Ftp::ListingResponse Member List

This is the complete list of members for sf::Ftp::Ftp::ListingResponse, including all inherited members.

<table>
<thead>
<tr>
<th>Member Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>BadCommandSequence</td>
<td>enum value sf::Ftp::Ftp::Response</td>
</tr>
<tr>
<td>ClosingConnection</td>
<td>enum value sf::Ftp::Ftp::Response</td>
</tr>
<tr>
<td>ClosingDataConnection</td>
<td>enum value sf::Ftp::Ftp::Response</td>
</tr>
<tr>
<td>CommandNotImplemented</td>
<td>enum value sf::Ftp::Ftp::Response</td>
</tr>
<tr>
<td>CommandUnknown</td>
<td>enum value sf::Ftp::Ftp::Response</td>
</tr>
<tr>
<td>ConnectionClosed</td>
<td>enum value sf::Ftp::Ftp::Response</td>
</tr>
<tr>
<td>ConnectionFailed</td>
<td>enum value sf::Ftp::Ftp::Response</td>
</tr>
<tr>
<td>DataConnectionAlreadyOpened</td>
<td>enum value sf::Ftp::Ftp::Response</td>
</tr>
<tr>
<td>DataConnectionOpened</td>
<td>enum value sf::Ftp::Ftp::Response</td>
</tr>
<tr>
<td>DataConnectionUnavailable</td>
<td>enum value sf::Ftp::Ftp::Response</td>
</tr>
<tr>
<td>DirectoryOk</td>
<td>enum value sf::Ftp::Ftp::Response</td>
</tr>
<tr>
<td>DirectoryStatus</td>
<td>enum value sf::Ftp::Ftp::Response</td>
</tr>
<tr>
<td>EnteringPassiveMode</td>
<td>enum value sf::Ftp::Ftp::Response</td>
</tr>
<tr>
<td>FileActionAborted</td>
<td>enum value sf::Ftp::Ftp::Response</td>
</tr>
<tr>
<td>FileActionOk</td>
<td>enum value sf::Ftp::Ftp::Response</td>
</tr>
<tr>
<td>FilenameNotAllowed</td>
<td>enum value sf::Ftp::Ftp::Response</td>
</tr>
<tr>
<td>FileStatus</td>
<td>enum value sf::Ftp::Ftp::Response</td>
</tr>
<tr>
<td>FileUnavailable</td>
<td>enum value sf::Ftp::Ftp::Response</td>
</tr>
<tr>
<td>GetCount() const</td>
<td>sf::Ftp::Ftp::ListingResponse</td>
</tr>
<tr>
<td>GetFilename(std::size_t Index) const</td>
<td>sf::Ftp::Ftp::ListingResponse</td>
</tr>
<tr>
<td>GetMessage() const</td>
<td>sf::Ftp::Ftp::Response</td>
</tr>
<tr>
<td>GetStatus() const</td>
<td>sf::Ftp::Ftp::Response</td>
</tr>
<tr>
<td>HelpMessage</td>
<td>enum value sf::Ftp::Ftp::Response</td>
</tr>
<tr>
<td>InsufficientStorageSpace</td>
<td>enum value sf::Ftp::Ftp::Response</td>
</tr>
<tr>
<td>InvalidFile</td>
<td>enum value sf::Ftp::Ftp::Response</td>
</tr>
<tr>
<td>InvalidResponse</td>
<td>enum value sf::Ftp::Ftp::Response</td>
</tr>
<tr>
<td>IsOk() const</td>
<td>sf::Ftp::Ftp::Response</td>
</tr>
<tr>
<td>Class/Enum</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>ListingResponse</td>
<td>(Response Resp, const std::vector&lt; char &gt; &amp;Data)</td>
</tr>
<tr>
<td>LocalError</td>
<td>enum value</td>
</tr>
<tr>
<td>LoggedIn</td>
<td>enum value</td>
</tr>
<tr>
<td>NeedAccountToLogIn</td>
<td>enum value</td>
</tr>
<tr>
<td>NeedAccountToStore</td>
<td>enum value</td>
</tr>
<tr>
<td>NeedInformation</td>
<td>enum value</td>
</tr>
<tr>
<td>NeedPassword</td>
<td>enum value</td>
</tr>
<tr>
<td>NotEnoughMemory</td>
<td>enum value</td>
</tr>
<tr>
<td>NotLoggedln</td>
<td>enum value</td>
</tr>
<tr>
<td>Ok</td>
<td>enum value</td>
</tr>
<tr>
<td>OpeningDataConnection</td>
<td>enum value</td>
</tr>
<tr>
<td>PageTypeUnknown</td>
<td>enum value</td>
</tr>
<tr>
<td>ParameterNotImplemented</td>
<td>enum value</td>
</tr>
<tr>
<td>ParametersUnknown</td>
<td>enum value</td>
</tr>
<tr>
<td>PointlessCommand</td>
<td>enum value</td>
</tr>
<tr>
<td>Response</td>
<td>(Status Code=InvalidResponse, const std::string &amp;Message=&quot;&quot;)</td>
</tr>
<tr>
<td>RestartMarkerReply</td>
<td>enum value</td>
</tr>
<tr>
<td>ServiceReady</td>
<td>enum value</td>
</tr>
<tr>
<td>ServiceReadySoon</td>
<td>enum value</td>
</tr>
<tr>
<td>ServiceUnavailable</td>
<td>enum value</td>
</tr>
<tr>
<td>Status</td>
<td>enum name</td>
</tr>
<tr>
<td>SystemStatus</td>
<td>enum value</td>
</tr>
<tr>
<td>SystemType</td>
<td>enum value</td>
</tr>
<tr>
<td>TransferAborted</td>
<td>enum value</td>
</tr>
</tbody>
</table>
This is the complete list of members for `sf::Ftp::Ftp::Response`, including all inherited members.

<table>
<thead>
<tr>
<th>Member Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>BadCommandSequence</td>
<td><code>enum value</code></td>
</tr>
<tr>
<td>ClosingConnection</td>
<td><code>enum value</code></td>
</tr>
<tr>
<td>ClosingDataConnection</td>
<td><code>enum value</code></td>
</tr>
<tr>
<td>CommandNotImplemented</td>
<td><code>enum value</code></td>
</tr>
<tr>
<td>CommandUnknown</td>
<td><code>enum value</code></td>
</tr>
<tr>
<td>ConnectionClosed</td>
<td><code>enum value</code></td>
</tr>
<tr>
<td>ConnectionFailed</td>
<td><code>enum value</code></td>
</tr>
<tr>
<td>DataConnectionAlreadyOpened</td>
<td><code>enum value</code></td>
</tr>
<tr>
<td>DataConnectionOpened</td>
<td><code>enum value</code></td>
</tr>
<tr>
<td>DataConnectionUnavailable</td>
<td><code>enum value</code></td>
</tr>
<tr>
<td>DirectoryOk</td>
<td><code>enum value</code></td>
</tr>
<tr>
<td>DirectoryStatus</td>
<td><code>enum value</code></td>
</tr>
<tr>
<td>EnteringPassiveMode</td>
<td><code>enum value</code></td>
</tr>
<tr>
<td>FileActionAborted</td>
<td><code>enum value</code></td>
</tr>
<tr>
<td>FileActionOk</td>
<td><code>enum value</code></td>
</tr>
<tr>
<td>FilenameNotAllowed</td>
<td><code>enum value</code></td>
</tr>
<tr>
<td>FileStatus</td>
<td><code>enum value</code></td>
</tr>
<tr>
<td>FileUnavailable</td>
<td><code>enum value</code></td>
</tr>
<tr>
<td>GetMessage() const</td>
<td><code>sf::Ftp::Ftp::Response</code></td>
</tr>
<tr>
<td>GetStatus() const</td>
<td><code>sf::Ftp::Ftp::Response</code></td>
</tr>
<tr>
<td>HelpMessage</td>
<td><code>sf::Ftp::Ftp::Response</code></td>
</tr>
<tr>
<td>InvalidFile</td>
<td><code>enum value</code></td>
</tr>
<tr>
<td>InsufficientStorageSpace</td>
<td><code>enum value</code></td>
</tr>
<tr>
<td>InvalidResponse</td>
<td><code>enum value</code></td>
</tr>
<tr>
<td>IsOk() const</td>
<td><code>sf::Ftp::Ftp::Response</code></td>
</tr>
<tr>
<td>LocalError</td>
<td><code>enum value</code></td>
</tr>
<tr>
<td>LoggedIn</td>
<td><code>enum value</code></td>
</tr>
<tr>
<td>NeedAccountToLogIn</td>
<td><code>enum value</code></td>
</tr>
<tr>
<td>NeedAccountToStore</td>
<td><code>enum value</code></td>
</tr>
<tr>
<td>NeedInformation</td>
<td><code>enum value</code></td>
</tr>
</tbody>
</table>
NeedPassword enum value sf::Ftp::Ftp::Response
NotEnoughMemory enum value sf::Ftp::Ftp::Response
NotLoggedIn enum value sf::Ftp::Ftp::Response
Ok enum value sf::Ftp::Ftp::Response
OpeningDataConnection enum value sf::Ftp::Ftp::Response
PageTypeUnknown enum value sf::Ftp::Ftp::Response
ParameterNotImplemented enum value sf::Ftp::Ftp::Response
ParametersUnknown enum value sf::Ftp::Ftp::Response
PointlessCommand enum value sf::Ftp::Ftp::Response
Response(Status Code=InvalidResponse, const std::string &Message="") sf::Ftp::Ftp::Response
RestartMarkerReply enum value sf::Ftp::Ftp::Response
ServiceReady enum value sf::Ftp::Ftp::Response
ServiceReadySoon enum value sf::Ftp::Ftp::Response
ServiceUnavailable enum value sf::Ftp::Ftp::Response
Status enum name sf::Ftp::Ftp::Response
SystemStatus enum value sf::Ftp::Ftp::Response
SystemType enum value sf::Ftp::Ftp::Response
TransferAborted enum value sf::Ftp::Ftp::Response
//SFML - Simple and Fast Multimedia Library
//Copyright (C) 2007-2009 Laurent Gomila (laurent.gomila@gmail.com)

//This software is provided 'as-is', without any
//express or implied warranty. In no event will the authors be held liable for
//any damages arising from the use of this software.

//Permission is granted to anyone to use this software for any purpose,
//including commercial applications, and to alter it and redistribute it freely,
subject to the following restrictions:

//1. The origin of this software must not be misrepresented;
you must not claim that you wrote the original software.
If you use this software in a product, an
acknowledgment in the product documentation would be appreciated.

//2. Altered source versions must be plainly marked as such,
and must not be misrepresented as being the original software.

//3. This notice may not be removed or altered.

#ifndef SFML_GLYPH_HPP
#define SFML_GLYPH_HPP

// Headers
#include <SFML/Config.hpp>
#include <SFML/Graphics/Rect.hpp>

namespace sf
{
class SFML_API Glyph
public:
    Glyph() : Advance(0) {}

// Member data
int Advance;
IntRect Rectangle;
FloatRect TexCoord;

};

// namespace sf

#endif // SFML_GLYPH_HPP
<table>
<thead>
<tr>
<th>Main Page</th>
<th>Namespaces</th>
<th>Classes</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Class List</td>
<td>Class Index</td>
<td>Class Hierarchy</td>
<td>Class Members</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
sf::Glyph Member List

This is the complete list of members for sf::Glyph, including all inherited members.

<table>
<thead>
<tr>
<th>Advance</th>
<th>sf::Glyph</th>
</tr>
</thead>
<tbody>
<tr>
<td>Glyph()</td>
<td>[inline]</td>
</tr>
<tr>
<td>Rectangle</td>
<td>sf::Glyph</td>
</tr>
<tr>
<td>TexCoords</td>
<td>sf::Glyph</td>
</tr>
</tbody>
</table>

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
including commercial applications, and to alter it and redistribute it freely,
subject to the following restrictions:

1. The origin of this software must not be misrepresented;
you must not claim that you wrote the original software.
If you use this software in a product, an acknowledgment
in the product documentation would be appreciated but is not required.

2. Altered source versions must be plainly marked as such,
and must not be misrepresented as being the original software.

3. This notice may not be removed or altered.

#ifndef SFML_HTTP_HPP
#define SFML_HTTP_HPP

// Headers
#include <SFML/System/NonCopyable.hpp>
#include <SFML/Network/IPAddress.hpp>
#include <SFML/Network/SocketTCP.hpp>
#include <map>
#include <string>

#endif // SFML_HTTP_HPP
namespace sf
{

class SFML_API Http : NonCopyable
{

public :

class SFML_API Request
{

public :

enum Method
{
    Get,
    Post,
    Head
};

Request(Method RequestMethod = Get,

void SetField(const std::string& Field,

void SetMethod(Method RequestMethod);

void SetURI(const std::string& URI);

void SetHttpVersion(unsigned int Major,

void SetBody(const std::string& Body);

private :

friend class Http;

std::string ToString() const;

bool HasField(const std::string& Field)
// Types
typedef std::map<std::string, std::string> FieldTable;

// Member data
FieldTable myFields;
Method myMethod;
std::string myURI;
unsigned int myMajorVersion;
unsigned int myMinorVersion;
std::string myBody;

// Member data

class SFML_API Response
{
public:
	enum Status
    {
        // 2xx: success
        Ok = 200,
        Created = 201,
        Accepted = 202,
        NoContent = 204,

        // 3xx: redirection
        MultipleChoices = 300,
        MovedPermanently = 301,
        MovedTemporarily = 302,
        NotModified = 304,

        // 4xx: client error
        BadRequest = 400,
        Unauthorized = 401,
        Forbidden = 403,
        NotFound = 404,

        // 5xx: server error
    }
};
InternalServerError = 500,
NotImplemented = 501,
BadGateway = 502,
ServiceNotAvailable = 503,

// 10xx: SFML custom codes
InvalidResponse = 1000,
ConnectionFailed = 1001

};
Response();

const std::string& GetField(const std::string& Field)
Status GetStatus() const;
unsigned int GetMajorHttpVersion() const
unsigned int GetMinorHttpVersion() const
const std::string& GetBody() const;

private :
friend class Http;

void FromString(const std::string& Data);

// Types
typedef std::map<std::string, std::string> FieldTable;

// Member data
FieldTable myFields;
Status myStatus;
unsigned int myMajorVersion;
unsigned int myMinorVersion;
std::string myBody;
Http();
Http(const std::string& Host, unsigned short);
void SetHost(const std::string& Host, unsigned short);
Response SendRequest(const Request& Req, float);

private :
  // Member data
  SocketTCP myConnection;
  IPAddress myHost;
  std::string myHostName;
  unsigned short myPort;
};

} // namespace sf

#endif // SFML_HTTP_HPP
sf::Http Member List

This is the complete list of members for sf::Http, including all inherited members.

<table>
<thead>
<tr>
<th>Function</th>
<th>Inheritance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Http()</td>
<td>sf::Http</td>
</tr>
<tr>
<td>Http(const std::string &amp;Host, unsigned short Port=0)</td>
<td>sf::Http</td>
</tr>
<tr>
<td>NonCopyable()</td>
<td>sf::NonCopyable</td>
</tr>
<tr>
<td></td>
<td>[inline, private]</td>
</tr>
<tr>
<td>SendRequest(const Request &amp;Req, float Timeout=0.f)</td>
<td>sf::Http</td>
</tr>
<tr>
<td>SetHost(const std::string &amp;Host, unsigned short Port=0)</td>
<td>sf::Http</td>
</tr>
</tbody>
</table>

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)
//
// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.
//
// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:
//
// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is not required.
// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.
// 3. This notice may not be removed or altered from any source distribution.

// Headers
#include <SFML/Network/Http.hpp>
#include <ctype.h>
#include <algorithm>
#include <iterator>
#include <sstream>

namespace
{
    // Convenience function to convert a string to lowercase
    
}
std::string ToLower(const std::string& Str) {
  std::string Ret = Str;
  for (std::string::iterator i = Ret.begin(); i != Ret.end(); ++i)
    *i = static_cast<char>(tolower(*i));
  return Ret;
}

namespace sf
{
  Http::Request::Request(Method RequestMethod, const std::string& URI) {
    SetMethod(RequestMethod);
    SetURI(URI);
    SetHttpVersion(1, 0);
    SetBody(Body);
  }

  void Http::Request::SetField(const std::string& Field, Value) {
    myFields[ToLower(Field)] = Value;
  }

  void Http::Request::SetMethod(Http::Request::Method RequestMethod) {
    myMethod = RequestMethod;
  }

  void Http::Request::SetURI(const std::string& URI) {
    myURI = URI;
  }
}
// Make sure it starts with a '/'
if (myURI.empty() || (myURI[0] != '/'))
    myURI.insert(0, "/");

void Http::Request::SetHttpVersion(unsigned int Major)
{
    myMajorVersion = Major;
    myMinorVersion = Minor;
}

void Http::Request::SetBody(const std::string& Body)
{
    myBody = Body;
}

std::string Http::Request::ToString() const
{
    std::ostringstream Out;

    // Convert the method to its string representation
    std::string RequestMethod;
    switch (myMethod)
    {
        default :
            case Get : RequestMethod = "GET"; break;
            case Post : RequestMethod = "POST"; break;
            case Head : RequestMethod = "HEAD"; break;
    }

    // Write the first line containing the request type
    Out << RequestMethod << " " << myURI << " HTTP/" << myMajorVersion << "." << 

// Write fields
for (FieldTable::const_iterator i = myFields.begin(); i != myFields.end(); ++i)
    Out << i->first << " : " << i->second << 

// Use an extra \r\n to separate the header
Out << "\r\n";

// Add the body
Out << myBody;
return Out.str();

bool Http::Request::HasField(const std::string& Field)
{
    return myFields.find(Field) != myFields.end();
}

Http::Response::Response()
    myStatus (ConnectionFailed),
    myMajorVersion(0),
    myMinorVersion(0)
{

    const std::string& Http::Response::GetField(const FieldTable::const_iterator It = myFields.find(ToLower(Field))
if (It != myFields.end())
{

return It->second;

else {
    static const std::string Empty = "";
    return Empty;
}

Http::Response::Status Http::Response::getStatus() const {
    return myStatus;
}

unsigned int Http::Response::getMajorHttpVersion() const {
    return myMajorVersion;
}

unsigned int Http::Response::getMinorHttpVersion() const {
    return myMinorVersion;
}

const std::string& Http::Response::getBody() const {
    return myBody;
}

void Http::Response::FromString(const std::string& Data) {
    std::istringstream In(Data);
std::string Version;
if (In >> Version)
{
    if ((Version.size() >= 8) && (Version[6] == Toupper(Version.substr(0, 5)) == "http/"
        isdigit(Version[5]) && isdigit(Version[7])
    {
        myMajorVersion = Version[5] - '0';
        myMinorVersion = Version[7] - '0';
    } else
    {
        // Invalid HTTP version
        myStatus = InvalidResponse;
        return;
    }
}

int StatusCode;
if (In >> StatusCode)
{
    myStatus = static_cast<Status>(StatusCode);
} else
{
    // Invalid status code
    myStatus = InvalidResponse;
    return;
}

// Ignore the end of the first line
In.ignore(10000, '\n');
std::string Line;

while (std::getline(In, Line) && (Line.size() > 2)) {
    std::string::size_type Pos = Line.find(':');
    if (Pos != std::string::npos) {
        // Extract the field name and its value
        std::string Field = Line.substr(0, Pos);
        std::string Value = Line.substr(Pos + 2);

        // Remove any trailing \r
        if (!Value.empty() && (*Value.rbegin() == '\r')) {
            Value.erase(Value.size() - 1);
        }

        // Add the field
        myFields[ToLower(Field)] = Value;
    }
}

// Finally extract the body
myBody.clear();
std::copy(std::istreambuf_iterator<char>(In),
          std::istreambuf_iterator<char>(),
          std::back_inserter(myBody));

Http::Http() :
    myHost(),
    myPort(0) {
}

Http::Http(const std::string& Host, unsigned short Port) {
    SetHost(Host, Port);
}
```cpp
void Http::SetHost(const std::string& Host, unsigned Port)
{
    // Detect the protocol used
    std::string Protocol = ToLower(Host.substr(0, 8));
    if (Protocol.substr(0, 7) == "http://")
    {
        // HTTP protocol
        myHostName = Host.substr(7);
        myPort = (Port != 0 ? Port : 80);
    }
    else if (Protocol == "https://")
    {
        // HTTPS protocol
        myHostName = Host.substr(8);
        myPort = (Port != 0 ? Port : 443);
    }
    else
    {
        // Undefined protocol - use HTTP
        myHostName = Host;
        myPort = (Port != 0 ? Port : 80);
    }

    // Remove any trailing '/\ from the host name
    if (!myHostName.empty() && (*myHostName.rbegin() == '/'))
        myHostName.erase(myHostName.size() - 1);

    myHost = sf::IPAddress(myHostName);
}

Http::Response Http::SendRequest(const Http::Request RequestToSend(Req)
{
    // First make sure the request is valid -- add missing mandatory fields
    RequestToSend(Req);
```
if (!ToSend.HasField("From"))
    { 
        ToSend.SetField("From", "user@sfml-dev.org");
    }
if (!ToSend.HasField("User-Agent"))
    { 
        ToSend.SetField("User-Agent", "libsfml-net");
    }
if (!ToSend.HasField("Host"))
    { 
        ToSend.SetField("Host", myHostName);
    }
if (!ToSend.HasField("Content-Length"))
    { 
        std::ostringstream Out;
        Out << ToSend.myBody.size();
        ToSend.SetField("Content-Length", Out.str());
    }
if ((ToSend.myMethod == Request::Post) && !)
    { 
        ToSend.SetField("Content-Type", "application/x-www-form-urlencoded");
    }
if ((ToSend.myMajorVersion * 10 + ToSend.myMinorVersion)
    { 
        ToSend.SetField("Connection", "close");
    }

// Prepare the response
Response Received;

// Connect the socket to the host
if (myConnection.Connect(myPort, myHost, Timeout)
    { 
        // Convert the request to string and send
        std::string RequestStr = ToSend.ToString;
        if (!RequestStr.empty())
{  // Send it through the socket
if (myConnection.Send(RequestStr.c_str(), RequestStr.size()) == sf::Socket::Done)
{
  // Wait for the server's response
  std::string ReceivedStr;
  std::size_t Size = 0;
  char Buffer[1024];
  while (myConnection.Receive(Buffer, Size) == sf::Socket::Done)
  {
    ReceivedStr.append(Buffer, Size);
  }
  // Build the Response object from the received data
  Received.FromString(ReceivedStr);
}
  // Close the connection
  myConnection.Close();
}
return Received; } // namespace sf
## sf::Http::Http::Request Member List

This is the complete list of members for `sf::Http::Http::Request`, including all inherited members.

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Get</strong> enum value</td>
<td><code>sf::Http::Http::Request</code></td>
</tr>
<tr>
<td><strong>Head</strong> enum value</td>
<td><code>sf::Http::Http::Request</code></td>
</tr>
<tr>
<td><strong>Http</strong> (defined in <code>sf::Http::Http::Request</code>)</td>
<td><code>sf::Http::Http::Request [friend]</code></td>
</tr>
<tr>
<td><strong>Method</strong> enum name</td>
<td><code>sf::Http::Http::Request</code></td>
</tr>
<tr>
<td><strong>Post</strong> enum value</td>
<td><code>sf::Http::Http::Request</code></td>
</tr>
<tr>
<td><strong>Request</strong>(Method RequestMethod=Get, const std::string &amp;URI=&quot;/&quot;, const std::string &amp;Body=&quot;&quot;)</td>
<td><code>sf::Http::Http::Request</code></td>
</tr>
<tr>
<td><strong>SetBody</strong>(const std::string &amp;Body)</td>
<td><code>sf::Http::Http::Request</code></td>
</tr>
<tr>
<td><strong>SetField</strong>(const std::string &amp;Field, const std::string &amp;Value)</td>
<td><code>sf::Http::Http::Request</code></td>
</tr>
<tr>
<td><strong>SetHttpVersion</strong>(unsigned int Major, unsigned int Minor)</td>
<td><code>sf::Http::Http::Request</code></td>
</tr>
<tr>
<td><strong>SetMethod</strong>(Method RequestMethod)</td>
<td><code>sf::Http::Http::Request</code></td>
</tr>
<tr>
<td><strong>SetURI</strong>(const std::string &amp;URI)</td>
<td><code>sf::Http::Http::Request</code></td>
</tr>
</tbody>
</table>
sf::Http::Http::Response Member List

This is the complete list of members for sf::Http::Http::Response, including all inherited members.

<table>
<thead>
<tr>
<th>Member</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Accepted</strong></td>
<td>enum value</td>
</tr>
<tr>
<td><strong>BadGateway</strong></td>
<td>enum value</td>
</tr>
<tr>
<td><strong>BadRequest</strong></td>
<td>enum value</td>
</tr>
<tr>
<td><strong>ConnectionFailed</strong></td>
<td>enum value</td>
</tr>
<tr>
<td><strong>Created</strong></td>
<td>enum value</td>
</tr>
<tr>
<td><strong>Forbidden</strong></td>
<td>enum value</td>
</tr>
<tr>
<td><strong>GetBody()</strong></td>
<td>const</td>
</tr>
<tr>
<td><strong>GetField(const std::string &amp;Field)</strong></td>
<td>const</td>
</tr>
<tr>
<td><strong>GetMajorHttpVersion()</strong></td>
<td>const</td>
</tr>
<tr>
<td><strong>GetMinorHttpVersion()</strong></td>
<td>const</td>
</tr>
<tr>
<td><strong>GetStatus()</strong></td>
<td>const</td>
</tr>
<tr>
<td><strong>Http</strong> (defined in sf::Http::Http::Response)</td>
<td>[friend]</td>
</tr>
<tr>
<td><strong>InternalServerError</strong></td>
<td>enum value</td>
</tr>
<tr>
<td><strong>InvalidResponse</strong></td>
<td>enum value</td>
</tr>
<tr>
<td><strong>MovedPermanently</strong></td>
<td>enum value</td>
</tr>
<tr>
<td><strong>MovedTemporarily</strong></td>
<td>enum value</td>
</tr>
<tr>
<td><strong>MultipleChoices</strong></td>
<td>enum value</td>
</tr>
<tr>
<td><strong>NoContent</strong></td>
<td>enum value</td>
</tr>
<tr>
<td><strong>NotFound</strong></td>
<td>enum value</td>
</tr>
<tr>
<td><strong>NotImplemented</strong></td>
<td>enum value</td>
</tr>
<tr>
<td><strong>NotModified</strong></td>
<td>enum value</td>
</tr>
<tr>
<td><strong>Ok</strong></td>
<td>enum value</td>
</tr>
<tr>
<td><strong>Response()</strong></td>
<td></td>
</tr>
<tr>
<td><strong>ServiceNotAvailable</strong></td>
<td>enum value</td>
</tr>
<tr>
<td><strong>Status</strong></td>
<td>enum name</td>
</tr>
<tr>
<td><strong>Unauthorized</strong></td>
<td>enum value</td>
</tr>
</tbody>
</table>
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
// you must not claim that you wrote the original software.
// If you use this software in a product, an acknowledgment
// in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
// and must not be misrepresented as being the original software.

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

#ifndef SFML_IMAGE_HPP
#define SFML_IMAGE_HPP

// Headers
#include <SFML/System/Resource.hpp>
#include <SFML/Graphics/Color.hpp>
#include <SFML/Graphics/Rect.hpp>
#include <string>
#include <vector>

#endif // SFML_IMAGE_HPP
namespace sf
{
    class RenderWindow;

    class SFML_API Image : public Resource<Image>
    {
        public:

        Image();

        Image(const Image& Copy);

        Image(unsigned int Width, unsigned int Height);

        Image(unsigned int Width, unsigned int Height);

        ~Image();

        bool LoadFromFile(const std::string& Filename);

        bool LoadFromMemory(const char* Data, std::size_t SizeInBytes);

        bool LoadFromPixels(unsigned int Width, unsigned int Height);

        bool SaveToFile(const std::string& Filename);

        bool Create(unsigned int Width, unsigned int Height);

        void CreateMaskFromColor(Color ColorKey, Uint8 Alpha = 0);

        void Copy(const Image& Source, unsigned int DestX);

        bool CopyScreen(RenderWindow& Window, const sf::Color& SourceColor = sf::Color::White);

        void SetPixel(unsigned int X, unsigned int Y, const sf::Color& Color);

        const sf::Color& GetPixel(unsigned int X, unsigned int Y);
    };
}
const Uint8* GetPixelsPtr() const;
void Bind() const;
void SetSmooth(bool Smooth);
unsigned int GetWidth() const;
unsigned int GetHeight() const;
bool IsSmooth() const;
FloatRect GetTexCoords(const IntRect& Rect);
static unsigned int GetValidTextureSize(unsigned int);
Image& operator =(const Image& Other);
private:
bool CreateTexture();
void EnsureTextureUpdate() const;
void EnsureArrayUpdate() const;
void Reset();
void DestroyTexture();

// Member data
unsigned int myWidth;
unsigned int myHeight;
unsigned int myTextureWidth;
unsigned int myTextureHeight;
unsigned int myTexture;
bool myIsSmooth;
mutable std::vector<Color> myPixels;
mutable bool myNeedTextureUpdate;
mutable bool myNeedArrayUpdate;

};

} // namespace sf

#error " SFML_IMAGE_HPP"
<table>
<thead>
<tr>
<th>Main Page</th>
<th>Namespaces</th>
<th>Classes</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class List</td>
<td>Class Index</td>
<td>Class Hierarchy</td>
<td>Class Members</td>
</tr>
</tbody>
</table>
# sf::Image Member List

This is the complete list of members for `sf::Image`, including all inherited members.

<table>
<thead>
<tr>
<th>Member Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bind() const</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Copy(const Image &amp;Source, unsigned int DestX, unsigned int DestY, const IntRect &amp;SourceRect=IntRect(0, 0, 0), bool ApplyAlpha=false)</strong></td>
<td><code>sf::Image</code></td>
</tr>
<tr>
<td><strong>CopyScreen(RenderWindow &amp;Window, const IntRect &amp;SourceRect=IntRect(0, 0, 0))</strong></td>
<td><code>sf::Image</code></td>
</tr>
<tr>
<td><strong>Create(unsigned int Width, unsigned int Height, Color Col=Color(0, 0, 0, 255))</strong></td>
<td><code>sf::Image</code></td>
</tr>
<tr>
<td><strong>CreateMaskFromColor(Color ColorKey, Uint8 Alpha=0)</strong></td>
<td><code>sf::Image</code></td>
</tr>
<tr>
<td><strong>GetHeight() const</strong></td>
<td><code>sf::Image</code></td>
</tr>
<tr>
<td><strong>GetPixel(unsigned int X, unsigned int Y) const</strong></td>
<td><code>sf::Image</code></td>
</tr>
<tr>
<td><strong>GetPixelsPtr() const</strong></td>
<td><code>sf::Image</code></td>
</tr>
<tr>
<td><strong>GetTexCoords(const IntRect &amp;Rect)</strong></td>
<td><code>sf::Image</code></td>
</tr>
<tr>
<td><strong>GetValidTextureSize(unsigned int Size)</strong></td>
<td><code>sf::Image</code> [static]</td>
</tr>
<tr>
<td><strong>GetWidth() const</strong></td>
<td><code>sf::Image</code></td>
</tr>
<tr>
<td><strong>Image(const Image &amp;Copy)</strong></td>
<td><code>sf::Image</code></td>
</tr>
<tr>
<td><strong>Image(unsigned int Width, unsigned int Height, const Color &amp;Col=Color(0, 0, 0, 255))</strong></td>
<td><code>sf::Image</code></td>
</tr>
<tr>
<td>*<em>Image(unsigned int Width, unsigned int Height, const Uint8 <em>Data)</em></em></td>
<td><code>sf::Image</code></td>
</tr>
<tr>
<td><strong>IsSmooth() const</strong></td>
<td><code>sf::Image</code></td>
</tr>
<tr>
<td><strong>LoadFromFile(const std::string &amp;Filename)</strong></td>
<td><code>sf::Image</code></td>
</tr>
<tr>
<td>*<em>LoadFromMemory(const char <em>Data, std::size_t SizeInBytes)</em></em></td>
<td><code>sf::Image</code></td>
</tr>
<tr>
<td>*<em>LoadFromPixels(unsigned int Width, unsigned int Height, const Uint8 <em>Data)</em></em></td>
<td><code>sf::Image</code></td>
</tr>
<tr>
<td><strong>operator=(const Image &amp;Other)</strong></td>
<td><code>sf::Image</code></td>
</tr>
<tr>
<td><strong>Resource&lt; Image &gt;::operator=(const Resource&lt; Image &gt; &amp;Other)</strong></td>
<td><code>sf::Resource&lt; Image &gt;</code> [protected]</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td><code>Resource()</code></td>
<td>sf::Resource&lt;Image&gt; [protected]</td>
</tr>
<tr>
<td><code>Resource(const Resource&lt;Image&gt; &amp;Copy)</code></td>
<td>sf::Resource&lt;Image&gt; [protected]</td>
</tr>
<tr>
<td><code>SaveToFile(const std::string &amp;Filename) const</code></td>
<td>sf::Image</td>
</tr>
<tr>
<td><code>SetPixel(unsigned int X, unsigned int Y, const Color &amp;Col)</code></td>
<td>sf::Image</td>
</tr>
<tr>
<td><code>SetSmooth(bool Smooth)</code></td>
<td>sf::Image</td>
</tr>
<tr>
<td><code>~Image()</code></td>
<td>sf::Image</td>
</tr>
<tr>
<td><code>~Resource()</code></td>
<td>sf::Resource&lt;Image&gt; [protected]</td>
</tr>
</tbody>
</table>

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
including commercial applications, and to alter it and redistribute it freely,
subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
you must not claim that you wrote the original software.
If you use this software in a product, an acknowledgment
in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
and must not be misrepresented as being the original software.

// 3. This notice may not be removed or altered.

// Headers
#include <SFML/Graphics/Image.hpp>
#include <SFML/Graphics/ImageLoader.hpp>
#include <SFML/Graphics/RenderWindow.hpp>
#include <SFML/Graphics/GraphicsContext.hpp>
#include <algorithm>
#include <iostream>
#include <vector>
#include <string.h>
namespace sf
{

Image::Image():
  myWidth (0),
  myHeight (0),
  myTextureWidth (0),
  myTextureHeight (0),
  myTexture (0),
  myIsSmooth (true),
  myNeedTextureUpdate(false),
  myNeedArrayUpdate (false)
{

}

Image::Image(const Image& Copy):
  Resource<Image> (Copy),
  myWidth (Copy.myWidth),
  myHeight (Copy.myHeight),
  myTextureWidth (Copy.myTextureWidth),
  myTextureHeight (Copy.myTextureHeight),
  myTexture (0),
  myIsSmooth (Copy.myIsSmooth),
  myPixels (Copy.myPixels),
  myNeedTextureUpdate(false),
  myNeedArrayUpdate (false)
{
    CreateTexture();
  }

Image::Image(unsigned int Width, unsigned int Height):
  myWidth (0),
  myHeight (0),
  myTextureWidth (0),
  myTextureHeight (0),
Image::Image(unsigned int Width, unsigned int Height) {
    myWidth = 0;
    myHeight = 0;
    myTextureWidth = 0;
    myTextureHeight = 0;
    myTexture = 0;
    myIsSmooth = true;
    myNeedTextureUpdate = false;
    myNeedArrayUpdate = false;
    
    Create(Width, Height, Col);
}

Image::~Image()
{
    // Destroy the OpenGL texture
    DestroyTexture();
}

bool Image::LoadFromFile(const std::string& Filename)
{
    // Let the image loader load the image into our pixel array
    bool Success = priv::ImageLoader::GetInstance().LoadImageFromFile(Filename, myPixels, myWidth, myHeight);
    if (Success) {
        
    }
    return Success;
}
// Loading succeeded: we can create the texture
if (CreateTexture())
    return true;

// Oops... something failed
Reset();

return false;

bool Image::LoadFromMemory(const char* Data, std::size_t SizeInBytes)
{
    // Check parameters
    if (!Data || (SizeInBytes == 0))
    {
        std::cerr << "Failed to image font from memory, no data provided"
        return false;
    }

    // Let the image loader load the image into our pixel array
    bool Success = priv::ImageLoader::GetInstance().LoadImageFromMemory(Data, SizeInBytes, myPixels, myWidth, myHeight);

    if (Success)
    {
        // Loading succeeded: we can create the texture
        if (CreateTexture())
            return true;
    }

    // Oops... something failed
    Reset();

    return false;
}
bool Image::LoadFromPixels(unsigned int Width, unsigned int Height) {
    if (Data)
    {
        // Store the texture dimensions
        myWidth = Width;
        myHeight = Height;

        // Fill the pixel buffer with the specified raw data
        const Color* Ptr = reinterpret_cast<const Color*>(Data);
        myPixels.assign(Ptr, Ptr + Width * Height);

        // We can create the texture
        if (CreateTexture())
        {
            return true;
        }
        else
        {
            // Oops... something failed
            Reset();
            return false;
        }
    }
    else
    {
        // No data provided: create a white image
        return Create(Width, Height, Color(255, 255, 255, 255));
    }
}

bool Image::SaveToFile(const std::string& Filename) {
    // Check if the array of pixels needs to be updated
    EnsureArrayUpdate();
// Let the image loader save our pixel array
return priv::ImageLoader::GetInstance().SaveImageToFile(Filename, myPixels, myWidth, myHeight);

bool Image::Create(unsigned int Width, unsigned int Height) {
    // Store the texture dimensions
    myWidth = Width;
    myHeight = Height;
    // Recreate the pixel buffer and fill it with the specified color
    myPixels.clear();
    myPixels.resize(Width * Height, Col);
    // We can create the texture
    if (CreateTexture())
        return true;
    else
    {
        // Oops... something failed
        Reset();
        return false;
    }
}

void Image::CreateMaskFromColor(Color ColorKey, Uint8 Alpha) {
    // Check if the array of pixels needs to be updated
    EnsureArrayUpdate();
    // Calculate the new color (old color with no alpha)
    Color NewColor(ColorKey.r, ColorKey.g, ColorKey.b, Alpha);
}
// Replace the old color by the new one
std::replace(myPixels.begin(), myPixels.end(), ColorKey, NewColor);

// The texture will need to be updated
myNeedTextureUpdate = true;

void Image::Copy(const Image& Source, unsigned int)
{
    // Make sure both images are valid
    if ((Source.myWidth == 0) || (Source.myHeight == 0))
        return;

    // Make sure both images have up-to-date arrays
    EnsureArrayUpdate();
    Source.EnsureArrayUpdate();

    // Adjust the source rectangle
    IntRect SrcRect = SourceRect;
    if (SrcRect.GetWidth() == 0 || (SrcRect.GetHeight() == 0))
    {
        SrcRect.Left = 0;
        SrcRect.Top = 0;
        SrcRect.Right = Source.myWidth;
        SrcRect.Bottom = Source.myHeight;
    }
    else
    {
        if (SrcRect.Left < 0) SrcRect.Left = 0;
        if (SrcRect.Top < 0) SrcRect.Top = 0;
        if (SrcRect.Right > static_cast<int>(Source.myWidth))
            SrcRect.Right = static_cast<int>(Source.myWidth);
        if (SrcRect.Bottom > static_cast<int>(Source.myHeight))
            SrcRect.Bottom = static_cast<int>(Source.myHeight);
    }

    // Then find the valid bounds of the destination

int Width = SrcRect.GetWidth();
int Height = SrcRect.GetHeight();
if (DestX + Width > myWidth) Width = myWidth - DestX;
if (DestY + Height > myHeight) Height = myHeight - DestY;

// Make sure the destination area is valid
if ((Width <= 0) || (Height <= 0))
    return;

// Precompute as much as possible
int Pitch = Width * 4;
int Rows = Height;
int SrcStride = Source.myWidth * 4;
int DstStride = myWidth * 4;
const Uint8* SrcPixels = Source.GetPixelsPtr();
Uint8* DstPixels = reinterpret_cast<Uint8*>(Source.GetPixelsPtr());

// Copy the pixels
if (ApplyAlpha)
{
    // Interpolation using alpha values, pixel by pixel
    for (int i = 0; i < Rows; ++i)
    {
        for (int j = 0; j < Width; ++j)
        {
            // Get a direct pointer to the components
            const Uint8* Src = SrcPixels + j * 4;
            Uint8* Dst = DstPixels + j * 4;

            // Interpolate RGB components using the alpha value of the source pixel
            Uint8 Alpha = Src[3];
            Dst[0] = (Src[0] * Alpha + Dst[0] * (255 - Alpha)) / 255;
        }
        SrcPixels += SrcStride;
    }
}
// Adjust the source rectangle
IntRect SrcRect = SourceRect;
if (SrcRect.GetWidth() == 0 || (SrcRect.GetHeight() == 0))
{
    SrcRect.Left = 0;
    SrcRect.Top = 0;
    SrcRect.Right = Window.GetWidth();
    SrcRect.Bottom = Window.GetHeight();
}
else
{
    if (SrcRect.Left < 0) SrcRect.Left = 0;
    if (SrcRect.Top < 0) SrcRect.Top = 0;
    if (SrcRect.Right > static_cast<int>(Window.GetWidth()))
        if (SrcRect.Bottom > static_cast<int>(Window.GetHeight()))
            }
// Store the texture dimensions
myWidth = SrcRect.GetWidth();
myHeight = SrcRect.GetHeight();

// We can then create the texture
if (Window.SetActive() && CreateTexture())
{
    GLint PreviousTexture;
    GLCheck(glGetIntegerv(GL_TEXTURE_BINDING_2D, &PreviousTexture));
    GLCheck(glBindTexture(GL_TEXTURE_2D, myTexture));
    GLCheck(glCopyTexSubImage2D(GL_TEXTURE_2D, 0, 0, 0, SrcRect.
        GLCheck(glBindTexture(GL_TEXTURE_2D, PreviousTexture));
    myNeedTextureUpdate = false;
    myNeedArrayUpdate = true;
    return true;
}
else
{
    Reset();
    return false;
}

void Image::SetPixel(unsigned int X, unsigned int Y)
{
    // First check if the array of pixels needs
    EnsureArrayUpdate();

    // Check if pixel is within the image bounds
    if ((X >= myWidth) || (Y >= myHeight))
    {
        //
std::cerr << "Cannot set pixel (" << X << "(width = " << myWidth << ")
return;
}

myPixels[X + Y * myWidth] = Col;

// The texture will need to be updated
myNeedTextureUpdate = true;

const Color& Image::GetPixel(unsigned int X, unsigned int Y) {
    // First check if the array of pixels needs to be updated
    EnsureArrayUpdate();

    // Check if pixel is within the image bounds
    if ((X >= myWidth) || (Y >= myHeight))
    {
        std::cerr << "Cannot get pixel (" << X << "(width = " << myWidth << ")
        return Color::Black;
    }

    return myPixels[X + Y * myWidth];
}

const Uint8* Image::GetPixelsPtr() const
{
    // First check if the array of pixels needs to be updated
    EnsureArrayUpdate();

    if (!myPixels.empty())
    {
        return reinterpret_cast<const Uint8*>(

}
else {
    std::cerr << "Trying to access the pixels of an empty image"
    return NULL;
} }

void Image::Bind() const {
    // First check if the texture needs to be updated
    EnsureTextureUpdate();

    // Bind it
    if (myTexture)
    {
        GLCheck(glEnable(GL_TEXTURE_2D));
        GLCheck(glBindTexture(GL_TEXTURE_2D, myTexture));
    }
}

void Image::SetSmooth(bool Smooth) {
    if (Smooth != myIsSmooth)
    {
        myIsSmooth = Smooth;

        if (myTexture)
        {
            // Make sure we have a valid context
            priv::GraphicsContext Ctx;

            GLint PreviousTexture;
            GLCheck(glGetIntegeriv(GL_TEXTURE_BINDING_2D, &PreviousTexture));
        }
    }
}
GLCheck(glBindTexture(GL_TEXTURE_2D, myTexture));

GLCheck(glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, myIsSmooth ? GL_LINEAR : GL_NEAREST));

GLCheck(glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, myIsSmooth ? GL_LINEAR : GL_NEAREST));

GLCheck(glBindTexture(GL_TEXTURE_2D, PreviousTexture));

}
}
}

unsigned int Image::GetWidth() const
{
    return myWidth;
}

unsigned int Image::GetHeight() const
{
    return myHeight;
}

bool Image::IsSmooth() const
{
    return myIsSmooth;
}

FloatRect Image::GetTexCoords(const IntRect& Rect)
{
    float Width = static_cast<float>(myTextureWidth);
    float Height = static_cast<float>(myTextureHeight);

    return FloatRect(Rect.Left / Width, Rect.Top / Height, Rect.Right / Width, Rect.Bottom / Height);
unsigned int Image::GetValidTextureSize(unsigned int Size) {
    // Make sure we have a valid context
    priv::GraphicsContext Ctx;
    if (glewIsSupported("GL_ARB_texture_non_power_of_two")
        {
        // If hardware supports NPOT textures, then just return the unmodified size
        return Size;
    }
    else
    {
        // If hardware doesn't support NPOT textures, we calculate the nearest power of two
        unsigned int PowerOfTwo = 1;
        while (PowerOfTwo < Size)
        PowerOfTwo *= 2;
        return PowerOfTwo;
    }
}

Image& Image::operator=(const Image& Other)
{
    Image Temp(Other);
    std::swap(myWidth, Temp.myWidth);
    std::swap(myHeight, Temp.myHeight);
    std::swap(myTextureWidth, Temp.myTextureWidth);
    std::swap(myTextureHeight, Temp.myTextureHeight);
    std::swap(myTexture, Temp.myTexture);
    std::swap(myIsSmooth, Temp.myIsSmooth);
    std::swap(myNeedArrayUpdate, Temp.myNeedArrayUpdate);
    std::swap(myNeedTextureUpdate, Temp.myNeedTextureUpdate);
}
myPixels.swap(Temp.myPixels);

return *this;

} // Check if texture parameters are valid before creating it
if (!myWidth || !myHeight)
    return false;

// Make sure we have a valid context
priv::GraphicsContext Ctx;

// Adjust internal texture dimensions depending on NPOT textures support
unsigned int TextureWidth = GetValidTextureSize();
unsigned int TextureHeight = GetValidTextureSize();

// Check the maximum texture size
GLint MaxSize;
GLCheck(glGetIntegerv(GL_MAX_TEXTURE_SIZE, &MaxSize));
if ((TextureWidth > static_cast<unsigned int>(MaxSize)) || (TextureHeight > static_cast<unsigned int>(MaxSize)))
    {
        std::cerr << "Failed to create image, its internal size is too high (";
        return false;
    }

// Destroy the previous OpenGL texture if it already exists with another size
if ((TextureWidth != myTextureWidth) || (TextureHeight != myTextureHeight))
    {
        DestroyTexture();
        myTextureWidth = TextureWidth;
        myTextureHeight = TextureHeight;
    }

// Create the OpenGL texture
if (!myTexture)
{
    GLint PreviousTexture;
    GLCheck(glGetIntegerv(GL_TEXTURE_BINDING_2D, &PreviousTexture));
    GLuint Texture = 0;
    GLCheck(glGenTextures(1, &Texture));
    GLCheck(glBindTexture(GL_TEXTURE_2D, Texture));
    GLCheck(glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA8, myTextureWidth, myTextureHeight, 0, GL_RGBA, GL_UNSIGNED_BYTE, NULL));
    GLCheck(glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP));
    GLCheck(glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP));
    GLCheck(glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, myIsSmooth ? GL_LINEAR : GL_NEAREST));
    GLCheck(glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, myIsSmooth ? GL_LINEAR : GL_NEAREST));
    myTexture = static_cast<unsigned int>(Texture);
    GLCheck(glBindTexture(GL_TEXTURE_2D, PreviousTexture));
}
myNeedTextureUpdate = true;
return true;
}

void Image::EnsureTextureUpdate() const
{
    if (myNeedTextureUpdate)
    {
        // Copy the pixels
        if (myTexture && !myPixels.empty())
        {
            GLint PreviousTexture;
            GLCheck(glGetIntegerv(GL_TEXTURE_BINDING_2D, &PreviousTexture));
            // Update the texture with the pixel array
            GLCheck(glBindTexture(GL_TEXTURE_2D, myTexture);
            GLCheck(glTexSubImage2D(GL_TEXTURE_2D, 0, 0, 0, myWidth, myHeight, GL_RGBA, GL_UNSIGNED_BYTE, &myPixels[0]));
        
        
        
    }
GLCheck(glBindTexture(GL_TEXTURE_2D, PreviousTexture));
{
    myNeedTextureUpdate = false;
}
}

void Image::EnsureArrayUpdate() const
{
    if (myNeedArrayUpdate)
    {
        // Save the previous texture
        GLint PreviousTexture;
        GLCheck(glGetIntegerv(GL_TEXTURE_BINDING_2D, &PreviousTexture));

        // Resize the destination array of pixels
        myPixels.resize(myWidth * myHeight);

        if ((myWidth == myTextureWidth) && (myHeight == myTextureHeight))
        {
            // Texture and array have the same size, we can use a direct copy
            // Copy pixels from texture to array
            GLCheck(glBindTexture(GL_TEXTURE_2D, myTexture));
            GLCheck(glGetTexImage(GL_TEXTURE_2D, 0, GL_RGBA, GL_UNSIGNED_BYTE, &myPixels[0]));
        }
        else
        {
            // Texture and array don't have the same size, we have to use a slower algorithm
            // All the pixels will first be copied to a temporary array
            std::vector<Color> AllPixels(myTextureWidth * myTextureHeight);
            GLCheck(glBindTexture(GL_TEXTURE_2D, myTexture));
            GLCheck(glGetTexImage(GL_TEXTURE_2D, 0, GL_RGBA, GL_UNSIGNED_BYTE, &AllPixels[0]));
        }
    }
}
The we copy the useful pixels from the temporary array to the final one.

```cpp
const Color* Src = &AllPixels[0];
Color* Dst = &myPixels[0];
for (unsigned int i = 0; i < myHeight; )
{
    std::copy(Src, Src + myWidth, Dst);
    Src += myTextureWidth;
    Dst += myWidth;
}
```

// Restore the previous texture
GLCheck(glBindTexture(GL_TEXTURE_2D, PreviousTexture));
myNeedArrayUpdate = false;
}

void Image::Reset()
{
    DestroyTexture();
    myWidth = 0;
    myHeight = 0;
    myTextureWidth = 0;
    myTextureHeight = 0;
    myTexture = 0;
    myIsSmooth = true;
    myNeedTextureUpdate = false;
    myNeedArrayUpdate = false;
    myPixels.clear();
}

void Image::DestroyTexture()
{
// Destroy the internal texture
if (myTexture)
{
    // Make sure we have a valid context
    priv::GraphicsContext Ctx;

    GLuint Texture = static_cast<GLuint>(myTexture);
    GLCheck(glDeleteTextures(1, &Texture));
    myTexture = 0;
    myNeedTextureUpdate = false;
    myNeedArrayUpdate = false;
}
} // namespace sf
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)
// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.
// Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:
// 1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.
// 2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.
// 3. This notice may not be removed or altered.

#ifndef SFML_INPUT_HPP
#define SFML_INPUT_HPP

// Headers
#include <SFML/Config.hpp>
#include <SFML/System/NonCopyable.hpp>
#include <SFML/Window/Event.hpp>
#include <SFML/Window/WindowListener.hpp>

namespace sf {


```cpp
class SFML_API Input : public WindowListener, NonCopyable
{

public:

    Input();

    bool IsKeyDown(Key::Code KeyCode) const;

    bool IsMouseButtonDown(Mouse::Button Button);

    bool IsJoystickButtonDown(unsigned int JoyId);

    int GetMouseX() const;

    int GetMouseY() const;

    float GetJoystickAxis(unsigned int JoyId, Joy::Axis Axis);

private:

    virtual void OnEvent(const Event& EventReceived);

    void ResetStates();

    // Member data

    bool myKeys[Key::Count];
    bool myMouseButtons[Mouse::ButtonCount];
    int myMouseX;
    int myMouseY;
    bool myJoystickButtons[Joy::Count][Joy::ButtonCount];
    float myJoystickAxis[Joy::Count][Joy::AxisCount];

};

} // namespace sf
```
# sf::Input Member List

This is the complete list of members for sf::Input, including all inherited members.

<table>
<thead>
<tr>
<th>Function</th>
<th>Class</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetJoystickAxis(unsigned int JoyId, Joy::Axis Axis) const</td>
<td>sf::Input</td>
<td></td>
</tr>
<tr>
<td>GetMouseX() const</td>
<td>sf::Input</td>
<td></td>
</tr>
<tr>
<td>GetMouseY() const</td>
<td>sf::Input</td>
<td></td>
</tr>
<tr>
<td>Input()</td>
<td>sf::Input</td>
<td></td>
</tr>
<tr>
<td>IsJoystickButtonDown(unsigned int JoyId, unsigned int Button) const</td>
<td>sf::Input</td>
<td></td>
</tr>
<tr>
<td>IsKeyDown(Key::Code KeyCode) const</td>
<td>sf::Input</td>
<td></td>
</tr>
<tr>
<td>IsMouseButtonDown(Mouse::Button Button) const</td>
<td>sf::Input</td>
<td></td>
</tr>
<tr>
<td>NonCopyable()</td>
<td>sf::NonCopyable</td>
<td>[inline, private]</td>
</tr>
<tr>
<td>~WindowListener()</td>
<td>sf::WindowListener</td>
<td>[inline, protected, virtual]</td>
</tr>
<tr>
<td>Main Page</td>
<td>Namespaces</td>
<td>Classes</td>
</tr>
<tr>
<td>-----------</td>
<td>------------</td>
<td>---------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
// you must not claim that you wrote the original software.
// If you use this software in a product, an acknowledgment
// in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
// and must not be misrepresented as being the original software.

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

// Headers
#include <SFML/Window/Input.hpp>

namespace sf {

Input::Input() :
    myMouseX(0),
    myMouseY(0)
{
    ResetStates();
bool Input::IsKeyDown(Key::Code KeyCode) const
{
    return myKeys[KeyCode];
}

bool Input::IsMouseButtonDown(Mouse::Button Button)
{
    return myMouseButtons[Button];
}

bool Input::IsJoystickButtonDown(unsigned int JoyId, Joy::Axis Axis)
{
    if ((JoyId < Joy::Count) && (Button < Joy::ButtonCount))
    {
        return myJoystickButtons[JoyId][Axis];
    } else
    {
        return false;
    }
}

int Input::GetMouseX() const
{
    return myMouseX;
}

int Input::GetMouseY() const
{
    return myMouseY;
}

float Input::GetJoystickAxis(unsigned int JoyId,
if (JoyId < Joy::Count)
    return myJoystickAxis[JoyId][Axis];
else
    return 0.f;
}

void Input::OnEvent(const Event& EventReceived)
{
    switch (EventReceived.Type)
    {
    // Key events
    case Event::KeyPressed:
        myKeys[EventReceived.Key] = 1;
        break;
    case Event::KeyReleased:
        myKeys[EventReceived.Key] = 0;
        break;
    // Mouse event
    case Event::MouseButtonPressed:
        myMouseButtons[EventReceived.Button] = 1;
        break;
    case Event::MouseButtonReleased:
        myMouseButtons[EventReceived.Button] = 0;
        break;
    // Mouse move event
    case Event::MouseMoved:
        myMouseX = EventReceived.MouseMove.X;
        myMouseY = EventReceived.MouseMove.Y;
        break;
    // Joystick button events
    case Event::JoyButtonPressed:
        myJoystickButtons[EventReceived.JoyId][EventReceived.Button] = 1;
        break;
    case Event::JoyButtonReleased:
        myJoystickButtons[EventReceived.JoyId][EventReceived.Button] = 0;
        break;
    // Joystick move event
    case Event::JoyMoved:
        myJoystickAxis[EventReceived.JoyId][EventReceived.Axis] = EventReceived.JoyMove;
        break;
    // Lost focus event: we must reset all persistent states
    case Event::LostFocus:
        break;
    }
```cpp
void Input::ResetStates()
{
    for (int i = 0; i < Key::Count; ++i)
        myKeys[i] = false;

    for (int i = 0; i < Mouse::ButtonCount; ++i)
        myMouseButtons[i] = false;

    for (int i = 0; i < Joy::Count; ++i)
    {
        for (int j = 0; j < Joy::ButtonCount; ++j)
            myJoystickButtons[i][j] = false;
        for (int j = 0; j < Joy::AxisCount; ++j)
            myJoystickAxis[i][j] = 0.f;
        myJoystickAxis[i][Joy::AxisPOV] = -1.f;
    }

    // namespace sf
}
```
IPAddress.hpp

#ifndef SFML_IPADDRESS_HPP
#define SFML_IPADDRESS_HPP

// Headers
#include <SFML/Config.hpp>
#include <istream>
#include <ostream>
#include <string>

namespace sf

class SFML_API IPAddress
{
public:
    IPAddress();
    IPAddress(const std::string& Address);
    IPAddress(const char* Address);
    IPAddress(Uint8 Byte0, Uint8 Byte1, Uint8 Byte2, Uint8 Byte3);
    IPAddress(Uint32 Address);
    bool IsValid() const;
    std::string ToString() const;
    Uint32 ToInteger() const;
    static IPAddress GetLocalAddress();
    static IPAddress GetPublicAddress(float Timeout = 0.f);
    bool operator == (const IPAddress& Other) const;
    bool operator != (const IPAddress& Other) const;
    bool operator < (const IPAddress& Other) const;
    bool operator > (const IPAddress& Other) const;
    bool operator <= (const IPAddress& Other) const;
    bool operator >= (const IPAddress& Other) const;
// Static member data
static const IPAddress LocalHost;

private :

// Member data
Uint32 myAddress;

};

SFML_API std::istream& operator>>(std::istream& Stream,
SFML_API std::ostream& operator<<(std::ostream& Stream,
} // namespace sf

# endif // SFML_IPADDRESS_HPP

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
sf::IPAddress Member List

This is the complete list of members for `sf::IPAddress`, including all inherited members.

<table>
<thead>
<tr>
<th>Function</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetLocalAddress()</td>
<td>sf::IPAddress</td>
</tr>
<tr>
<td>GetPublicAddress(float Timeout=0.f)</td>
<td>sf::IPAddress</td>
</tr>
<tr>
<td>IPAddress()</td>
<td>sf::IPAddress</td>
</tr>
<tr>
<td>IPAddress(const std::string &amp;Address)</td>
<td>sf::IPAddress</td>
</tr>
<tr>
<td>IPAddress(const char *Address)</td>
<td>sf::IPAddress</td>
</tr>
<tr>
<td>IPAddress(Uint8 Byte0, Uint8 Byte1, Uint8 Byte2, Uint8 Byte3)</td>
<td>sf::IPAddress</td>
</tr>
<tr>
<td>IPAddress(Uint32 Address)</td>
<td>sf::IPAddress</td>
</tr>
<tr>
<td>IsValid() const</td>
<td>sf::IPAddress</td>
</tr>
<tr>
<td>LocalHost</td>
<td>sf::IPAddress</td>
</tr>
<tr>
<td>operator!=(const IPAddress &amp;Other) const</td>
<td>sf::IPAddress</td>
</tr>
<tr>
<td>operator&lt;(const IPAddress &amp;Other) const</td>
<td>sf::IPAddress</td>
</tr>
<tr>
<td>operator&lt;=(const IPAddress &amp;Other) const</td>
<td>sf::IPAddress</td>
</tr>
<tr>
<td>operator==(const IPAddress &amp;Other) const</td>
<td>sf::IPAddress</td>
</tr>
<tr>
<td>operator&gt;=(const IPAddress &amp;Other) const</td>
<td>sf::IPAddress</td>
</tr>
<tr>
<td>operator&gt;(const IPAddress &amp;Other) const</td>
<td>sf::IPAddress</td>
</tr>
<tr>
<td>operator==(const IPAddress &amp;Other) const</td>
<td>sf::IPAddress</td>
</tr>
<tr>
<td>ToInteger() const</td>
<td>sf::IPAddress</td>
</tr>
<tr>
<td>ToString() const</td>
<td>sf::IPAddress</td>
</tr>
</tbody>
</table>

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved  ::
Documentation generated by doxygen 1.5.2  ::
IPAddress.cpp

00001 //
00002 // SFML - Simple and Fast Multimedia Library
00003 // Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)
00004 //
00005 // This software is provided 'as-is', without any
00006 // In no event will the authors be held liable for
00007 // In no event will the authors be held liable for any
00008 //
00009 // Permission is granted to anyone to use this software
00010 // including commercial applications, and to alter it
00011 // subject to the following restrictions:
00012 //
00013 // 1. The origin of this software must not be
00014 // you must not claim that you wrote the original
00015 // If you use this software in a product, and
00016 // in the product documentation would be appreciated.
00017 //
00018 // 2. Altered source versions must be plainly marked
00019 // and must not be misrepresented as being the
00020 //
00021 // 3. This notice may not be removed or altered.
00022 //
00023 // Headers
00024 #include <SFML/Network/IPAddress.hpp>
00025 #include <SFML/Network/Http.hpp>
00026 #include <SFML/Network/SocketHelper.hpp>
00027 #include <string.h>
00028
00029 namespace sf {
00030
00031 const IPAddress IPAddress::LocalHost("127.0.0.1")
00032
00033
00034 namespace sf
00035 {
00036 const IPAddress IPAddress::LocalHost("127.0.0.1")
00037 }
IPAddress::IPAddress() :
    myAddress(INADDR_NONE)
{
    }

IPAddress::IPAddress(const std::string& Address)
{
    // First try to convert it as a byte representation
    myAddress = inet_addr(Address.c_str());

    // If not successful, try to convert it as a host name
    if (!IsValid())
    {
        hostent* Host = gethostbyname(Address.c_str());
        if (Host)
        {
            // Host found, extract its IP address
            myAddress = reinterpret_cast<in_addr*>(Host->h_addr);
        }
        else
        {
            // Host name not found on the network
            myAddress = INADDR_NONE;
        }
    }

    }

IPAddress::IPAddress(const char* Address)
{
    // First try to convert it as a byte representation
    myAddress = inet_addr(Address);

    // If not successful, try to convert it as a
if (!IsValid())
{
    hostent* Host = gethostbyname(Address);
    if (Host)
    {
        // Host found, extract its IP address
        myAddress = reinterpret_cast<in_addr*>(Host->h_addr);
    }
    else
    {
        // Host name not found on the network
        myAddress = INADDR_NONE;
    }
}

IPAddress::IPAddress(Uint8 Byte0, Uint8 Byte1, Uint8 Byte2, Uint8 Byte3)
{
    myAddress = htonl((Byte0 << 24) | (Byte1 << 16) | (Byte2 << 8) | Byte3);
}

IPAddress::IPAddress(Uint32 Address)
{
    myAddress = htonl(Address);
}

bool IPAddress::IsValid() const
{
    return myAddress != INADDR_NONE;
}

std::string IPAddress::ToString() const
{
in_addr InAddr;
InAddr.s_addr = myAddress;
return inet_ntoa(InAddr);
}

Uint32 IPAddress::ToInteger() const
{
    return ntohl(myAddress);
}

IPAddress IPAddress::GetLocalAddress()
{
    // The method here is to connect a UDP socket
    // and get the local socket address with the
    // UDP connection will not send anything to
    IPAddress LocalAddress;

    // Create the socket
    SocketHelper::SocketType Socket = socket(PF_INET, SOCK_DGRAM, 0);
    if (Socket == SocketHelper::InvalidSocket())
        return LocalAddress;

    // Build the host address (use a random port)
    sockaddr_in SockAddr;
    memset(SockAddr.sin_zero, 0, sizeof(SockAddr.sin_zero));
    SockAddr.sin_addr.s_addr = INADDR_LOOPBACK;
    SockAddr.sin_family = AF_INET;
    SockAddr.sin_port = htons(4567);

    // Connect the socket
    if (connect(Socket, reinterpret_cast<sockaddr*>(&SockAddr),
    {          
        SocketHelper::Close(Socket);
    return LocalAddress;

// Get the local address of the socket connection
SocketHelper::LengthType Size = sizeof(SockAddr);
if (getsockname(Socket, reinterpret_cast<sockaddr*>(&SockAddr),
               &Size) == -1)
{
    SocketHelper::Close(Socket);
    return LocalAddress;
}

// Close the socket
SocketHelper::Close(Socket);

// Finally build the IP address
LocalAddress.myAddress = SockAddr.sin_addr.s_addr;
return LocalAddress;

IPAddress IPAddress::GetPublicAddress(float Timeout)
{
    // The trick here is more complicated, because
    // to get our public IP address is to get it
    // Here we get the web page from http://www.
    // and parse the result to extract our IP address
    // (not very hard : the web page contains only
    Http Server("www.sfml-dev.org");
    Http::Request Request(Http::Request::Get, "/ip-provider.php");
    Http::Response Page = Server.SendRequest(Request, Timeout);
    if (Page.GetStatus() == Http::Response::Ok)
        return IPAddress(Page.GetBody());
    // Something failed: return an invalid address
    return IPAddress();
bool IPAddress::operator ==(const IPAddress& Other)
{
    return myAddress == Other.myAddress;
}

bool IPAddress::operator !=(const IPAddress& Other)
{
    return myAddress != Other.myAddress;
}

bool IPAddress::operator <(const IPAddress& Other)
{
    return myAddress < Other.myAddress;
}

bool IPAddress::operator >(const IPAddress& Other)
{
    return myAddress > Other.myAddress;
}

bool IPAddress::operator <=(const IPAddress& Other)
{
    return myAddress <= Other.myAddress;
}

bool IPAddress::operator >=(const IPAddress& Other)
{
    return myAddress >= Other.myAddress;
}
std::istream& operator >> (std::istream& Stream,
  
  std::string Str;
  Stream >> Str;
  Address = IPAddress (Str);
  
  return Stream;
  
  std::ostream& operator << (std::ostream& Stream,
  
  return Stream << Address.ToString();
  
  // namespace sf

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
Listener.hpp

```cpp
#ifndef SFML_LISTENER_HPP
#define SFML_LISTENER_HPP

// Headers
#include <SFML/Config.hpp>
#include <SFML/System/Vector3.hpp>

namespace sf {

class SFML_API Listener
```
{ public:
    static void SetGlobalVolume(float Volume);
    static float GetGlobalVolume();
    static void SetPosition(float X, float Y, float Z);
    static void SetPosition(const Vector3f& Position);
    static Vector3f GetPosition();
    static void SetTarget(float X, float Y, float Z);
    static void SetTarget(const Vector3f& Target);
    static Vector3f GetTarget();
};
}
namespace sf
#endif // SFML_LISTENER_HPP

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
sf::Listener Member List

This is the complete list of members for `sf::Listener`, including all inherited members.

<table>
<thead>
<tr>
<th>Method</th>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetGlobalVolume()</td>
<td>sf::Listener</td>
<td>[static]</td>
</tr>
<tr>
<td>GetPosition()</td>
<td>sf::Listener</td>
<td>[static]</td>
</tr>
<tr>
<td>GetTarget()</td>
<td>sf::Listener</td>
<td>[static]</td>
</tr>
<tr>
<td>SetGlobalVolume(float Volume)</td>
<td>sf::Listener</td>
<td>[static]</td>
</tr>
<tr>
<td>SetPosition(float X, float Y, float Z)</td>
<td>sf::Listener</td>
<td>[static]</td>
</tr>
<tr>
<td>SetPosition(const Vector3f &amp;Position)</td>
<td>sf::Listener</td>
<td>[static]</td>
</tr>
<tr>
<td>SetTarget(float X, float Y, float Z)</td>
<td>sf::Listener</td>
<td>[static]</td>
</tr>
<tr>
<td>SetTarget(const Vector3f &amp;Target)</td>
<td>sf::Listener</td>
<td>[static]</td>
</tr>
</tbody>
</table>
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.

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

// Headers
#include <SFML/Audio/Listener.hpp>
#include <SFML/Audio/OpenAL.hpp>
namespace sf
{
    void Listener::SetGlobalVolume(float Volume)
    {
        ALCheck(alListenerf(AL_GAIN, Volume * 0.01f);
    }
}
float Listener::GetGlobalVolume()
{
    float Volume = 0.f;
    ALCheck(alGetListenerf(AL_GAIN, &Volume));
    return Volume;
}

void Listener::SetPosition(float X, float Y, float Z)
{
    ALCheck(alListener3f(AL_POSITION, X, Y, Z));
}

void Listener::SetPosition(const Vector3f& Position)
{
    SetPosition(Position.x, Position.y, Position.z);
}

Vector3f Listener::GetPosition()
{
    Vector3f Position;
    ALCheck(alGetListener3f(AL_POSITION, &Position));
    return Position;
}

void Listener::SetTarget(float X, float Y, float Z)
{
    float Orientation[] = {X, Y, Z, 0.f, 1.f, 0.f};
    ALCheck(alListenerfv(AL_ORIENTATION, Orientation));
}
void Listener::SetTarget(const Vector3f& Target) {
    SetTarget(Target.x, Target.y, Target.z);
}

Vector3f Listener::GetTarget()
{
    float Orientation[6];
    ALCheck(alGetListenerfv(AL_ORIENTATION, Orientation));
    return Vector3f(Orientation[0], Orientation[1], Orientation[2]);
}

// namespace sf

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
// you must not claim that you wrote the original software.
// If you use this software in a product, an acknowledgment
// in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
// and must not be misrepresented as being the original software.

// 3. This notice may not be removed or altered.

#ifndef SFML_LOCK_HPP
#define SFML_LOCK_HPP

// Headers
#include <SFML/System/NonCopyable.hpp>

namespace sf
{
  class Mutex;
}
class SFML_API Lock : NonCopyable
{
    public :
    public:
    Lock(Mutex& Mutex);
    ~Lock();

    private :
    private:
    // Member data
    Mutex& myMutex;
};

} // namespace sf

#endif // SFML_LOCK_HPP
## sf::Lock Member List

This is the complete list of members for `sf::Lock`, including all inherited members.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lock(Mutex &amp;Mutex)</td>
<td>sf::Lock</td>
</tr>
<tr>
<td>NonCopyable()</td>
<td>sf::NonCopyable [inline, private]</td>
</tr>
<tr>
<td>~Lock()</td>
<td>sf::Lock</td>
</tr>
</tbody>
</table>

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)
// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.
// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:
// 1. The origin of this software must not be misrepresented;
// you must not claim that you wrote the original software.
// If you use this software in a product, an acknowledgment
// in the product documentation would be appreciated but is not required.
// 2. Altered source versions must be plainly marked as such,
// and must not be misrepresented as being the original software.
// 3. This notice may not be removed or altered from any source distribution.

// Headers
#include <SFML/System/Lock.hpp>
#include <SFML/System/Mutex.hpp>

namespace sf
{
    Lock::Lock(Mutex& Mutex) :
        myMutex(Mutex)
    {
        myMutex.Lock();
    }
}
Lock::~Lock()
{
  myMutex.Unlock();
}

// namespace sf
Matrix3.hpp

#ifndef SFML_MATRIX3_HPP
#define SFML_MATRIX3_HPP

// Headers
#include <SFML/Config.hpp>
#include <SFML/System/Vector2.hpp>
#include <math.h>

namespace sf {


class SFML_API Matrix3 {
public :
    Matrix3();
    Matrix3(float a00, float a01, float a02,
             float a10, float a11, float a12,
             float a20, float a21, float a22);

    void SetFromTransformations(const Vector2f& Center,
                                 Vector2f Transform(const Vector2f& Point) const;
    Matrix3 GetInverse() const;
    const float* Get4x4Elements() const;

    float operator ()(unsigned int Row, unsigned int Col) const;
    float& operator ()(unsigned int Row, unsigned int Col);
    Matrix3 operator *(const Matrix3& Mat) const;
    Matrix3& operator *=(const Matrix3& Mat);

    // Static member data
    static const Matrix3 Identity;

private :
    // Member data
    float myData[16];
};

#include <SFML/Graphics/Matrix3.inl>
} // namespace sf
#endif // SFML_MATRIX3_HPP
sf::Matrix3 Member List

This is the complete list of members for sf::Matrix3, including all inherited members.

<table>
<thead>
<tr>
<th>Function</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get4x4Elements() const</td>
<td>sf::Matrix3</td>
</tr>
<tr>
<td>GetInverse() const</td>
<td>sf::Matrix3</td>
</tr>
<tr>
<td>Identity</td>
<td>sf::Matrix3  [static]</td>
</tr>
<tr>
<td>Matrix3()</td>
<td>sf::Matrix3</td>
</tr>
<tr>
<td>Matrix3(float a00, float a01, float a02, float a10, float a11, float a12, float a20, float a21, float a22)</td>
<td>sf::Matrix3</td>
</tr>
<tr>
<td>operator()(unsigned int Row, unsigned int Col) const</td>
<td>sf::Matrix3</td>
</tr>
<tr>
<td>operator()(unsigned int Row, unsigned int Col) (defined in sf::Matrix3)</td>
<td>sf::Matrix3</td>
</tr>
<tr>
<td>operator*(const Matrix3 &amp;Mat) const</td>
<td>sf::Matrix3</td>
</tr>
<tr>
<td>operator*(const Matrix3 &amp;Mat)</td>
<td>sf::Matrix3</td>
</tr>
<tr>
<td>SetFromTransformations(const Vector2f &amp;Center, const Vector2f &amp;Translation, float Rotation, const Vector2f &amp;Scale)</td>
<td>sf::Matrix3</td>
</tr>
<tr>
<td>Transform(const Vector2f &amp;Point) const</td>
<td>sf::Matrix3</td>
</tr>
</tbody>
</table>
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gomila@gmail.com)

// This software is provided 'as-is', without any
// express or implied warranty. In no event will the authors be held liable for
// any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented; you must not
   claim that you wrote the original software. If you use this software in a
   product, an acknowledgment in the product documentation would be
   appreciated but is not required.

// 2. Altered source versions must be plainly marked as such, and must not
   be misrepresented as being the original software.

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

// Headers
#include <SFML/Graphics/Matrix3.hpp>

namespace sf
{
  // Static member data
  const Matrix3 Matrix3::Identity(1, 0, 0,
                                0, 1, 0,
                                0, 0, 1);
} // namespace sf

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
SFML - Simple and Fast Multimedia Library
Copyright (C) 2007-2009 Laurent Gomila (laurent.gomila@gmail.com)

This software is provided 'as-is', without any express or implied warranty.
In no event will the authors be held liable for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,
including commercial applications, and to alter it and redistribute it freely,
subject to the following restrictions:

1. The origin of this software must not be misrepresented;
you must not claim that you wrote the original software.
If you use this software in a product, an acknowledgment
in the product documentation would be appreciated but is not required.

2. Altered source versions must be plainly marked as such,
and must not be misrepresented as being the original software.

3. This notice may not be removed or altered.

#ifndef SFML_MUSIC_HPP
#define SFML_MUSIC_HPP

// Headers
#include <SFML/Audio/SoundStream.hpp>
#include <string>
#include <vector>

namespace sf {

namespace priv
{

class SoundFile;
}

class SFML_API Music : public SoundStream
{

public:

explicit Music(std::size_t BufferSize = 44100);
~Music();

bool OpenFromFile(const std::string& Filename);

bool OpenFromMemory(const char* Data, std::size_t SizeInBytes);

float GetDuration() const;

private:

virtual bool OnStart();

virtual bool OnGetData(Chunk& Data);

// Member data
priv::SoundFile* myFile;
float myDuration;
std::vector<Int16> mySamples;

};

} // namespace sf

#ifdef // SFML_MUSIC_HPP

#endif // SFML_MUSIC_HPP
This is the complete list of members for `sf::Music`, including all inherited members.

<table>
<thead>
<tr>
<th>Member Function</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetChannelsCount() const</td>
<td>sf::SoundStream</td>
</tr>
<tr>
<td>GetDuration() const</td>
<td>sf::Music</td>
</tr>
<tr>
<td>GetLoop() const</td>
<td>sf::SoundStream</td>
</tr>
<tr>
<td>GetPlayingOffset() const</td>
<td>sf::SoundStream</td>
</tr>
<tr>
<td>GetSampleRate() const</td>
<td>sf::SoundStream</td>
</tr>
<tr>
<td>GetStatus() const</td>
<td>sf::SoundStream</td>
</tr>
<tr>
<td>Initialize(unsigned int ChannelsCount, unsigned int SampleRate)</td>
<td>sf::SoundStream [protected]</td>
</tr>
<tr>
<td>Music(std::size_t BufferSize=44100)</td>
<td>sf::Music [explicit]</td>
</tr>
<tr>
<td>OpenFromFile(const std::string &amp;Filename)</td>
<td>sf::Music</td>
</tr>
<tr>
<td>OpenFromMemory(const char *Data, std::size_t SizeInBytes)</td>
<td>sf::Music</td>
</tr>
<tr>
<td>Play()</td>
<td>sf::SoundStream</td>
</tr>
<tr>
<td>SetLoop(bool Loop)</td>
<td>sf::SoundStream [protected]</td>
</tr>
<tr>
<td>SoundStream()</td>
<td>sf::SoundStream</td>
</tr>
<tr>
<td>Status enum name</td>
<td>sf::SoundStream</td>
</tr>
<tr>
<td>Stop()</td>
<td>sf::SoundStream</td>
</tr>
<tr>
<td>~Music()</td>
<td>sf::Music</td>
</tr>
<tr>
<td>~SoundStream()</td>
<td>sf::SoundStream [virtual]</td>
</tr>
<tr>
<td>Main Page</td>
<td>Namespaces</td>
</tr>
<tr>
<td>-----------</td>
<td>------------</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Sound.hpp

// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
// you must not claim that you wrote the original software.
// If you use this software in a product, an acknowledgment
// in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
// and must not be misrepresented as being the original software.

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

#ifndef SFML_SOUND_HPP
#define SFML_SOUND_HPP

// Headers
#include <SFML/System/Resource.hpp>
#include <SFML/System/Vector3.hpp>
#include <SFML/Audio/AudioResource.hpp>
#include <cstdlib>

namespace sf

```cpp
class SoundBuffer;

class SFML_API Sound : public AudioResource
{
    public :

    enum Status
    {
        Stopped,
        Paused,
        Playing
    };

    Sound();

    explicit Sound(const SoundBuffer& Buffer, bool);

    Sound(const Sound& Copy);

    ~Sound();

    void Play();

    void Pause();

    void Stop();

    void SetBuffer(const SoundBuffer& Buffer);

    void SetLoop(bool Loop);

    void SetPitch(float Pitch);

    void SetVolume(float Volume);

    void SetPosition(float X, float Y, float Z);
```
void SetPosition(const Vector3f& Position);
void SetRelativeToListener(bool Relative);
void SetMinDistance(float MinDistance);
void SetAttenuation(float Attenuation);
void SetPlayingOffset(float TimeOffset);
const SoundBuffer* GetBuffer() const;
bool GetLoop() const;
float GetPitch() const;
float GetVolume() const;
Vector3f GetPosition() const;
bool IsRelativeToListener() const;
float GetMinDistance() const;
float GetAttenuation() const;
Status GetStatus() const;
float GetPlayingOffset() const;
Sound& operator =(const Sound& Other);
void ResetBuffer();
private:
friend class SoundStream;

// Member data
unsigned int mySource;
ResourcePtr<SoundBuffer> myBuffer;

};

} namespace sf

#endif // SFML_SOUND_HPP
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any
// express or implied warranty. In no event will the authors
// be held liable for any damages arising from the use
// of this software.

// Permission is granted to anyone to use this software
// for any purpose, including commercial applications,
// and to alter it and redistribute it freely, subject
// to the following restrictions:

// 1. The origin of this software must not be
//    misrepresented; you must not claim that you wrote
//    the original software. If you use this software in
//    a product, an acknowledgment in the product
//    documentation would be appreciated but is not
//    required.

// 2. Altered source versions must be plainly marked
//    as such, and must not be misrepresented as being
//    the original software.

// 3. This notice may not be removed or altered.

// Headers
#include <SFML/Audio/Music.hpp>
#include <SFML/Audio/OpenAL.hpp>
#include <SFML/Audio/SoundFile.hpp>
#include <fstream>
#include <iostream>

namespace sf
{

Music::Music(std::size_t BufferSize) :

myFile   (NULL),
myDuration(0.f),
mySamples (BufferSize)
{
}

Music::~Music()
{
    // We must stop before destroying the file :)
    Stop();
    delete myFile;
}

bool Music::OpenFromFile(const std::string& Filename)
{
    // First stop the music if it was already running
    Stop();
    // Create the sound file implementation, and open it
    delete myFile;
    myFile = priv::SoundFile::CreateRead(Filename);
    if (!myFile)
    {
        std::cerr << "Failed to open " << Filename << " file.
        return false;
    }
    // Compute the duration
    myDuration = static_cast<float>(myFile->GetSamplesCount()) / myFile->GetSampleRate() / myFile->GetChannelsCount();
    // Initialize the stream
    Initialize(myFile->GetChannelsCount(), myFile->GetSampleRate());
return true;
}

bool Music::OpenFromMemory(const char* Data, std::size_t SizeInBytes)
{
    // First stop the music if it was already running
    Stop();
    // Create the sound file implementation, and delete our old one
    delete myFile;
    myFile = priv::SoundFile::CreateRead(Data, SizeInBytes);
    if (!myFile)
    {
        std::cerr << "Failed to open music from memory for reading"
        return false;
    }
    // Compute the duration
    myDuration = static_cast<float>(myFile->GetSamplesCount()) / myFile->GetSampleRate();
    // Initialize the stream
    Initialize(myFile->GetChannelsCount(), myFile->GetSampleRate());
    return true;
}

bool Music::OnStart()
{
    return myFile && myFile->Restart();
}

bool Music::OnGetData(SoundStream::Chunk& Data)
{
    if (myFile)
{ // Fill the chunk parameters
    Data.Samples = &mySamples[0];
    Data.NbSamples = myFile->Read(&mySamples[0], mySamples.size());

    // Check if we have reached the end of the file
    return Data.NbSamples == mySamples.size();
}
else
{
    return false;
}

float Music::GetDuration() const
{
    return myDuration;
}

} // namespace sf

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)
// This software is provided 'as-is', without any
// warranty of any kind. In no event will the authors be held
// liable for any damages arising from the use of this software.
// Permission is granted to anyone to use this software for any
// purpose, including commercial applications, and to alter it
// and redistribute it freely, subject to the following restrictions:
// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an
//    acknowledgment in the product documentation would be appreciated
//    but is not required.
// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the
//    original software.
// 3. This notice may not be removed or altered.

// Headers
#include <SFML/Audio/SoundStream.hpp>
#include <SFML/Audio/AudioDevice.hpp>
#include <SFML/Audio/OpenAL.hpp>
#include <SFML/System/Sleep.hpp>

namespace sf {
    SoundStream::SoundStream() :
        myIsStreaming (false),
SoundStream::~SoundStream() {
    // Stop the sound if it was playing
    Stop();
}

void SoundStream::Initialize(unsigned int ChannelsCount) {
    myChannelsCount = ChannelsCount;
    mySampleRate = SampleRate;
    // Deduce the format from the number of channels
    myFormat = priv::AudioDevice::GetInstance().GetFormatFromChannelsCount(ChannelsCount);
    // Check if the format is valid
    if (myFormat == 0) {
        myChannelsCount = 0;
        mySampleRate = 0;
        std::cerr << "Unsupported number of channels";
    }
}

void SoundStream::Play() {

// Check if the sound parameters have been set
if (myFormat == 0)
{
    std::cerr << "Failed to play audio stream: sound parameters have not been initialized (call Initialize first)"
    return;
}

// If the sound is already playing (probably paused), just resume it
if (myIsStreaming)
{
    Sound::Play();
    return;
}

// Notify the derived class
if (OnStart())
{
    // Start updating the stream in a separate thread to avoid blocking the application
    mySamplesProcessed = 0;
    myIsStreaming = true;
    Launch();
}

void SoundStream::Stop()
{
    // Wait for the thread to terminate
    myIsStreaming = false;
    Wait();
}

unsigned int SoundStream::GetChannelsCount() const
{
    return myChannelsCount;
}
unsigned int SoundStream::GetSampleRate() const
{
    return mySampleRate;
}

Sound::Status SoundStream::GetStatus() const
{
    Status Status = Sound::GetStatus();
    // To compensate for the lag between Play() and PlayStream()
    if ((Status == Stopped) && myIsStreaming)
        Status = Playing;
    return Status;
}

float SoundStream::GetPlayingOffset() const
{
    return Sound::GetPlayingOffset() + static_cast<float>(myOffset);
}

void SoundStream::SetLoop(bool Loop)
{
    myLoop = Loop;
}

bool SoundStream::GetLoop() const
{
    return myLoop;
}
void SoundStream::Run()
{
    // Create the buffers
    ALCheck(alGenBuffers(BuffersCount, myBuffers);
    for (int i = 0; i < BuffersCount; ++i)
        myEndBuffers[i] = false;

    // Fill the queue
    bool RequestStop = FillQueue();

    // Play the sound
    Sound::Play();

    while (myIsStreaming)
    {
        // The stream has been interrupted !
        if (Sound::GetStatus() == Stopped)
            {
                if (!RequestStop)
                    {
                        // Just continue
                        Sound::Play();
                    }
                else
                    {
                        // End streaming
                        myIsStreaming = false;
                    }
            }

        // Get the number of buffers that have been processed
        ALInt NbProcessed;
        ALCheck(alGetSourcei(Sound::mySource, AL_BUFFERS_PROCESSED, &NbProcessed));

        while (NbProcessed--)
        {
            
        }
    }
}
// Pop the first unused buffer from the queue
ALuint Buffer;
ALCheck(alSourceUnqueueBuffers(Sound::mySource, 1, &Buffer));

// Find its number
unsigned int BufferNum = 0;
for (int i = 0; i < BuffersCount; ++i)
    if (myBuffers[i] == Buffer)
        { BufferNum = i; break; }

// Retrieve its size and add it to the samples count
if (myEndBuffers[BufferNum])
    { // This was the last buffer: reset the sample count
        mySamplesProcessed = 0;
        myEndBuffers[BufferNum] = false;
    }
else
    {
        ALint Size;
        ALCheck(alGetBufferi(Buffer, AL_SIZE, &Size));
        mySamplesProcessed += Size / sizeof;
    }

// Fill it and push it back into the playing queue
if (!RequestStop)
    { if (FillAndPushBuffer(BufferNum))
        RequestStop = true;
    }

// Leave some time for the other threads
if (Sound::GetStatus() != Stopped)
Sleep(0.1f);

// Stop the playback
Sound::Stop();

// Unblock any buffer left in the queue
ClearQueue();

// Delete the buffers
ALCheck(alSourcei(Sound::mySource, AL_BUFFER, 0));
ALCheck(alDeleteBuffers(BuffersCount, myBuffers));

bool SoundStream::FillAndPushBuffer(unsigned int BufferNum)
{
    bool RequestStop = false;

    // Acquire audio data
    Chunk Data = {NULL, 0};

    if (!OnGetData(Data))
    {
        // Mark the buffer as the last one (so that we know when to reset the playing position)
        myEndBuffers[BufferNum] = true;

        // Check if the stream must loop or stop
        if (myLoop && OnStart())
        {
            // If we succeeded to restart and we previously had no data, try to fill the buffer once again
            if (!Data.Samples || (Data.NbSamples == 0))
            {
                return FillAndPushBuffer(BufferNum);
            }
        }
        else
        {
            // If we failed to restart, continue
            return false;
        }
    }
    else
    {
        // Check if we have enough data
        if (Data.NbSamples == 0)
        {
            return false;
        }
    }

    return true;
}
// Not looping or restart failed: request stop
RequestStop = true;
}
while
}

// Fill the buffer if some data was returned
if (Data.Samples && Data.NbSamples)
{
    unsigned int Buffer = myBuffers[BufferNum];
    // Fill the buffer
    ALsizei Size = static_cast<ALsizei>(Data.NbSamples) *
    ALCheck(alBufferData(Buffer, myFormat, Data.Samples, Size, mySampleRate));
    // Push it into the sound queue
    ALCheck(alSourceQueueBuffers(Sound::mySource, 1, &Buffer));
}

return RequestStop;

bool SoundStream::FillQueue()
{
    // Fill and enqueue all the available buffers
    bool RequestStop = false;
    for (int i = 0; (i < BuffersCount) && !RequestStop;
    {
        if (FillAndPushBuffer(i))
            RequestStop = true;
    }

    return RequestStop;
}

void SoundStream::ClearQueue()
ALCheck(alGetSourcei(Sound::mySource, AL_BUFFERS_QUEUED, &NbQueued));

// Unqueue them all
ALuint Buffer;
for (ALint i = 0; i < NbQueued; ++i)
    ALCheck(alSourceUnqueueBuffers(Sound::mySource, 1, &Buffer));
}

bool SoundStream::OnStart()
{
    // Does nothing by default
    return true;
}

} // namespace sf

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
MethodInfo.hpp

/* SFML - Simple and Fast Multimedia Library */
/* Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com) */
/* This software is provided 'as-is', without any express or implied warranty. */
/* In no event will the authors be held liable for any damages arising from the use of this software. */
/* Permission is granted to anyone to use this software for any purpose, 
   including commercial applications, and to alter it and redistribute it freely, 
   subject to the following restrictions: */
/* 1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. */
/* If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required. */
/* 2. Altered source versions must be plainly marked as such and must not be misrepresented as being the original software. */
/* 3. This notice may not be removed or altered from any source distribution. */

#ifndef SFML_MUTEXWIN32_HPP
#define SFML_MUTEXWIN32_HPP

// Headers
#include <SFML/System/NonCopyable.hpp>
#include <windows.h>

namespace sf
{
  class SFML_API Mutex : NonCopyable
  {
  #endif
```cpp
public:
Mutex();
~Mutex();
void Lock();
void Unlock();

private:
// Member data
CRITICAL_SECTION myHandle;

};
#endif // SFML_MUTEXWIN32_HPP
```
sf::Mutex Member List

This is the complete list of members for sf::Mutex, including all inherited members.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lock()</td>
<td>sf::Mutex</td>
</tr>
<tr>
<td>Mutex()</td>
<td>sf::Mutex</td>
</tr>
<tr>
<td>NonCopyable()</td>
<td>sf::NonCopyable [inline, private]</td>
</tr>
<tr>
<td>Unlock()</td>
<td>sf::Mutex</td>
</tr>
<tr>
<td>~Mutex()</td>
<td>sf::Mutex</td>
</tr>
</tbody>
</table>

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any
// express or implied warranty. In no event will the authors
// be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any
// purpose, including commercial applications, and to alter it and
// redistribute it freely, subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an
//    acknowledgment in the product documentation would be
//    appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original
//    software.

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

// Headers
#include <SFML/System/Win32/Mutex.hpp>

namespace sf
{
    Mutex::Mutex()
    {
        InitializeCriticalSection(&myHandle);
    }
}
Mutex::~Mutex()
{
    DeleteCriticalSection(&myHandle);
}

void Mutex::Lock()
{
    EnterCriticalSection(&myHandle);
}

void Mutex::Unlock()
{
    LeaveCriticalSection(&myHandle);
}

// namespace sf

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.

// 3. This notice may not be removed or altered.

#ifndef SFML_NONCOPYABLE_HPP
#define SFML_NONCOPYABLE_HPP

namespace sf
{

struct SFML_API NonCopyable
{

#endif
protected:
NonCopyable() {}

private:
NonCopyable(const NonCopyable&);
NonCopyable& operator=(const NonCopyable&);
sf::NonCopyable Member List

This is the complete list of members for sf::NonCopyable, including all inherited members.

NonCopyable() sf::NonCopyable [inline, protected]
Main Page  Namespaces  Classes  Files

File List
Packet.hpp

#ifndef SFML_PACKET_HPP
#define SFML_PACKET_HPP

// Headers
#include <SFML/Config.hpp>
#include <string>
#include <vector>

namespace sf {


class SFML_API Packet {
  public :
  Packet();
  virtual ~Packet();
  void Append(const void* Data, std::size_t SizeInBytes);
  void Clear();
  const char* GetData() const;
  std::size_t GetDataSize() const;
  bool EndOfPacket() const;
  operator bool() const;

  Packet& operator >> (bool& Data);
  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 >> (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 << (bool Data);
  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<<(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);

private:

friend class SocketTCP;
friend class SocketUDP;

bool CheckSize(std::size_t Size);

virtual const char* OnSend(std::size_t& DataSize);

virtual void OnReceive(const char* Data, std::size_t DataSize);

// Member data
std::vector<char> myData;
std::size_t myReadPos;
bool myIsValid;

};

} // namespace sf

#endif // SFML_PACKET_HPP
**sf::Packet Member List**

This is the complete list of members for sf::Packet, including all inherited members.

<table>
<thead>
<tr>
<th>Member</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Append</td>
<td>(const void *Data, std::size_t SizeInBytes)</td>
</tr>
<tr>
<td>Clear</td>
<td></td>
</tr>
<tr>
<td>EndOfPacket() const</td>
<td></td>
</tr>
<tr>
<td>GetData() const</td>
<td></td>
</tr>
<tr>
<td>GetDataSize() const</td>
<td></td>
</tr>
<tr>
<td>operator bool() const</td>
<td></td>
</tr>
<tr>
<td>operator&lt;&lt;(bool Data)</td>
<td></td>
</tr>
<tr>
<td>operator&lt;&lt;(Int8 Data)</td>
<td>(defined in sf::Packet)</td>
</tr>
<tr>
<td>operator&lt;&lt;(Uint8 Data)</td>
<td>(defined in sf::Packet)</td>
</tr>
<tr>
<td>operator&lt;&lt;(Int16 Data)</td>
<td>(defined in sf::Packet)</td>
</tr>
<tr>
<td>operator&lt;&lt;(Uint16 Data)</td>
<td>(defined in sf::Packet)</td>
</tr>
<tr>
<td>operator&lt;&lt;(Int32 Data)</td>
<td>(defined in sf::Packet)</td>
</tr>
<tr>
<td>operator&lt;&lt;(Uint32 Data)</td>
<td>(defined in sf::Packet)</td>
</tr>
<tr>
<td>operator&lt;&lt;(float Data)</td>
<td>(defined in sf::Packet)</td>
</tr>
<tr>
<td>operator&lt;&lt;(double Data)</td>
<td>(defined in sf::Packet)</td>
</tr>
<tr>
<td>operator&gt;&gt;(const char *Data)</td>
<td>(defined in sf::Packet)</td>
</tr>
<tr>
<td>operator&gt;&gt;(const std::string &amp;Data)</td>
<td>(defined in sf::Packet)</td>
</tr>
<tr>
<td>operator&gt;&gt;(const wchar_t *Data)</td>
<td>(defined in sf::Packet)</td>
</tr>
<tr>
<td>operator&gt;&gt;(const std::wstring &amp;Data)</td>
<td>(defined in sf::Packet)</td>
</tr>
<tr>
<td>operator&gt;&gt;(bool &amp;Data)</td>
<td></td>
</tr>
<tr>
<td>operator&gt;&gt;(Int8 &amp;Data)</td>
<td>(defined in sf::Packet)</td>
</tr>
<tr>
<td>operator&gt;&gt;(Uint8 &amp;Data)</td>
<td>(defined in sf::Packet)</td>
</tr>
<tr>
<td>operator&gt;&gt;(Int16 &amp;Data)</td>
<td>(defined in sf::Packet)</td>
</tr>
<tr>
<td>operator&gt;&gt;(Uint16 &amp;Data)</td>
<td>(defined in sf::Packet)</td>
</tr>
<tr>
<td>operator&gt;&gt;(Int32 &amp;Data)</td>
<td>(defined in sf::Packet)</td>
</tr>
<tr>
<td>operator&gt;&gt;(Uint32 &amp;Data)</td>
<td>(defined in sf::Packet)</td>
</tr>
<tr>
<td>operator&gt;&gt;(float &amp;Data)</td>
<td>(defined in sf::Packet)</td>
</tr>
<tr>
<td>operator&gt;&gt;(double &amp;Data)</td>
<td>(defined in sf::Packet)</td>
</tr>
<tr>
<td>operator&gt;&gt;(char *Data)</td>
<td>(defined in sf::Packet)</td>
</tr>
<tr>
<td>Function</td>
<td>Definition</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td><code>operator&gt;&gt;(std::string &amp;Data)</code></td>
<td>(defined in <code>sf::Packet</code>)</td>
</tr>
<tr>
<td><code>operator&gt;&gt;(wchar_t *Data)</code></td>
<td>(defined in <code>sf::Packet</code>)</td>
</tr>
<tr>
<td><code>operator&gt;&gt;(std::wstring &amp;Data)</code></td>
<td>(defined in <code>sf::Packet</code>)</td>
</tr>
<tr>
<td><code>Packet()</code></td>
<td></td>
</tr>
<tr>
<td><code>SocketTCP</code></td>
<td>(defined in <code>sf::Packet</code>)</td>
</tr>
<tr>
<td><code>SocketUDP</code></td>
<td>(defined in <code>sf::Packet</code>)</td>
</tr>
<tr>
<td><code>~Packet()</code></td>
<td></td>
</tr>
</tbody>
</table>
Packet.cpp

// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.

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

// Headers
#include <SFML/Network/Packet.hpp>
#include <SFML/Network/SocketHelper.hpp>
#include <string.h>

namespace sf
{
    Packet::Packet()
    {
        myReadPos(0),
        myIsValid(true)
    }
}
Packet::~Packet()
{
}

void Packet::Append(const void* Data, std::size_t SizeInBytes)
{
    if (Data && (SizeInBytes > 0))
    {
        std::size_t Start = mData.size();
        mData.resize(Start + SizeInBytes);
        memcpy(&mData[Start], Data, SizeInBytes);
    }
}

void Packet::Clear()
{
    mData.clear();
    mReadPos = 0;
    mIsValid = true;
}

const char* Packet::GetData() const
{
    return !mData.empty() ? &mData[0] : NULL;
}

std::size_t Packet::GetDataSize() const
{
```cpp
{ return myData.size(); }

bool Packet::EndOfPacket() const
{
    return myReadPos >= myData.size();
}

Packet::operator bool() const
{
    return myIsValid;
}

Packet& Packet::operator >> (bool& Data)
{
    Uint8 Value;
    if (*this >> Value)
        Data = (Value != 0);
    return *this;
}

Packet& Packet::operator >> (Int8& Data)
{
    if (CheckSize(sizeof(Data)))
    {
        Data = *reinterpret_cast<const Int8*>(GetData);
        myReadPos += sizeof(Data);
    }
    return *this;
}

Packet& Packet::operator >> (Uint8& Data)
{
```
if (CheckSize(sizeof(Data)))
{
    Data = reinterpret_cast<const Uint8*>(myReadPos);
    myReadPos += sizeof(Data);
}

return *this;

Packet& Packet::operator>>(Int16& Data)
{
    if (CheckSize(sizeof(Data)))
    {
        Data = ntohs(*reinterpret_cast<const Int16*>(myReadPos));
        myReadPos += sizeof(Data);
    }
    return *this;
}

Packet& Packet::operator>>(Uint16& Data)
{
    if (CheckSize(sizeof(Data)))
    {
        Data = ntohs(*reinterpret_cast<const Uint16*>(myReadPos));
        myReadPos += sizeof(Data);
    }
    return *this;
}

Packet& Packet::operator>>(Int32& Data)
{
    if (CheckSize(sizeof(Data)))
    {
        Data = ntohl(*reinterpret_cast<const Int32*>(myReadPos));
        myReadPos += sizeof(Data);
    }
    return *this;
Packet& Packet::operator>>(Uint32& Data)
{
    if (CheckSize(sizeof(Data)))
    {
        Data = ntohl(*reinterpret_cast<const Uint32*>(myReadPos += sizeof(Data));
    }

    return *this;
}

Packet& Packet::operator>>(float& Data)
{
    if (CheckSize(sizeof(Data)))
    {
        Data = *reinterpret_cast<const float*>(myReadPos += sizeof(Data));
    }

    return *this;
}

Packet& Packet::operator>>(double& Data)
{
    if (CheckSize(sizeof(Data)))
    {
        Data = *reinterpret_cast<const double*>(myReadPos += sizeof(Data));
    }

    return *this;
}

Packet& Packet::operator>>(char* Data)
{
    // First extract string length
    Uint32 Length;
    *this >> Length;
    
    return *this;
}
if ((Length > 0) && CheckSize(Length)) {
    // Then extract characters
    memcpy(Data, GetData() + myReadPos, Length);
    Data[Length] = '\'0';
    // Update reading position
    myReadPos += Length;
}
return *this;
}

Packet& Packet::operator>>(std::string& Data) {
    // First extract string length
    Uint32 Length;
    *this >> Length;
    Data.clear();
    if ((Length > 0) && CheckSize(Length)) {
        // Then extract characters
        Data.assign(GetData() + myReadPos, Length);
        // Update reading position
        myReadPos += Length;
    }
    return *this;
}

Packet& Packet::operator>>(wchar_t* Data) {
    // First extract string length
    Uint32 Length;
    *this >> Length;
    if ((Length > 0) && CheckSize(Length * sizeof wchar_t)) {
        // Then extract characters
        Data->assign(GetData() + myReadPos, Length);
        // Update reading position
        myReadPos += Length;
    }
    return *this;
}
{ // Then extract characters
  for (Uint32 i = 0; i < Length; ++i)
  {
    Uint32 c;
    *this >> c;
    Data[i] = static_cast<wchar_t>(c);
  }
  Data[Length] = L'\0';
}

return *this;

Packet& Packet::operator>>(std::wstring& Data)
{
  // First extract string length
  Uint32 Length;
  *this >> Length;
  Data.clear();
  if ((Length > 0) && CheckSize(Length * sizeof(wchar_t)))
  {
    // Then extract characters
    for (Uint32 i = 0; i < Length; ++i)
    {
      Uint32 c;
      *this >> c;
      Data += static_cast<wchar_t>(c);
    }
  }
  return *this;
}

Packet& Packet::operator<<(bool Data)
{

*this << static_cast<Uint8>(Data);
return *this;

Packet& Packet::operator <<(Int8 Data)
{
    Append(&Data, sizeof(Data));
    return *this;
}

Packet& Packet::operator <<(Uint8 Data)
{
    Append(&Data, sizeof(Data));
    return *this;
}

Packet& Packet::operator <<(Int16 Data)
{
    Int16 ToWrite = htons(Data);
    Append(&ToWrite, sizeof(ToWrite));
    return *this;
}

Packet& Packet::operator <<(Uint16 Data)
{
    Uint16 ToWrite = htons(Data);
    Append(&ToWrite, sizeof(ToWrite));
    return *this;
}

Packet& Packet::operator <<(Int32 Data)
{
    Int32 ToWrite = htonl(Data);
    Append(&ToWrite, sizeof(ToWrite));
    return *this;
}

Packet& Packet::operator <<(Uint32 Data)
{
    Uint32 ToWrite = htonl(Data);
    Append(&ToWrite, sizeof(ToWrite));
    return *this;
}
Packet& Packet::operator << (float Data)
{
    Append(&Data, sizeof(Data));
    return *this;
}

Packet& Packet::operator << (double Data)
{
    Append(&Data, sizeof(Data));
    return *this;
}

Packet& Packet::operator << (const char* Data)
{
    // First insert string length
    Uint32 Length = 0;
    for (const char* c = Data; *c != '\0'; ++c)
        ++Length;
    *this << Length;

    // Then insert characters
    Append(Data, Length * sizeof(char));
    return *this;
}

Packet& Packet::operator << (const std::string& Data)
{
    // First insert string length
    Uint32 Length = static_cast<Uint32>(Data.size());
    *this << Length;

    // Then insert characters
    if (Length > 0)
    {
        Append(Data.c_str(), Length * sizeof(std::string::value_type));
    }
    return *this;
}
Packet& Packet::operator <<(const wchar_t* Data) {
    // First insert string length
    Uint32 Length = 0;
    for (const wchar_t* c = Data; *c != L'\0'; ++Length)
        *this << Length;

    // Then insert characters
    for (const wchar_t* c = Data; *c != L'\0'; ++c)
        *this << static_cast<Int32>(*c);

    return *this;
}

Packet& Packet::operator <<(const std::wstring& Data) {
    // First insert string length
    Uint32 Length = static_cast<Uint32>(Data.size());
    *this << Length;

    // Then insert characters
    if (Length > 0)
    {
        for (std::wstring::const_iterator c = Data.begin(); c != Data.end(); ++c)
            *this << static_cast<Int32>(*c);
    }

    return *this;
}

bool Packet::CheckSize(std::size_t Size) {
    myIsValid = myIsValid && (myReadPos + Size < myData.size());
    return myIsValid;
}
const char* Packet::OnSend(std::size_t& DataSize) {
    DataSize = GetDataSize();
    return GetData();
}

void Packet::OnReceive(const char* Data, std::size_t DataSize) {
    Append(Data, DataSize);
}

} // namespace sf

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any
// In no event will the authors be held liable for

// Permission is granted to anyone to use this software
// including commercial applications, and to alter it and
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
// you must not claim that you wrote the original software.
// If you use this software in a product, an
// in the product documentation would be appreciated.

// 2. Altered source versions must be plainly marked as such,
// and must not be misrepresented as being the

// 3. This notice may not be removed or altered.

#ifndef SFML_POSTFX_HPP
#define SFML_POSTFX_HPP

// Headers
#include <SFML/Graphics/Drawable.hpp>
#include <SFML/Graphics/Image.hpp>
#include <istream>
#include <map>
#include <string>

#endif // SFML_POSTFX_HPP
namespace sf {

class SFML_API PostFX : public Drawable {

public:

PostFX();
PostFX(const PostFX& Copy);
~PostFX();

bool LoadFromFile(const std::string& Filename);
bool LoadFromMemory(const std::string& Effect);

void SetParameter(const std::string& Name, float);
void SetParameter(const std::string& Name, int);
void SetParameter(const std::string& Name, bool);
void SetTexture(const std::string& Name, Image);

PostFX& operator =(const PostFX& Other);

static bool CanUsePostFX();

protected:

virtual void Render(RenderTarget& Target) const;

private:

static std::string PreprocessEffect(std::istream& File);
void CreateProgram();

// Types
typedef std::map<std::string, const Image*> TextureTable;

// Member data
unsigned int myShaderProgram;
TextureTable myTextures;
std::string myFragmentShader;
mutable Image myFrameBuffer;

} // namespace sf

#endif // SFML_POSTFX_HPP
This is the complete list of members for \texttt{sf::PostFX}, including all inherited members.

<table>
<thead>
<tr>
<th>Member function</th>
<th>Return Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{CanUsePostFX()}</td>
<td>\texttt{sf::PostFX} [static]</td>
<td></td>
</tr>
<tr>
<td>\texttt{Drawable(const Vector2f &amp;Position=Vector2f(0, 0), const Vector2f &amp;Scale=Vector2f(1, 1), float Rotation=0.f, const Color &amp;Col=Color(255, 255, 255, 255))}</td>
<td>\texttt{sf::Drawable}</td>
<td></td>
</tr>
<tr>
<td>\texttt{GetBlendMode() const}</td>
<td>\texttt{sf::Drawable}</td>
<td></td>
</tr>
<tr>
<td>\texttt{GetCenter() const}</td>
<td>\texttt{sf::Drawable}</td>
<td></td>
</tr>
<tr>
<td>\texttt{GetColor() const}</td>
<td>\texttt{sf::Drawable}</td>
<td></td>
</tr>
<tr>
<td>\texttt{GetInverseMatrix() const}</td>
<td>\texttt{sf::Drawable} [protected]</td>
<td></td>
</tr>
<tr>
<td>\texttt{GetMatrix() const}</td>
<td>\texttt{sf::Drawable} [protected]</td>
<td></td>
</tr>
<tr>
<td>\texttt{GetPosition() const}</td>
<td>\texttt{sf::Drawable}</td>
<td></td>
</tr>
<tr>
<td>\texttt{GetRotation() const}</td>
<td>\texttt{sf::Drawable}</td>
<td></td>
</tr>
<tr>
<td>\texttt{GetScale() const}</td>
<td>\texttt{sf::Drawable}</td>
<td></td>
</tr>
<tr>
<td>\texttt{LoadFromFile(const std::string &amp;Filename)}</td>
<td>\texttt{sf::PostFX}</td>
<td></td>
</tr>
<tr>
<td>\texttt{LoadFromMemory(const std::string &amp;Effect)}</td>
<td>\texttt{sf::PostFX}</td>
<td></td>
</tr>
<tr>
<td>\texttt{Move(float OffsetX, float OffsetY)}</td>
<td>\texttt{sf::Drawable}</td>
<td></td>
</tr>
<tr>
<td>\texttt{Move(const Vector2f &amp;Offset)}</td>
<td>\texttt{sf::Drawable}</td>
<td></td>
</tr>
<tr>
<td>\texttt{operator=(const PostFX &amp;Other)}</td>
<td>\texttt{sf::PostFX}</td>
<td></td>
</tr>
<tr>
<td>\texttt{PostFX()}</td>
<td>\texttt{sf::PostFX}</td>
<td></td>
</tr>
<tr>
<td>\texttt{PostFX(const PostFX &amp;Copy)}</td>
<td>\texttt{sf::PostFX}</td>
<td></td>
</tr>
<tr>
<td>\texttt{Render(RenderTarget &amp;Target) const}</td>
<td>\texttt{sf::PostFX} [protected, virtual]</td>
<td></td>
</tr>
<tr>
<td>\texttt{Rotate(float Angle)}</td>
<td>\texttt{sf::Drawable}</td>
<td></td>
</tr>
<tr>
<td>\texttt{Scale(float FactorX, float FactorY)}</td>
<td>\texttt{sf::Drawable}</td>
<td></td>
</tr>
<tr>
<td>\texttt{Scale(const Vector2f &amp;Factor)}</td>
<td>\texttt{sf::Drawable}</td>
<td></td>
</tr>
<tr>
<td>\texttt{SetBlendMode(Blend::Mode Mode)}</td>
<td>\texttt{sf::Drawable}</td>
<td></td>
</tr>
<tr>
<td>\texttt{SetCenter(float CenterX, float CenterY)}</td>
<td>\texttt{sf::Drawable}</td>
<td></td>
</tr>
<tr>
<td>\texttt{SetCenter(const Vector2f &amp;Center)}</td>
<td>\texttt{sf::Drawable}</td>
<td></td>
</tr>
<tr>
<td>\texttt{SetColor(const Color &amp;Col)}</td>
<td>\texttt{sf::Drawable}</td>
<td></td>
</tr>
<tr>
<td>\texttt{SetParameter(const std::string &amp;Name, float X)}</td>
<td>\texttt{sf::PostFX}</td>
<td></td>
</tr>
<tr>
<td>\texttt{SetParameter(const std::string &amp;Name, float X, float Y)}</td>
<td>\texttt{sf::PostFX}</td>
<td></td>
</tr>
</tbody>
</table>

\texttt{sf::PostFX}
<table>
<thead>
<tr>
<th>Member Function</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sf::PostFX::SetParameter(const std::string &amp;Name, float X, float Y, float Z)</code></td>
<td>sf::PostFX</td>
</tr>
<tr>
<td><code>sf::PostFX::SetParameter(const std::string &amp;Name, float X, float Y, float Z, float W)</code></td>
<td>sf::PostFX</td>
</tr>
<tr>
<td><code>sf::Drawable::setPosition(float X, float Y)</code></td>
<td>sf::Drawable</td>
</tr>
<tr>
<td><code>sf::Drawable::setPosition(const Vector2f &amp;Position)</code></td>
<td>sf::Drawable</td>
</tr>
<tr>
<td><code>sf::Drawable::setRotation(float Rotation)</code></td>
<td>sf::Drawable</td>
</tr>
<tr>
<td><code>sf::Drawable::setScale(float ScaleX, float ScaleY)</code></td>
<td>sf::Drawable</td>
</tr>
<tr>
<td><code>sf::Drawable::setScale(const Vector2f &amp;Scale)</code></td>
<td>sf::Drawable</td>
</tr>
<tr>
<td><code>sf::Drawable::setScaleX(float FactorX)</code></td>
<td>sf::Drawable</td>
</tr>
<tr>
<td><code>sf::Drawable::setScaleY(float FactorY)</code></td>
<td>sf::Drawable</td>
</tr>
<tr>
<td><code>sf::PostFX::SetX(float X)</code></td>
<td>sf::PostFX</td>
</tr>
<tr>
<td><code>sf::PostFX::SetY(float Y)</code></td>
<td>sf::PostFX</td>
</tr>
<tr>
<td><code>sf::Drawable::TransformToGlobal(const sf::Vector2f &amp;Point) const</code></td>
<td>sf::Drawable</td>
</tr>
<tr>
<td><code>sf::Drawable::TransformToLocal(const sf::Vector2f &amp;Point) const</code></td>
<td>sf::Drawable</td>
</tr>
<tr>
<td><code>~sf::Drawable()</code></td>
<td>sf::Drawable [virtual]</td>
</tr>
<tr>
<td><code>~sf::PostFX()</code></td>
<td>sf::PostFX</td>
</tr>
</tbody>
</table>
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any
// express or implied warranty. In no event will the authors
// be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any
// purpose, including commercial applications, and to alter it and
// redistribute it freely, subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an
//    acknowledgment in the product documentation would be appreciated
//    but is not required.

// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.

// 3. This notice may not be removed or altered.

// Headers
#include <SFML/Graphics/PostFX.hpp>
#include <SFML/Graphics/RenderWindow.hpp>
#include <SFML/Graphics/GraphicsContext.hpp>
#include <fstream>
#include <iostream>
#include <set>
#include <sstream>
namespace sf
{
PostFX::PostFX() :
myShaderProgram(0)
{
    // No filtering on frame buffer
    myFrameBuffer.SetSmooth(false);
}

PostFX::PostFX(const PostFX& Copy) :
Drawable (Copy),
myShaderProgram (0),
myTextures (Copy.myTextures),
myFragmentShader(Copy.myFragmentShader),
myFrameBuffer  (Copy.myFrameBuffer)
{
    // No filtering on frame buffer
    myFrameBuffer.SetSmooth(false);
    // Create the shaders and the program
    if (Copy.myShaderProgram)
        CreateProgram();
}

PostFX::~PostFX()
{
    // Destroy effect program
    if (myShaderProgram)
    {
        // Make sure we have a valid context
        priv::GraphicsContext Ctx;
        GLCheck(glDeleteObjectARB(myShaderProgram));
    }
}
bool PostFX::LoadFromFile(const std::string& Filename) {
    // Open the file
    std::ifstream File(Filename.c_str());
    if (!File) {
        std::cerr << "Failed to open effect file" << std::endl;
        return false;
    }
    // Apply the preprocessing pass to the fragment shader code
    myFragmentShader = PreprocessEffect(File);
    // Create the shaders and the program
    CreateProgram();
    return myShaderProgram != 0;
}

bool PostFX::LoadFromMemory(const std::string& Effect) {
    // Open a stream and copy the effect code
    std::istringstream Stream(Effect.c_str());
    // Apply the preprocessing pass to the fragment shader code
    myFragmentShader = PreprocessEffect(Stream);
    // Create the shaders and the program
    CreateProgram();
    return myShaderProgram != 0;
}
void PostFX::SetParameter(const std::string& Name)
{
    if (myShaderProgram)
    {
        // Enable program
        GLCheck(glUseProgramObjectARB(myShaderProgram));

        // Get parameter location and assign it
        GLint Location = glGetUniformLocationARB(myShaderProgram, Name.c_str());
        if (Location != -1)
            GLCheck(glUniform1fARB(Location, X));
        else
            std::cerr << "Parameter " << Name;

        // Disable program
        GLCheck(glUseProgramObjectARB(0));
    }
}

void PostFX::SetParameter(const std::string& Name)
{
    if (myShaderProgram)
    {
        // Enable program
        GLCheck(glUseProgramObjectARB(myShaderProgram));

        // Get parameter location and assign it
        GLint Location = glGetUniformLocationARB(myShaderProgram, Name.c_str());
        if (Location != -1)
            GLCheck(glUniform2fARB(Location, X, Y));
        else
            std::cerr << "Parameter " << Name;

        // Disable program
        GLCheck(glUseProgramObjectARB(0));
    }
}
void PostFX::SetParameter(const std::string& Name) {
    if (myShaderProgram) {
        // Enable program
        GLCheck(glUseProgramObjectARB(myShaderProgram));

        // Get parameter location and assign it
        GLint Location = glGetUniformLocationARB(myShaderProgram, Name.c_str());
        if (Location != -1) {
            GLCheck(glUniform3fARB(Location, X, Y, Z));
        } else {
            std::cerr << "Parameter " << Name << std::endl;
        }

        // Disable program
        GLCheck(glUseProgramObjectARB(0));
    }
}

void PostFX::SetParameter(const std::string& Name) {
    if (myShaderProgram) {
        // Enable program
        GLCheck(glUseProgramObjectARB(myShaderProgram));

        // Get parameter location and assign it
        GLint Location = glGetUniformLocationARB(myShaderProgram, Name.c_str());
        if (Location != -1) {
            GLCheck(glUniform4fARB(Location, X, Y, Z, W));
        } else {
            std::cerr << "Parameter " << Name << std::endl;
        }
    }
}
// Disable program
GLCheck(glUseProgramObjectARB(0));
}
} // PostFX

void PostFX::SetTexture(const std::string& Name, { // Check that the current texture unit is available
GLint MaxUnits;
GLCheck(glGetIntegerv(GL_MAX_TEXTURE_COORDS_ARB, &MaxUnits));
if (myTextures.size() >= static_cast<std::size_t>(MaxUnits)) {
    std::cerr << "Impossible to use texture " << Name << "!
return;
}

// Make sure the given name is a valid variable
int Location = glGetUniformLocationARB(myShaderProgram, Name.c_str());
if (Location == -1) {
    std::cerr << "Texture " << Name << " not found in post-effect"
return;
}

// Store the texture for later use
myTextures[Name] = Texture ? Texture : &myFrameBuffer;
}
PostFX& PostFX::operator =(const PostFX& Other) {
    PostFX Temp(Other);
    std::swap(myShaderProgram, Temp.myShaderProgram);
    std::swap(myTextures, Temp.myTextures);
    std::swap(myFragmentShader, Temp.myFragmentShader);
    return *this;
}
std::swap(myFrameBuffer, Temp.myFrameBuffer)

return *this;

bool PostFX::CanUsePostFX()
{
    // Make sure we have a valid context
    priv::GraphicsContext Ctx;

    return glewIsSupported("GL_ARB_shading_language_100")
        && glewIsSupported("GL_ARB_shader_objects")
        && glewIsSupported("GL_ARB_vertex_shader")
        && glewIsSupported("GL_ARB_fragment_shader");
}

void PostFX::Render(RenderTarget& Target) const
{
    // Check that we have a valid program
    if (!myShaderProgram)
        return;

    // Copy the current framebuffer pixels to our frame buffer texture
    // The ugly cast is temporary until PostFx are rewritten :)
    myFrameBuffer.CopyScreen((RenderWindow&)Target);

    // Enable program
    GLCheck(glUseProgramObjectARB(myShaderProgram));

    // Bind textures
    TextureTable::const_iterator ItTex = myTextures.begin();
    for (std::size_t i = 0; i < myTextures.size(); ++i)
    {
        int Location = glGetUniformLocationARB(myShaderProgram, ItTex->first.c_str());
        GLCheck(glUniform1iARB(Location, static_cast<GLint>(i)));
    }
}
GLCheck(glActiveTextureARB(static_cast<GLenum>(GL_TEXTURE0_ARB + i)));
ItTex->second->Bind();
ItTex++;
}

// Compute the texture coordinates (in case the texture is larger than the screen, or flipped)
IntRect FrameBufferRect(0, 0, myFrameBuffer.
FloatRect TexCoords = myFrameBuffer.GetTexCoords

// Render a fullscreen quad using the effect on our framebuffer
FloatRect Screen = Target.GetView().GetRect();

glBegin(GL_QUADS);
    glTexCoord2f(TexCoords.Left, TexCoords;
    glTexCoord2f(TexCoords.Right, TexCoords;
    glTexCoord2f(TexCoords.Right, TexCoords;
    glTexCoord2f(TexCoords.Left, TexCoords;
    glEnd();

// Disable program
GLCheck(glUseProgramObjectARB(0));

// Disable texture units
for (std::size_t i = 0; i < myTextures.size(); ++i)
{
    GLCheck(glActiveTextureARB(static_cast<GLenum>(GL_TEXTURE0_ARB + i)));
    GLCheck(glBindTexture(GL_TEXTURE_2D, 0));
}
GLCheck(glActiveTextureARB(GL_TEXTURE0_ARB));

std::string PostFX::PreprocessEffect(std::istream& File)
{
    // Initialize output string
    std::set<std::string> myTextures;
    std::string Out = "";

// Variable declarations
std::string Line;

while (std::getline(File, Line) && (Line.substr(0, 6) != {
    // Remove the ending '\r', if any
    if (!Line.empty() && (Line[Line.size() - 1] == Line.erase(Line.size() - 1));

    // Skip empty lines
    if (Line == "")
        continue;

    // Extract variables type and name and convert them
    std::string Type, Name;
    std::istringstream iss(Line);
    if (!(iss >> Type >> Name))
        {
            std::cerr << "Post-effect error : in
            " << Type << "> " << Line << std::endl;
            return "";
        }

    if (Type == "texture")
    {
        // Textures need some checking and conversion
        if (myTextures.find(Name) != myTextures.end())
            {
                std::cerr << "Post-effect error : texture " << Name << std::endl;
                return "";
            }

        myTextures.insert(Name);
        Out += "uniform sampler2D " + Name +
        myTextures.insert(Name);
    }

    else
    {
        // Other types are just copied to output
    }
Out += "uniform " + Type + " " + Name + "
"
}

// Effect code
Out += "void main()\n";
while (std::getline(File, Line))
{
    // Replace any texture lookup "T(" by "texture2D(T,
    for (std::set<std::string>::const_iterator i = myTextures.begin(); i != myTextures.end(); ++i)
    {
        std::string::size_type Pos = Line.find(*i);
        if (Pos != std::string::npos)
            Line.replace(Pos, i->size() + 1,
    }

    // Replace "_in" by "gl_TexCoord[0].xy"
    for (std::string::size_type Pos = Line.find("_in"),
    Line.replace(Pos, 3, "gl_TexCoord[0].xy",
    }

    // Replace "_out" by "gl_FragColor"
    for (std::string::size_type Pos = Line.find("_out"),
    Line.replace(Pos, 4, "gl_FragColor",
    }

    // Write modified line to output string
    Out += Line + "\n";
}
return Out;

PostFX::CreateProgram()
{
    // Make sure we have a valid context

    priv::GraphicsContext Ctx;

// Check that we can use post-FX!
if (!CanUsePostFX())
{
    std::cerr << "Failed to create a PostFX:
return;
}

// Destroy effect program if it was already created
if (myShaderProgram)
    GLCheck(glDeleteObjectARB(myShaderProgram));

// Define vertex shader source (we provide it directly as it doesn't have to change)
static const std::string VertexShaderSrc =
    "void main()
    {
        gl_TexCoord[0] = gl_MultiTexCoord0;
        gl_Position = ftransform();
    }";

// Create the program
myShaderProgram = glCreateProgramObjectARB();

// Create the shaders
GLhandleARB VertexShader = glCreateShaderObjectARB(GL_VERTEX_SHADER_ARB);
GLhandleARB FragmentShader = glCreateShaderObjectARB(GL_FRAGMENT_SHADER_ARB);

// Compile them
const char* VertexSrc = VertexShaderSrc.c_str();
const char* FragmentSrc = myFragmentShader.c_str();
GLCheck(glShaderSourceARB(VertexShader, 1, &VertexSrc, NULL));
GLCheck(glShaderSourceARB(FragmentShader, 1, &FragmentSrc, NULL));
GLCheck(glCompileShaderARB(VertexShader));
GLCheck(glCompileShaderARB(FragmentShader));

// Check the compile logs
GLint Success;
GLCheck(glGetObjectParameterivARB(VertexShader, GL_OBJECT_COMPILE_STATUS_ARB, &Success));
if (Success == GL_FALSE) {
    char CompileLog[1024];
    GLCheck(glGetInfoLogARB(VertexShader, sizeof(CompileLog), &Success));
    std::cerr << "Failed to compile post-effect:
    " << CompileLog << std::endl;
    GLCheck(glDeleteObjectARB(VertexShader));
    GLCheck(glDeleteObjectARB(FragmentShader));
    GLCheck(glDeleteObjectARB(myShaderProgram));
    myShaderProgram = 0;
    return;
}

GLCheck(glGetObjectParameterivARB(FragmentShader, GL_OBJECT_COMPILE_STATUS_ARB, &Success));
if (Success == GL_FALSE) {
    char CompileLog[1024];
    GLCheck(glGetInfoLogARB(FragmentShader, sizeof(CompileLog), &Success));
    std::cerr << "Failed to compile post-effect:
    " << CompileLog << std::endl;
    GLCheck(glDeleteObjectARB(VertexShader));
    GLCheck(glDeleteObjectARB(FragmentShader));
    GLCheck(glDeleteObjectARB(myShaderProgram));
    myShaderProgram = 0;
    return;
}

// Attach the shaders to the program
GLCheck(glAttachObjectARB(myShaderProgram, VertexShader, GL_VERTEX_SHADER_ARB));
GLCheck(glAttachObjectARB(myShaderProgram, FragmentShader, GL_FRAGMENT_SHADER_ARB));

// We can now delete the shaders
GLCheck(glDeleteObjectARB(VertexShader));
GLCheck(glDeleteObjectARB(FragmentShader));

// Link the program
GLCheck(glLinkProgramARB(myShaderProgram));
// Get link log
GLCheck(glGetObjectParameterivARB(myShaderProgram,
GL_OBJECT_LINK_STATUS_ARB,
&Success));
if (Success == GL_FALSE)
{
    // Oops... link errors
    char LinkLog[1024];
    GLCheck(glGetInfoLogARB(myShaderProgram,
    std::cerr << "Failed to link post-effect"
    << LinkLog << std::endl;
    GLCheck(glDeleteObjectARB(myShaderProgram);
    myShaderProgram = 0;
    return;
}
} // namespace sf
<table>
<thead>
<tr>
<th>Main Page</th>
<th>Namespaces</th>
<th>Classes</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>File List</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
#ifndef SFML_RANDOMIZER_HPP
#define SFML_RANDOMIZER_HPP

// Headers
#include <SFML/Config.hpp>

namespace sf
{
    class SFML_API Randomizer
    {
    
}
public:

static void SetSeed(unsigned int Seed);
static unsigned int GetSeed();
static float Random(float Begin, float End);
static int Random(int Begin, int End);

private:

// Static member variables
static unsigned int ourSeed;

} // namespace sf

#endif // SFML_RANDOMIZER_HPP
**sf::Randomizer Member List**

This is the complete list of members for **sf::Randomizer**, including all inherited members.

<table>
<thead>
<tr>
<th>Method</th>
<th>sf::Randomizer [static]</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetSeed()</td>
<td></td>
</tr>
<tr>
<td>Random(float Begin, float End)</td>
<td>sf::Randomizer [static]</td>
</tr>
<tr>
<td>Random(int Begin, int End)</td>
<td>sf::Randomizer [static]</td>
</tr>
<tr>
<td>SetSeed(unsigned int Seed)</td>
<td>sf::Randomizer [static]</td>
</tr>
</tbody>
</table>

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any expressed or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.

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

// Headers
#include <SFML/System/Randomizer.hpp>
#include <stdlib.h>
#include <time.h>

namespace
{
    // Set the random numbers sequence seed with:
    unsigned int SetRandomSeed()
    {

unsigned int Seed = static_cast<unsigned int>(srand(Seed));
return Seed;
}

namespace sf
{

// Static member variables
unsigned int Randomizer::ourSeed = SetRandomSeed();

void Randomizer::SetSeed(unsigned int Seed)
{
    srand(Seed);
    ourSeed = Seed;
}

unsigned int Randomizer::GetSeed()
{
    return ourSeed;
}

float Randomizer::Random(float Begin, float End)
{
    // This is not the best algorithm, but it is
    // (see Google for best approaches)
    return static_cast<float>(rand()) / RAND_MAX;
}

int Randomizer::Random(int Begin, int End)
{
This is not the best algorithm, but it is fast and will be enough in most cases (see Google for best approaches)

```c++
return rand() % (End - Begin + 1) + Begin;
}
```

} // namespace sf

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by Doxygen 1.5.2 ::
//
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)
//
// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.
//
// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:
//
// 1. The origin of this software must not be misrepresented;
// you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is not required.
// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.
// 3. This notice may not be removed or altered from any source distribution.
#
#ifndef SFML_RECT_HPP
#define SFML_RECT_HPP

namespace sf
{

    template<typename T>
    class Rect
public:

Rect();

Rect(T LeftCoord, T TopCoord, T RightCoord, T BottomCoord);

T GetWidth() const;

T GetHeight() const;

void Offset(T OffsetX, T OffsetY);

bool Contains(T X, T Y) const;

bool Intersects(const Rect<T>& Rectangle, Rect<T>&

// Member data

T Left;

T Top;

T Right;

T Bottom;

};

#include <SFML/Graphics/Rect.inl>

// Define the most common types

typedef Rect<int> IntRect;

typedef Rect<float> FloatRect;

} // namespace sf

#endif // SFML_RECT_HPP

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
This is the complete list of members for `sf::Rect< T >`, including all inherited members.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Bottom</code></td>
<td>sf::Rect&lt; T &gt;</td>
</tr>
<tr>
<td><code>Contains(T X, T Y) const</code></td>
<td>sf::Rect&lt; T &gt;</td>
</tr>
<tr>
<td><code>GetHeight() const</code></td>
<td>sf::Rect&lt; T &gt;</td>
</tr>
<tr>
<td><code>GetWidth() const</code></td>
<td>sf::Rect&lt; T &gt;</td>
</tr>
<tr>
<td><code>Intersects(const Rect&lt; T &gt; &amp;Rectangle, Rect&lt; T &gt; *OverlappingRect=NULL) const</code></td>
<td>sf::Rect&lt; T &gt;</td>
</tr>
<tr>
<td><code>Left</code></td>
<td>sf::Rect&lt; T &gt;</td>
</tr>
<tr>
<td><code>Offset(T OffsetX, T OffsetY)</code></td>
<td>sf::Rect&lt; T &gt;</td>
</tr>
<tr>
<td><code>Rect()</code></td>
<td>sf::Rect&lt; T &gt;</td>
</tr>
<tr>
<td><code>Rect(T LeftCoord, T TopCoord, T RightCoord, T BottomCoord)</code></td>
<td>sf::Rect&lt; T &gt;</td>
</tr>
<tr>
<td><code>Right</code></td>
<td>sf::Rect&lt; T &gt;</td>
</tr>
<tr>
<td><code>Top</code></td>
<td>sf::Rect&lt; T &gt;</td>
</tr>
</tbody>
</table>
SFML - Simple and Fast Multimedia Library
Copyright (C) 2007-2009 Laurent Gomila (laurent.gomila@gmail.com)

This software is provided 'as-is', without any express or implied warranty.
In no event will the authors be held liable for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,
including commercial applications, and to alter it and redistribute it freely,
subject to the following restrictions:

1. The origin of this software must not be misrepresented;
you must not claim that you wrote the original software.
If you use this software in a product, an acknowledgment
in the product documentation would be appreciated but is not required.

2. Altered source versions must be plainly marked as such,
and must not be misrepresented as being the original software.

3. This notice may not be removed or altered.

#ifndef SFML_RENDERTARGET_HPP
#define SFML_RENDERTARGET_HPP

namespace sf {

#include <SFML/Graphics/Color.hpp>
#include <SFML/Graphics/View.hpp>
#include <SFML/Graphics/Rect.hpp>

namespace sf {

#ifdef SFML_RENDERTARGET_HPP

#endif

} // namespace sf


class Drawable;

class SFML_API RenderTarget
{
    public :
        virtual ~RenderTarget();

        void Clear(const Color& FillColor = Color(0, 0, 0));

        virtual void Draw(const Drawable& Object);

        virtual unsigned int GetWidth() const = 0;

        virtual unsigned int GetHeight() const = 0;

        void SetView(const View& NewView);

        const View& GetView() const;

        View& GetDefaultView();

        void PreserveOpenGLStates(bool Preserve);

    protected :
       RenderTarget();

        void Initialize();

    private :
        virtual bool Activate(bool Active) = 0;

        void SetRenderStates();

        // Member data
View myDefaultView;
const View* myCurrentView;
bool myPreserveStates;
bool myIsDrawing;
}

// namespace sf
}

#endif // SFML_RENDERTARGET_HPP
sf::RenderTarget Member List

This is the complete list of members for `sf::RenderTarget`, including all inherited members.

<table>
<thead>
<tr>
<th>Function</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clear(const Color &amp;FillColor=Color(0, 0, 0))</td>
<td><code>sf::RenderTarget</code></td>
</tr>
<tr>
<td>Draw(const Drawable &amp;Object)</td>
<td><code>sf::RenderTarget</code> [virtual]</td>
</tr>
<tr>
<td>GetDefaultView()</td>
<td><code>sf::RenderTarget</code></td>
</tr>
<tr>
<td>GetHeight() const =0</td>
<td><code>sf::RenderTarget</code> [pure virtual]</td>
</tr>
<tr>
<td>GetView() const</td>
<td><code>sf::RenderTarget</code></td>
</tr>
<tr>
<td>GetWidth() const =0</td>
<td><code>sf::RenderTarget</code> [pure virtual]</td>
</tr>
<tr>
<td>Initialize()</td>
<td><code>sf::RenderTarget</code> [protected]</td>
</tr>
<tr>
<td>PreserveOpenGLStates(bool Preserve)</td>
<td><code>sf::RenderTarget</code></td>
</tr>
<tr>
<td>RenderTarget()</td>
<td><code>sf::RenderTarget</code> [protected]</td>
</tr>
<tr>
<td>SetView(const View &amp;NewView)</td>
<td><code>sf::RenderTarget</code></td>
</tr>
<tr>
<td>~RenderTarget()</td>
<td><code>sf::RenderTarget</code> [virtual]</td>
</tr>
</tbody>
</table>

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
RenderTarget.cpp

00001 //
00002 //
00003 // SFML - Simple and Fast Multimedia Library
00004 // Copyright (C) 2007-2009 Laurent Gomila (laurent.gomila@gmail.com)
00005 //
00006 // This software is provided 'as-is', without any express or implied warranty.
00007 // In no event will the authors be held liable for any damages arising from the use
00008 //
00009 // Permission is granted to anyone to use this software for any purpose,
00010 // including commercial applications, and to alter it and redistribute it freely,
00011 // subject to the following restrictions:
00012 //
00013 // 1. The origin of this software must not be misrepresented;
00014 // you must not claim that you wrote the original software.
00015 // If you use this software in a product, an acknowledgment
00016 // in the product documentation would be appreciated but is not required.
00017 //
00018 // 2. Altered source versions must be plainly marked as such,
00019 // and must not be misrepresented as being the original software.
00020 //
00021 // 3. This notice may not be removed or altered.
00022 //
00026 // Headers
00028 #include <SFML/Graphics/RenderTarget.hpp>
00029 #include <SFML/Graphics/Drawable.hpp>
00030 #include <SFML/Graphics/GraphicsContext.hpp>
00031 #include <iostream>
00032
00034 namespace sf
00035 {
00039 RenderTarget::RenderTarget() :
00040    myCurrentView  (&myDefaultView),
myPreserveStates(false),
myIsDrawing (false)
{
}
RenderTarget::~RenderTarget()
{
   // Nothing to do
}
RenderTarget::Clear(const Color& FillColor)
{
   if (Activate(true))
   {
      // Clear the frame buffer
      GLCheck(glClearColor(FillColor.r / 255.f, FillColor.g / 255.f, FillColor.b / 255.f, 1.f));
      GLCheck(glClear(GL_COLOR_BUFFER_BIT));
      Activate(false);
   }
}
RenderTarget::Draw(const Drawable& Object)
{
   // Check whether we are called from the outside
   if (!myIsDrawing)
   {
      myIsDrawing = true;
   }
   // Set our target as the current target
   if (Activate(true))
   {
      // Save the current render states and...
   }
}
if (myPreserveStates)
{
    GLCheck(glPushAttrib(GL_COLOR_BUFFER_BIT |
                         GL_TEXTURE_BIT |
                         GL_ENABLE_BIT |
                         GL_VIEWPORT_BIT));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glPushMatrix());
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glPushMatrix());
    SetRenderStates();
}

// Set the window viewport and transforms
GLCheck(glViewport(0, 0, GetWidth(), GetHeight()));
GLCheck(glMatrixMode(GL_PROJECTION));
GLCheck(glMatrixMode(GL_MODELVIEW));

// Let the object draw itself
Object::Draw(*this);

// Restore render states
if (myPreserveStates)
{
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJECTION));
    GLCheck(glMatrixMode(GL_MODELVIEW));
    GLCheck(glMatrixMode(GL_PROJE
void RenderTarget::SetView(const View& NewView) {
    myCurrentView = &NewView;
}

const View& RenderTarget::GetView() const {
    return *myCurrentView;
}

View& RenderTarget::GetDefaultView() {
    return myDefaultView;
}

void RenderTarget::PreserveOpenGLStates(bool Preserve) {
    myPreserveStates = Preserve;
}

void RenderTarget::Initialize() {
    // Set the default rendering states
    SetRenderStates();
    // Setup the default view
    myDefaultView.SetFromRect(FloatRect(0, 0, static_cast<float>(SetView(myDefaultView));
}
```cpp
void RenderTarget::SetRenderStates()
{
    GLCheck(glDisable(GL_ALPHA_TEST));
    GLCheck(glDisable(GL_DEPTH_TEST));
    GLCheck(glDisable(GL_LIGHTING));
}
```

// namespace sf

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
RenderWindow.hpp

// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)
// This software is provided 'as-is', without any expressed or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.
// Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:
// 1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.
// 2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.
// 3. This notice may not be removed or altered.

#ifndef SFML_RENDERWINDOW_HPP
#define SFML_RENDERWINDOW_HPP

namespace sf

#include <SFML/Graphics/Image.hpp>
#include <SFML/Graphics/RenderTarget.hpp>
#include <SFML/Window/Window.hpp>
#include <string>

#endif // SFML_RENDERWINDOW_HPP
class Drawable;

class SFML_API RenderWindow : public Window, public WindowHandle
{
public :
    RenderWindow();
    RenderWindow(VideoMode Mode, const std::string& Title);
    RenderWindow(WindowHandle Handle, const WindowSettings&);

virtual ~RenderWindow();

virtual unsigned int GetWidth() const;
virtual unsigned int GetHeight() const;
Image Capture() const;

sf::Vector2f ConvertCoords(unsigned int WindowX);

private :
    virtual void OnCreate();
    virtual bool Activate(bool Active);
};

// namespace sf

#endif // SFML_RENDERWINDOW_HPP
Documentation generated by doxygen 1.5.2 ::
This is the complete list of members for `sf::RenderWindow`, including all inherited members.

<table>
<thead>
<tr>
<th>Function</th>
<th>Return Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Capture()</code> const</td>
<td><code>sf::RenderWindow</code></td>
</tr>
<tr>
<td><code>Clear(const Color &amp;FillColor=Color(0, 0, 0))</code></td>
<td><code>sf::RenderTarget</code></td>
</tr>
<tr>
<td><code>Close()</code></td>
<td><code>sf::Window</code></td>
</tr>
<tr>
<td><code>ConvertCoords</code> (unsigned int WindowX, unsigned int WindowY, const View *TargetView=NULL) const`</td>
<td><code>sf::RenderWindow</code></td>
</tr>
<tr>
<td>`Create(VideoMode Mode, const std::string &amp;Title, unsigned long WindowStyle=Style::Resize</td>
<td>Style::Close, const WindowSettings &amp;Params=WindowSettings())`</td>
</tr>
<tr>
<td><code>Create(WindowHandle Handle, const WindowSettings &amp;Params=WindowSettings())</code></td>
<td><code>sf::Window</code></td>
</tr>
<tr>
<td><code>Display()</code></td>
<td><code>sf::Window</code></td>
</tr>
<tr>
<td><code>Draw(const Drawable &amp;Object)</code></td>
<td><code>sf::RenderTarget</code> [virtual]</td>
</tr>
<tr>
<td><code>EnableKeyRepeat(bool Enabled)</code></td>
<td><code>sf::Window</code></td>
</tr>
<tr>
<td><code>GetDefaultView()</code></td>
<td><code>sf::RenderTarget</code></td>
</tr>
<tr>
<td><code>GetEvent(Event &amp;EventReceived)</code></td>
<td><code>sf::Window</code></td>
</tr>
<tr>
<td><code>GetFrameTime()</code> const</td>
<td><code>sf::Window</code></td>
</tr>
<tr>
<td><code>GetHeight()</code> const</td>
<td><code>sf::RenderWindow</code> [virtual]</td>
</tr>
<tr>
<td><code>GetInput()</code> const</td>
<td><code>sf::Window</code></td>
</tr>
<tr>
<td><code>GetSettings()</code> const</td>
<td><code>sf::Window</code></td>
</tr>
<tr>
<td><code>GetView()</code> const</td>
<td><code>sf::RenderTarget</code> [virtual]</td>
</tr>
<tr>
<td><code>Initialize()</code></td>
<td><code>sf::RenderTarget</code> [protected]</td>
</tr>
<tr>
<td><code>IsOpened()</code> const</td>
<td><code>sf::Window</code></td>
</tr>
<tr>
<td><code>PreserveOpenGLStates</code> (bool Preserve)`</td>
<td><code>sf::RenderTarget</code></td>
</tr>
<tr>
<td><code>RenderTarget()</code></td>
<td><code>sf::RenderTarget</code> [protected]</td>
</tr>
<tr>
<td><code>RenderWindow()</code></td>
<td><code>sf::RenderWindow</code></td>
</tr>
<tr>
<td>`RenderWindow(VideoMode Mode, const std::string &amp;Title, unsigned long WindowStyle=Style::Resize</td>
<td>Style::Close, const WindowSettings &amp;Params=WindowSettings())`</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>RenderWindow</code></td>
<td>sf::RenderWindow</td>
</tr>
<tr>
<td><code>SetActive(bool Active=true) const</code></td>
<td>sf::Window</td>
</tr>
<tr>
<td><code>SetCursorPosition(unsigned int Left, unsigned int Top)</code></td>
<td>sf::Window</td>
</tr>
<tr>
<td><code>SetFramerateLimit(unsigned int Limit)</code></td>
<td>sf::Window</td>
</tr>
<tr>
<td><code>SetIcon(unsigned int Width, unsigned int Height, const Uint8 *Pixels)</code></td>
<td>sf::Window</td>
</tr>
<tr>
<td><code>SetJoystickThreshold(float Threshold)</code></td>
<td>sf::Window</td>
</tr>
<tr>
<td><code>setPosition(int Left, int Top)</code></td>
<td>sf::Window</td>
</tr>
<tr>
<td><code>setSize(unsigned int Width, unsigned int Height)</code></td>
<td>sf::Window</td>
</tr>
<tr>
<td><code>setView(const View &amp;NewView)</code></td>
<td>sf::RenderTarget</td>
</tr>
<tr>
<td><code>show(bool State)</code></td>
<td>sf::Window</td>
</tr>
<tr>
<td><code>showMouseCursor(bool Show)</code></td>
<td>sf::Window</td>
</tr>
<tr>
<td><code>useVerticalSync(bool Enabled)</code></td>
<td>sf::Window</td>
</tr>
<tr>
<td><code>window()</code></td>
<td>sf::Window</td>
</tr>
<tr>
<td>`window(VideoMode Mode, const std::string &amp;Title, unsigned long WindowStyle=Style::Resize</td>
<td>Style::Close, const WindowSettings &amp;Params=WindowSettings())`</td>
</tr>
<tr>
<td>`window(VideoMode Mode, const std::string &amp;Title, unsigned long WindowStyle=Style::Resize</td>
<td>Style::Close, const WindowSettings &amp;Params=WindowSettings())`</td>
</tr>
<tr>
<td><code>~RenderTarget()</code></td>
<td>sf::RenderTarget</td>
</tr>
<tr>
<td><code>~RenderWindow()</code></td>
<td>sf::RenderWindow</td>
</tr>
<tr>
<td><code>~Window()</code></td>
<td>sf::Window</td>
</tr>
<tr>
<td><code>~WindowListener()</code></td>
<td>sf::WindowListener</td>
</tr>
</tbody>
</table>
RenderWindow.cpp

#include <SFML/Graphics/RenderWindow.hpp>
#include <SFML/Graphics/Drawable.hpp>
#include <SFML/Graphics/Image.hpp>
#include <SFML/Graphics/GraphicsContext.hpp>
#include <iostream>

namespace sf
{
RenderWindow::RenderWindow()
}
{  // Nothing to do
}

RenderWindow::RenderWindow(VideoMode Mode, const 
{  Create(Mode, Title, WindowStyle, Params);
}

RenderWindow::RenderWindow(WindowHandle Handle, 
{  Create(Handle, Params);

RenderWindow::~RenderWindow()
{
    // Nothing to do
}

bool RenderWindow::Activate(bool Active)
{
    // For performances and consistency reasons,
    if (Active)
        return SetActive();
    else
        return true;

unsigned int RenderWindow::GetWidth() const 
{
    return sf::Window::GetWidth();
}
unsigned int RenderWindow::GetHeight() const
{
    return sf::Window::GetHeight();
}

Image RenderWindow::Capture() const
{
    // Get the window dimensions
    const unsigned int Width = GetWidth();
    const unsigned int Height = GetHeight();

    // Set our window as the current target for rendering
    if (SetActive())
    {
        // Make sure we have a valid context
        priv::GraphicsContext Ctx;

        // Get pixels from the backbuffer
        std::vector<Uint8> Pixels(Width * Height);
        Uint8* PixelsPtr = &Pixels[0];
        GLCheck(glReadPixels(0, 0, Width, Height, GL_RGBA, GL_UNSIGNED_BYTE, PixelsPtr));

        // Flip the pixels
        unsigned int Pitch = Width * 4;
        for (unsigned int y = 0; y < Height / 2; ++y)
            std::swap_ranges(PixelsPtr + y * Pitch, PixelsPtr + (y + 1) * Pitch, PixelsPtr + (Height - y - 1) * Pitch);

        // Create an image from the pixel buffer
        return Image(Width, Height, PixelsPtr);
    }
    else
    {
        return Image(Width, Height, Color::White);
    }
}
sf::Vector2f RenderWindow::ConvertCoords(unsigned)
{
    // Use the current view if none has been passed
    if (!TargetView)
        TargetView = &GetView();

    float Left  = TargetView->GetCenter().x - 1;
    float Top   = TargetView->GetCenter().y - 1;
    float Right = TargetView->GetCenter().x + 1;
    float Bottom= TargetView->GetCenter().y + 1;

    return sf::Vector2f(Left + WindowX * (Right - Left)
                              Top + WindowY * (Bottom - Top))
}

void RenderWindow::OnCreate()
{
    // We can now initialize the render target
   RenderTarget::Initialize();
}

} // namespace sf

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.

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

// Headers
#include <SFML/Window/Window.hpp>
#include <SFML/Window/Context.hpp>
#include <SFML/Window/WindowImpl.hpp>
#include <SFML/System/Sleep.hpp>
#include <iostream>

// Private data
namespace
{

const sf::Window* FullscreenWindow = NULL;

namespace sf
{
    Window::Window()
    {
    }
    Window::Window(VideoMode Mode, const std::string& Title)
    {
        Create(Mode, Title, WindowStyle, Params);
    }
    Window::Window(WindowHandle Handle, const WindowSettings
    {
    }
    Window::Window(WindowHandle Handle, const Window

Create(Handle, Params);

Window::~Window()
{
    // Close the window
    Close();
}

void Window::Create(VideoMode Mode, const std::string& Title)
{
    // Destroy the previous window implementation
    Close();

    // Fullscreen style requires some tests
    if (WindowStyle & Style::Fullscreen)
    {
        // Make sure there's not already a fullscreen window
        if (FullscreenWindow)
        {
            std::cerr << "Creating two fullscreen windows is not allowed, switching to windowed mode"
        }
        else
        {
            // Make sure the chosen video mode is valid
            if (!Mode.IsValid())
            {
                std::cerr << "The requested video mode is not available, switching to a valid mode"
                Mode = VideoMode::GetMode(0);
            }
        }
    }
    else
    {
        // Update the fullscreen window
        FullscreenWindow = this;
    }
}
// Check validity of style
if ((WindowStyle & Style::Close) || (WindowStyle & Style::Titlebar));

// Activate the global context
Context::GetGlobal().SetActive(true);

mySettings = Params;
Initialize(priv::WindowImpl::New(Mode, Title, WindowStyle, mySettings));

void Window::Create(WindowHandle Handle, const WindowSettings &Params)
{
    // Destroy the previous window implementation
    Close();

    // Activate the global context
    Context::GetGlobal().SetActive(true);

    mySettings = Params;
    Initialize(priv::WindowImpl::New(Handle, mySettings));
}

void Window::Close()
{
    // Delete the window implementation
    delete myWindow;
    myWindow = NULL;

    // Update the fullscreen window
    if (this == FullscreenWindow)
        FullscreenWindow = NULL;
bool Window::IsOpened() const
{
    return myWindow != NULL;
}

unsigned int Window::GetWidth() const
{
    return myWindow ? myWindow->GetWidth() : 0;
}

unsigned int Window::GetHeight() const
{
    return myWindow ? myWindow->GetHeight() : 0;
}

const WindowSettings& Window::GetSettings() const
{
    return mySettings;
}

bool Window::GetEvent(Event& EventReceived)
{
    // Let the window implementation process incoming events if the queue is empty
    if (myWindow && myEvents.empty())
        myWindow->DoEvents();

    // Pop first event of queue, if not empty
    if (!myEvents.empty())
    {
        EventReceived = myEvents.front();
    }
myEvents.pop();

return true;

return false;

void Window::UseVerticalSync(bool Enabled)
{
    if (SetActive())
        myWindow->UseVerticalSync(Enabled);
}

void Window::ShowMouseCursor(bool Show)
{
    if (myWindow)
        myWindow->ShowMouseCursor(Show);
}

void Window::SetCursorPosition(unsigned int Left, int Top)
{
    if (myWindow)
    {
        // Keep coordinates for later checking (}
        mySetCursorPosX = Left;
        mySetCursorPosY = Top;
        myWindow->SetCursorPosition(Left, Top);
    }
}

void Window::SetPosition(int Left, int Top)
if (!myIsExternal)
{
    if (myWindow)
        myWindow->SetPosition(Left, Top);
    else
        {
            std::cerr << "Warning: trying to change the position of an external SFML window, which is not allowed"
        }
}

void Window::SetSize(unsigned int Width, unsigned int Height)
{
    if (myWindow)
        myWindow->SetSize(Width, Height);
}

void Window::Show(bool State)
{
    if (!myIsExternal)
    {
        if (myWindow)
            myWindow->Show(State);
    }
}

void Window::EnableKeyRepeat(bool Enabled)
{
    if (myWindow)
        myWindow->EnableKeyRepeat(Enabled);
}
void Window::SetIcon(unsigned int Width, unsigned int Height, Pixels)
{
    if (myWindow)
        myWindow->SetIcon(Width, Height, Pixels);
}

bool Window::SetActive(bool Active) const
{
    if (myWindow)
    {
        myWindow->SetActive(Active);
        return true;
    }
    return false;
}

void Window::Display()
{
    // Limit the framerate if needed
    if (myFramerateLimit > 0)
    {
        float RemainingTime = 1.f / myFramerateLimit;
        if (RemainingTime > 0)
            Sleep(RemainingTime);
    }
    // Measure the time elapsed since last frame
    myLastFrameTime = myClock.GetElapsedTime();
    myClock.Reset();
    // Display the backbuffer on screen
    if (SetActive())
        myWindow->Display();
}
const Input& Window::GetInput() const
{
    return myInput;
}

void Window::SetFramerateLimit(unsigned int Limit)
{
    myFramerateLimit = Limit;
}

float Window::GetFrameTime() const
{
    return myLastFrameTime;
}

void Window::SetJoystickThreshold(float Threshold)
{
    if (myWindow)
        myWindow->SetJoystickThreshold(Threshold);
}

void Window::OnCreate()
{
    // Nothing by default
}

void Window::OnEvent(const Event& EventReceived)
{
    // Discard MouseMove events generated by SetCursorPosition
    if ((EventReceived.Type == Event::MouseMove)
        }
(EventReceived.MouseMove.X == mySetCursorPosX)
(EventReceived.MouseMove.Y == mySetCursorPosY))
{
    mySetCursorPosX = 0xFFFF;
    mySetCursorPosY = 0xFFFF;
    return;
}
myEvents.push(EventReceived);
}

void Window::Initialize(priv::WindowImpl* Window)
{
    // Assign and initialize the new window
    myWindow = Window;
    myWindow->Initialize();

    // Clear the event queue
    while (!myEvents.empty())
    {
        myEvents.pop();
    }

    // Listen to events from the new window
    myWindow->AddListener(this);
    myWindow->AddListener(&myInput);

    // Setup default behaviours (to get a consistent behaviour across different implementations)
    Show(true);
    UseVerticalSync(false);
    ShowMouseCursor(true);
    EnableKeyRepeat(true);

    // Reset frame time
    myClock.Reset();
    myLastFrameTime = 0.f;

    // Activate the window
SetActive(true);

// Notify the derived class OnCreate();
} // namespace sf
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any
// express or implied warranty. In no event will the authors be held liable for
// any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an
//    acknowledgment in the product documentation would be appreciated but
//    is not required.

// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.

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

#ifndef SFML_RESOURCE_HPP
#define SFML_RESOURCE_HPP

// Headers
#include <set>

namespace sf
{

// These two classes are defined in the same header,
// they depend on each other. And as they're template

// they must be entirely defined in header files,
// prevents from proper separate compiling

template <typename> class ResourcePtr;

template <typename T>
class Resource
{
    protected:
        Resource();
        Resource(const Resource<T>& Copy);
        ~Resource();
        Resource<T>& operator=(const Resource<T>& Other);
    private:
        friend class ResourcePtr<T>;
        void Connect(ResourcePtr<T>& Observer) const;
        void Disconnect(ResourcePtr<T>& Observer) const;
        // Member data
        mutable std::set<ResourcePtr<T>*>* myObservers;
};

template <typename T>
class ResourcePtr
{
    public:
        ResourcePtr();
};
ResourcePtr(const T* Resource);
ResourcePtr(const ResourcePtr<T>& Copy);
~ResourcePtr();
ResourcePtr<T>& operator=(const ResourcePtr<T>&);
ResourcePtr<T>& operator=(const T* Resource);
operator const T*() const;
const T& operator*() const;
const T* operator->() const;
void OnResourceDestroyed();

private:
   // Member data
   const T* myResource;

#include <SFML/System/Resource.inl>
#include <SFML/System/ResourcePtr.inl>
} // namespace sf

#endif // SFML_RESOURCE_HPP
# sf::Resource< T > Member List

This is the complete list of members for `sf::Resource< T >`, including all inherited members.

<table>
<thead>
<tr>
<th>Member</th>
<th>sf::Resource&lt; T &gt;</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>operator=(const Resource&lt; T &gt; &amp;Other)</code></td>
<td>sf::Resource&lt; T &gt;</td>
<td>[protected]</td>
</tr>
<tr>
<td><code>Resource()</code></td>
<td>sf::Resource&lt; T &gt;</td>
<td>[protected]</td>
</tr>
<tr>
<td><code>Resource(const Resource&lt; T &gt; &amp;Copy)</code></td>
<td>sf::Resource&lt; T &gt;</td>
<td>[protected]</td>
</tr>
<tr>
<td><code>ResourcePtr&lt; T &gt;</code> (defined in sf::Resource&lt; T &gt;)</td>
<td>sf::Resource&lt; T &gt;</td>
<td>[friend]</td>
</tr>
<tr>
<td><code>~Resource()</code></td>
<td>sf::Resource&lt; T &gt;</td>
<td>[protected]</td>
</tr>
</tbody>
</table>

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
<table>
<thead>
<tr>
<th>Main Page</th>
<th>Namespaces</th>
<th>Classes</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class List</td>
<td>Class Index</td>
<td>Class Hierarchy</td>
<td>Class Members</td>
</tr>
</tbody>
</table>
This is the complete list of members for `sf::ResourcePtr< T >`, including all inherited members.

<table>
<thead>
<tr>
<th>Function</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>OnResourceDestroyed()</code></td>
<td><code>sf::ResourcePtr&lt; T &gt;</code></td>
</tr>
<tr>
<td><code>operator const T *()</code> const</td>
<td><code>sf::ResourcePtr&lt; T &gt;</code></td>
</tr>
<tr>
<td><code>operator*()</code> const</td>
<td><code>sf::ResourcePtr&lt; T &gt;</code></td>
</tr>
<tr>
<td><code>operator-&gt;()</code> const</td>
<td><code>sf::ResourcePtr&lt; T &gt;</code></td>
</tr>
<tr>
<td><code>operator=(const ResourcePtr&lt; T &gt; &amp;Other)</code></td>
<td><code>sf::ResourcePtr&lt; T &gt;</code></td>
</tr>
<tr>
<td><code>operator=(const T *Resource)</code></td>
<td><code>sf::ResourcePtr&lt; T &gt;</code></td>
</tr>
<tr>
<td><code>ResourcePtr()</code></td>
<td><code>sf::ResourcePtr&lt; T &gt;</code></td>
</tr>
<tr>
<td><code>ResourcePtr(const T *Resource)</code></td>
<td><code>sf::ResourcePtr&lt; T &gt;</code></td>
</tr>
<tr>
<td><code>ResourcePtr(const ResourcePtr&lt; T &gt; &amp;Copy)</code></td>
<td><code>sf::ResourcePtr&lt; T &gt;</code></td>
</tr>
<tr>
<td><code>~ResourcePtr()</code></td>
<td><code>sf::ResourcePtr&lt; T &gt;</code></td>
</tr>
</tbody>
</table>
//SFML - Simple and Fast Multimedia Library
//Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)
//This software is provided 'as-is', without any express or implied warranty.
//In no event will the authors be held liable for any damages arising from the use of this software.
//Permission is granted to anyone to use this software for any purpose,
including commercial applications, and to alter it and redistribute it freely,
subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
//you must not claim that you wrote the original software.
//If you use this software in a product, an acknowledgment
//in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
//and must not be misrepresented as being the original software.

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

#ifndef SFML_SELECTOR_HPP
#define SFML_SELECTOR_HPP

// Headers
#include <SFML/Network/SocketUDP.hpp>
#include <SFML/Network/SocketTCP.hpp>
#include <SFML/Network/SelectorBase.hpp>
#include <map>

namespace sf

template <typename Type>
class Selector : private SelectorBase {
public:
    void Add(Type Socket);
    void Remove(Type Socket);
    void Clear();
    unsigned int Wait(float Timeout = 0.f);

    Type GetSocketReady(unsigned int Index);

private:
    // Types
    typedef std::map<SocketHelper::SocketType, Type> SocketTable;
    // Member data
    SocketTable mySockets;
};

#include <SFML/Network/Selector.inl>

// Let's define the two only valid types of Selector
typedef Selector<SocketUDP> SelectorUDP;
typedef Selector<SocketTCP> SelectorTCP;

} // namespace sf

#define SFML_SELECTOR_HPP
sf::Selector< Type > Member List

This is the complete list of members for sf::Selector< Type >, including all inherited members.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add(Type Socket)</td>
<td>sf::Selector&lt; Type &gt;</td>
</tr>
<tr>
<td>sf::SelectorBase::Add(SocketHelper::SocketType Socket)</td>
<td>sf::SelectorBase [private]</td>
</tr>
<tr>
<td>Clear()</td>
<td>sf::Selector&lt; Type &gt;</td>
</tr>
<tr>
<td>GetSocketReady(unsigned int Index)</td>
<td>sf::Selector&lt; Type &gt;</td>
</tr>
<tr>
<td>Remove(Type Socket)</td>
<td>sf::Selector&lt; Type &gt;</td>
</tr>
<tr>
<td>sf::SelectorBase::Remove(SocketHelper::SocketType Socket)</td>
<td>sf::SelectorBase [private]</td>
</tr>
<tr>
<td>SelectorBase()</td>
<td>sf::SelectorBase [private]</td>
</tr>
<tr>
<td>Wait(float Timeout=0.f)</td>
<td>sf::Selector&lt; Type &gt;</td>
</tr>
</tbody>
</table>

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
<table>
<thead>
<tr>
<th>Main Page</th>
<th>Namespaces</th>
<th>Classes</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>File List</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gomila@gmail.com)
//
// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.
//
// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:
//
// 1. The origin of this software must not be misrepresented;
// you must not claim that you wrote the original software.
// If you use this software in a product, an acknowledgment
// in the product documentation would be appreciated but is not required.
// 2. Altered source versions must be plainly marked as such,
// and must not be misrepresented as being the original software.
// 3. This notice may not be removed or altered.

#ifndef SFML_SELECTORBASE_HPP
#define SFML_SELECTORBASE_HPP

// Headers
#include <SFML/Config.hpp>
#include <SFML/Network/SocketHelper.hpp>
#include <map>

namespace sf
{


class SFML_API SelectorBase
{
  public :
  SelectorBase();

  void Add(SocketHelper::SocketType Socket);
  void Remove(SocketHelper::SocketType Socket);
  void Clear();

  unsigned int Wait(float Timeout = 0.f);

  SocketHelper::SocketType GetSocketReady(unsigned);

private :
  // Member data
  fd_set mySet;
  fd_set mySetReady;
  int myMaxSocket;
};

} // namespace sf

#endif // SFML_SELECTORBASE_HPP
sf::SelectorBase Member List

This is the complete list of members for `sf::SelectorBase`, including all inherited members.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Add</code></td>
<td>(SocketHelper::SocketType Socket)</td>
</tr>
<tr>
<td><code>Clear</code></td>
<td>()</td>
</tr>
<tr>
<td><code>GetSocketReady</code></td>
<td>(unsigned int Index)</td>
</tr>
<tr>
<td><code>Remove</code></td>
<td>(SocketHelper::SocketType Socket)</td>
</tr>
<tr>
<td><code>SelectorBase</code></td>
<td>()</td>
</tr>
<tr>
<td><code>Wait</code></td>
<td>(float Timeout=0.f)</td>
</tr>
</tbody>
</table>

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.

// 3. This notice may not be removed or altered.

// Headers
#include <SFML/Network/SelectorBase.hpp>
namespace sf
{
    SelectorBase::SelectorBase():

#ifdef _MSC_VER
#pragma warning(disable : 4127) // "conditional expression is constant"
#endif
    // Headers
#include <SFML/Network/SelectorBase.hpp>
namespace sf
{
    SelectorBase::SelectorBase():

#ifdef _MSC_VER
#pragma warning(disable : 4127) // "conditional expression is constant"
#endif
}
myMaxSocket(0)
{
    Clear();
}

void SelectorBase::Add(SocketHelper::SocketType Socket)
{
    FD_SET(Socket, &mySet);

    int Size = static_cast<int>(Socket);
    if (Size > myMaxSocket)
        myMaxSocket = Size;
}

void SelectorBase::Remove(SocketHelper::SocketType Socket)
{
    FD_CLR(Socket, &mySet);
}

void SelectorBase::Clear()
{
    FD_ZERO(&mySet);
    FD_ZERO(&mySetReady);
    myMaxSocket = 0;
}

unsigned int SelectorBase::Wait(float Timeout)
{
    timeval Time;
    Time.tv_sec = static_cast<long>(Timeout);
    Time.tv_usec = (static_cast<long>(Timeout *
// Prepare the set of sockets to return
mySetReady = mySet;

// Wait until one of the sockets is ready for reading, or timeout is reached
int NbSockets = select(myMaxSocket + 1, &mySetReady, NULL, NULL, Timeout > 0 ? &Time : NULL);
return NbSockets >= 0 ? static_cast<unsigned>(SocketHelper::SocketType)SelectorBase::GetSocketReady(Index) :
SocketHelper::InvalidSocket();
SFML - Simple and Fast Multimedia Library
Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

This software is provided 'as-is', without any express or implied warranty.
In no event will the authors be held liable for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software.
   If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.

2. Altered source versions must be plainly marked as such and must not be misrepresented as being the original software.

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

#ifndef SFML_SHAPE_HPP
#define SFML_SHAPE_HPP

namespace sf {

#include <SFML/Graphics/Drawable.hpp>
#include <SFML/System/Vector2.hpp>
#include <vector>

namespace sf {
class SFML_API Shape : public sf::Drawable
{
public :
    Shape();

    void AddPoint(float X, float Y, const Color& Col);
    void AddPoint(const Vector2f& Position, const Color& Col);

    unsigned int GetNbPoints() const;

    void EnableFill(bool Enable);
    void EnableOutline(bool Enable);

    void SetPointPosition(unsigned int Index, const Vector2f& Position);
    void SetPointPosition(unsigned int Index, float X, float Y);

    void SetPointColor(unsigned int Index, const Color& Color);
    void SetPointOutlineColor(unsigned int Index, const Color& Color);

    void SetOutlineWidth(float Width);

    const Vector2f& GetPointPosition(unsigned int Index) const;
    const Color& GetPointColor(unsigned int Index) const;
    const Color& GetPointOutlineColor(unsigned int Index) const;

    float GetOutlineWidth() const;

    static Shape Line(float P1X, float P1Y, float P2X, float P2Y);
    static Shape Line(const Vector2f& P1, const Vector2f& P2);
static Shape Rectangle(float P1X, float P1Y,
static Shape Rectangle(const Vector2f &P1, 
static Shape Circle(float X, float Y, float 
static Shape Circle(const Vector2f &Center,
protected :
virtual void Render(RenderTarget &Target) const
private :
void Compile();
static bool ComputeNormal(const Vector2f &P1,
struct Point
{
    Point(const Vector2f &Pos = Vector2f(0,
    Vector2f Position;
    Vector2f Normal;
    Color Col;
    Color OutlineCol;
};

    // Member data
std::vector<Point> myPoints;
float myOutline;
bool myIsFillEnabled;
bool myIsOutlineEnabled;
bool myIsCompiled;
};
} // namespace sf
#endif // SFML_SHAPE_HPP

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
sf::Shape Member List

This is the complete list of members for sf::Shape, including all inherited members.

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddPoint(float X, float Y, const Color &amp;Col=Color(255, 255, 255), const Color &amp;OutlineCol=Color(0, 0, 0))</td>
<td>sf::Shape</td>
</tr>
<tr>
<td>AddPoint(const Vector2f &amp;Position, const Color &amp;Col=Color(255, 255, 255), const Color &amp;OutlineCol=Color(0, 0, 0))</td>
<td>sf::Shape</td>
</tr>
<tr>
<td>Circle(float X, float Y, float Radius, const Color &amp;Col, float Outline=0.f, const Color &amp;OutlineCol=sf::Color(0, 0, 0))</td>
<td>sf::Shape [static]</td>
</tr>
<tr>
<td>Circle(const Vector2f &amp;Center, float Radius, const Color &amp;Col, float Outline=0.f, const Color &amp;OutlineCol=sf::Color(0, 0, 0))</td>
<td>sf::Shape [static]</td>
</tr>
<tr>
<td>Drawable(const Vector2f &amp;Position=Vector2f(0, 0), const Vector2f &amp;Scale=Vector2f(1, 1), float Rotation=0.f, const Color &amp;Col=Color(255, 255, 255, 255))</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td>EnableFill(bool Enable)</td>
<td>sf::Shape</td>
</tr>
<tr>
<td>EnableOutline(bool Enable)</td>
<td>sf::Shape</td>
</tr>
<tr>
<td>GetBlendMode() const</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td>GetCenter() const</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td>GetColor() const</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td>GetInverseMatrix() const</td>
<td>sf::Drawable [protected]</td>
</tr>
<tr>
<td>GetMatrix() const</td>
<td>sf::Drawable [protected]</td>
</tr>
<tr>
<td>GetNbPoints() const</td>
<td>sf::Shape</td>
</tr>
<tr>
<td>GetOutlineWidth() const</td>
<td>sf::Shape</td>
</tr>
<tr>
<td>GetPointColor(unsigned int Index) const</td>
<td>sf::Shape</td>
</tr>
<tr>
<td>GetPointOutlineColor(unsigned int Index) const</td>
<td>sf::Shape</td>
</tr>
<tr>
<td>GetPointPosition(unsigned int Index) const</td>
<td>sf::Shape</td>
</tr>
<tr>
<td>GetPosition() const</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td>GetRotation() const</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td>GetScale() const</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td>Line(float P1X, float P1Y, float P2X, float P2Y, float Thickness, const Color &amp;Col, float Outline=0.f, const</td>
<td>sf::Shape [static]</td>
</tr>
<tr>
<td>Function</td>
<td>Type</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td><code>Line(const Vector2f &amp;P1, const Vector2f &amp;P2, float Thickness, const Color &amp;Col, float Outline=0.f, const Color &amp;OutlineCol=sf::Color(0, 0, 0))</code></td>
<td><code>sf::Shape</code></td>
</tr>
<tr>
<td><code>Move(float OffsetX, float OffsetY)</code></td>
<td><code>sf::Drawable</code></td>
</tr>
<tr>
<td><code>Move(const Vector2f &amp;Offset)</code></td>
<td><code>sf::Drawable</code></td>
</tr>
<tr>
<td><code>Rectangle(float P1X, float P1Y, float P2X, float P2Y, const Color &amp;Col, float Outline=0.f, const Color &amp;OutlineCol=sf::Color(0, 0, 0))</code></td>
<td><code>sf::Shape</code></td>
</tr>
<tr>
<td><code>Rectangle(const Vector2f &amp;P1, const Vector2f &amp;P2, const Color &amp;Col, float Outline=0.f, const Color &amp;OutlineCol=sf::Color(0, 0, 0))</code></td>
<td><code>sf::Shape</code></td>
</tr>
<tr>
<td><code>Render(RenderTarget &amp;Target) const</code></td>
<td><code>sf::Shape</code></td>
</tr>
<tr>
<td><code>Rotate(float Angle)</code></td>
<td><code>sf::Drawable</code></td>
</tr>
<tr>
<td><code>Scale(float FactorX, float FactorY)</code></td>
<td><code>sf::Drawable</code></td>
</tr>
<tr>
<td><code>Scale(const Vector2f &amp;Factor)</code></td>
<td><code>sf::Drawable</code></td>
</tr>
<tr>
<td><code>SetBlendMode(Blend::Mode Mode)</code></td>
<td><code>sf::Drawable</code></td>
</tr>
<tr>
<td><code>SetCenter(float CenterX, float CenterY)</code></td>
<td><code>sf::Drawable</code></td>
</tr>
<tr>
<td><code>SetCenter(const Vector2f &amp;Center)</code></td>
<td><code>sf::Drawable</code></td>
</tr>
<tr>
<td><code>SetColor(const Color &amp;Col)</code></td>
<td><code>sf::Drawable</code></td>
</tr>
<tr>
<td><code>SetOutlineWidth(float Width)</code></td>
<td><code>sf::Drawable</code></td>
</tr>
<tr>
<td><code>SetPointColor(unsigned int Index, const Color &amp;Col)</code></td>
<td><code>sf::Shape</code></td>
</tr>
<tr>
<td><code>SetPointOutlineColor(unsigned int Index, const Color &amp;OutlineCol)</code></td>
<td><code>sf::Shape</code></td>
</tr>
<tr>
<td><code>SetPointPosition(unsigned int Index, const Vector2f &amp;Position)</code></td>
<td><code>sf::Shape</code></td>
</tr>
<tr>
<td><code>SetPointPosition(unsigned int Index, float X, float Y)</code></td>
<td><code>sf::Shape</code></td>
</tr>
<tr>
<td><code>setPosition(float X, float Y)</code></td>
<td><code>sf::Drawable</code></td>
</tr>
<tr>
<td><code>setPosition(const Vector2f &amp;Position)</code></td>
<td><code>sf::Drawable</code></td>
</tr>
<tr>
<td><code>setRotation(float Rotation)</code></td>
<td><code>sf::Drawable</code></td>
</tr>
<tr>
<td><code>setScale(float ScaleX, float ScaleY)</code></td>
<td><code>sf::Drawable</code></td>
</tr>
<tr>
<td><code>setScale(const Vector2f &amp;Scale)</code></td>
<td><code>sf::Drawable</code></td>
</tr>
<tr>
<td><code>setScaleX(float FactorX)</code></td>
<td><code>sf::Drawable</code></td>
</tr>
<tr>
<td><code>setScaleY(float FactorY)</code></td>
<td><code>sf::Drawable</code></td>
</tr>
<tr>
<td><code>setX(float X)</code></td>
<td><code>sf::Drawable</code></td>
</tr>
<tr>
<td><code>setY(float Y)</code></td>
<td><code>sf::Drawable</code></td>
</tr>
<tr>
<td>Function</td>
<td>Declaration</td>
</tr>
<tr>
<td>--------------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>Shape()</td>
<td>sf::Shape</td>
</tr>
<tr>
<td>TransformToGlobal(const sf::Vector2f &amp;Point) const</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td>TransformToLocal(const sf::Vector2f &amp;Point) const</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td>~Drawable()</td>
<td>sf::Drawable [virtual]</td>
</tr>
</tbody>
</table>

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)
//
// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.
//
// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:
//
// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is not required.
// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.
// 3. This notice may not be removed or altered from any source distribution.

// Headers
#include <SFML/Graphics/Shape.hpp>
#include <SFML/Graphics/GraphicsContext.hpp>
#include <math.h>

namespace sf
{
    Shape::Shape()
    {
        myOutline       (0.f),
        myIsFillEnabled (true),
    }
}
myIsOutlineEnabled(true),
myIsCompiled (false)
{
    // Put a placeholder for the center of the shape
    myPoints.push_back(Point());
}

void Shape::AddPoint(float X, float Y, const Color Col)
{
    AddPoint(Vector2f(X, Y), Col, OutlineCol);
}

void Shape::AddPoint(const Vector2f& Position, const Color& Col, const Color& OutlineCol)
{
    myPoints.push_back(Point(Position, Col, OutlineCol));
    myIsCompiled = false;
}

unsigned int Shape::GetNbPoints() const
{
    return static_cast<unsigned int>(myPoints.size() - 1);
}

void Shape::EnableFill(bool Enable)
{
    myIsFillEnabled = Enable;
}

void Shape::EnableOutline(bool Enable)
{
    myIsOutlineEnabled = Enable;
}
```cpp
void Shape::SetPointPosition(unsigned int Index, {
    myPoints[Index + 1].Position = Position;
    myIsCompiled = false;
}

void Shape::SetPointPosition(unsigned int Index, {
    SetPointPosition(Index, Vector2f(X, Y));
}

void Shape::SetPointColor(unsigned int Index, const {
    myPoints[Index + 1].Col = Col;
    myIsCompiled = false;
}

void Shape::SetPointOutlineColor(unsigned int Index, {
    myPoints[Index + 1].OutlineCol = OutlineCol;
    myIsCompiled = false;
}

void Shape::SetOutlineWidth(float Width) {
    myOutline = Width;
}

const Vector2f& Shape::GetPointPosition(unsigned int Index) {
```
return myPoints[Index + 1].Position;
}

const Color& Shape::GetPointColor(unsigned int Index) {
    return myPoints[Index + 1].Col;
}

const Color& Shape::GetPointOutlineColor(unsigned int Index) {
    return myPoints[Index + 1].OutlineCol;
}

float Shape::GetOutlineWidth() const {
    return myOutline;
}

Shape Shape::Line(float P1X, float P1Y, float P2X, float P2Y) {
    Vector2f P1(P1X, P1Y);
    Vector2f P2(P2X, P2Y);

    // Compute the extrusion direction
    Vector2f Normal;
    ComputeNormal(P1, P2, Normal);
    Normal *= Thickness / 2;

    // Create the shape's points
    Shape S;
    S.AddPoint(P1 - Normal, Col, OutlineCol);
    S.AddPoint(P2 - Normal, Col, OutlineCol);
    S.AddPoint(P2 + Normal, Col, OutlineCol);
00199  S.AddPoint(P1 + Normal, Col, OutlineCol);
00200  S.SetOutlineWidth(Outline);
00201  // Compile it
00202  S.Compile();
00203  return S;
00204 
00205 }  
00206 
00207  
00208 Shape Shape::Line(const Vector2f& P1, const Vector2f& P2, Thickness, Col, Outline, OutlineCol)  
00212  {  
00213    return Shape::Line(P1.x, P1.y, P2.x, P2.y, Thickness, Col, Outline, OutlineCol);
00214  }
00215  
00216  
00217 Shape Shape::Rectangle(float P1X, float P1Y, float P2X, float P2Y, float Thickness, Col, Outline, OutlineCol)  
00222  {  
00223    // Create the shape's points
00224    Shape S;
00225    S.AddPoint(Vector2f(P1X, P1Y), Col, OutlineCol);
00226    S.AddPoint(Vector2f(P2X, P1Y), Col, OutlineCol);
00227    S.AddPoint(Vector2f(P2X, P2Y), Col, OutlineCol);
00228    S.AddPoint(Vector2f(P1X, P2Y), Col, OutlineCol);
00229    S.SetOutlineWidth(Outline);
00230    // Compile it
00231    S.Compile();
00232    return S;
00233  }
00234  
00235  
00236  
00237 Shape Shape::Rectangle(const Vector2f& P1, const Vector2f& P2, Thickness, Col, Outline, OutlineCol)  
00242  {  
00243    return Shape::Rectangle(P1.x, P1.y, P2.x, P2.y, Thickness, Col, Outline, OutlineCol);
00244  }
Shape::Circle(float X, float Y, float Radius) {
    static const int NbSegments = 40;
    // Create the points set
    Shape S;
    Vector2f Center(X, Y);
    for (int i = 0; i < NbSegments; ++i)
    {
        float Angle = i * 2 * 3.141592654f / NbSegments;
        Vector2f Offset(cos(Angle), sin(Angle));
        S.AddPoint(Center + Offset * Radius, Col, OutlineCol);
    }
    // Compile it
    S.SetOutlineWidth(Outline);
    S.Compile();
    return S;
}

Shape::Circle(const Vector2f& Center, float Radius) {
    return Shape::Circle(Center.x, Center.y, Radius);
}

void Shape::Render(RenderTarget&) const
{
    // Make sure the shape has at least 3 points
    if (myPoints.size() < 4)
        return;
}
// Make sure the shape is compiled
if (!myIsCompiled)
    const_cast<Shape*>(this)->Compile();

// Shapes only use color, no texture
GLCheck(glDisable(GL_TEXTURE_2D));

// Draw the shape
if (myIsFillEnabled)
{
    glBegin(GL_TRIANGLE_FAN);
    {
        for (std::vector<Point>::const_iterator i = myPoints.begin(); i != myPoints.end(); ++i)
            Color PointColor = i->Col * GetColor();
        glColor4f(PointColor.r / 255.f, PointColor.g / 255.f, PointColor.b / 255.f, PointColor.a);
        glVertex2f(i->Position.x, i->Position.y);
    }
    glEnd();
}

// Close the shape by duplicating the first point
Color PointColor = myPoints[1].Col * GetColor();
    glColor4f(PointColor.r / 255.f, PointColor.g / 255.f, PointColor.b / 255.f, PointColor.a);
    glVertex2f(myPoints[1].Position.x, myPoints[1].Position.y);

// Draw the outline
if (myIsOutlineEnabled)
{
    glBegin(GL_TRIANGLE_STRIP);
    {
        for (std::size_t i = 1; i < myPoints.size(); ++i)
            Color PointColor = myPoints[i].OutlineCol * GetColor();
        glColor4f(PointColor.r / 255.f, PointColor.g / 255.f, PointColor.b / 255.f, PointColor.a);
        glVertex2f(myPoints[i].Position.x, myPoints[i].Position.y);
    }
    glEnd();
}
void Shape::Compile()
{
    // Compute the center
    float NbPoints = static_cast<float>(myPoints.size() - 1);
    float R = 0, G = 0, B = 0, A = 0;
    Point Center(Vector2f(0, 0), Color(0, 0, 0, 0));
    for (std::size_t i = 1; i < myPoints.size(); ++i)
    {
        Center.Position += myPoints[i].Position / NbPoints;
        R += myPoints[i].Col.r / NbPoints;
        G += myPoints[i].Col.g / NbPoints;
        B += myPoints[i].Col.b / NbPoints;
        A += myPoints[i].Col.a / NbPoints;
    }
    Center.Col.r = static_cast<Uint8>(R);
    Center.Col.g = static_cast<Uint8>(G);
    Center.Col.b = static_cast<Uint8>(B);
    Center.Col.a = static_cast<Uint8>(A);
    myPoints[0] = Center;
    // Compute the outline
    // Close the shape by duplicating the first point at the end
    Color PointColor = myPoints[1].OutlineCol;
    glColor4f(PointColor.r / 255.f, PointColor.g / 255.f, PointColor.b / 255.f, PointColor.a / 255.f);
    glVertex2f(myPoints[1].Position.x, myPoints[1].Position.y);
    glColor4f(PointColor.r / 255.f, PointColor.g / 255.f, PointColor.b / 255.f, PointColor.a / 255.f);
    glVertex2f(myPoints[1].Position.x + myPoints[1].Normal.x * myOutline, myPoints[1].Position.y + myPoints[1].Normal.y * myOutline);
    glEnd();
}
for (std::size_t i = 1; i < myPoints.size(); ++i) {
  // Get the two segments shared by the current point
  Point& P0 = (i == 1) ? myPoints[myPoints.size() - 1] : myPoints[i - 1];
  Point& P1 = myPoints[i];
  Point& P2 = (i == myPoints.size() - 1) ? myPoints[1] : myPoints[i + 1];

  // Compute their normal
  Vector2f Normal1, Normal2;
  if (!ComputeNormal(P0.Position, P1.Position, Normal1)
      || !ComputeNormal(P1.Position, P2.Position, Normal2))
    continue;

  // Add them to get the extrusion direction
  float Factor = 1.f + (Normal1.x * Normal2.x + Normal1.y * Normal2.y);
  P1.Normal = (Normal1 + Normal2) / Factor;

  // Make sure it points towards the outside
  float Dot = (P1.Position.x - Center.Position.x) * P1.Normal.x
  if (Dot < 0)
    P1.Normal = -P1.Normal;

  myIsCompiled = true;
}

bool Shape::ComputeNormal(const Vector2f& P1, const Vector2f& P2) {
  Normal.x = P1.y - P2.y;
  Normal.y = P2.x - P1.x;

  float Len = sqrt(Normal.x * Normal.x + Normal.y * Normal.y);
  if (Len == 0.f)
    return false;

  Normal.x /= Len;
  Normal.y /= Len;
return true;
}
Shape::Point::Point(const Vector2f& Pos, const Color& C, Position (Pos), Normal (0.f, 0.f), Col (C), OutlineCol(OutlineC) {
} // namespace sf
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gomila@gmail.com)
// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.
// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:
// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an
//    acknowledgment in the product documentation would be appreciated but is not required.
// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.
// 3. This notice may not be removed or altered from any source distribution.
#ifndef SFML_SOCKETHELPERWIN32_HPP
#define SFML_SOCKETHELPERWIN32_HPP

// Headers
#include <winsock2.h>

namespace sf
{
class SFML_API SocketHelper
{
public :

// Define some socket types
typedef SOCKET SocketType;
typedef int LengthType;

static SocketType InvalidSocket();

static bool Close(SocketType Socket);

static void SetBlocking(SocketType Socket, bool);

static Socket::Status GetErrorStatus();

} // namespace sf

#endif // SFML_SOCKETHELPERWIN32_HPP
## sf::SocketHelper Member List

This is the complete list of members for `sf::SocketHelper`, including all inherited members.

<table>
<thead>
<tr>
<th>Function</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Close</strong>(SocketType Socket)</td>
<td>sf::SocketHelper [static]</td>
</tr>
<tr>
<td><strong>GetErrorStatus</strong>()</td>
<td>sf::SocketHelper [static]</td>
</tr>
<tr>
<td><strong>InvalidSocket</strong>()</td>
<td>sf::SocketHelper [static]</td>
</tr>
<tr>
<td><strong>LengthType</strong> typedef (defined in sf::SocketHelper)</td>
<td>sf::SocketHelper</td>
</tr>
<tr>
<td><strong>SetBlocking</strong>(SocketType Socket, bool Block)</td>
<td>sf::SocketHelper [static]</td>
</tr>
<tr>
<td><strong>SocketType</strong> typedef (defined in sf::SocketHelper)</td>
<td>sf::SocketHelper</td>
</tr>
</tbody>
</table>
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.

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

// Headers
#include <SFML/Network/SocketHelper.hpp>

namespace sf
{

SocketHelper::SocketType SocketHelper::InvalidSocket
{
  return INVALID_SOCKET;
}
}
bool SocketHelper::Close(SocketHelper::SocketType Socket) {
    return closesocket(Socket) != -1;
}

void SocketHelper::SetBlocking(SocketHelper::SocketType Socket, bool Block) {
    unsigned long Blocking = Block ? 0 : 1;
    ioctlsocket(Socket, FIONBIO, &Blocking);
}

Socket::Status SocketHelper::GetErrorStatus() {
    switch (WSAGetLastError()) {
    case WSAEWOULDBLOCK : return Socket::NotReady;
    case WSAECONNABORTED : return Socket::Disconnected;
    case WSAECONNRESET : return Socket::Disconnected;
    case WSAETIMEDOUT : return Socket::Disconnected;
    case WSAENETRESET : return Socket::Disconnected;
    case WSAENOTCONN : return Socket::Disconnected;
    default : return Socket::Error;
    }
}

// Windows needs some initialization and cleanup to get sockets working properly... so let's create a class that will do it automatically

struct SocketInitializer {
    SocketInitializer() {
        WSADATA InitData;
    }
};
WSAStartup(MAKEWORD(2,2), &.InitData);

~SocketInitializer()
{
    WSACleanup();
}

SocketInitializer GlobalInitializer;

} // namespace sf
SocketTCP.hpp

// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
subject to the following restrictions:

1. The origin of this software must not be misrepresented;
   you must not claim that you wrote the original software.
   If you use this software in a product, an acknowledgment
   in the product documentation would be appreciated but is not required.

2. Altered source versions must be plainly marked as such,
   and must not be misrepresented as being the original software.

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

#ifndef SFML_SOCKETTCP_HPP
#define SFML_SOCKETTCP_HPP

// Headers
#include <SFML/Network/SocketHelper.hpp>
#include <vector>

namespace sf {

class Packet;
}
#endif // SFML_SOCKETTCP_HPP
class IPAddress;

template<typename> class Selector;

class SFML_API SocketTCP
{
  public :
    SocketTCP();
    void SetBlocking(bool Blocking);
    bool Listen(unsigned short Port);
    bool Close();
    bool IsValid() const;
  private :
    bool operator == (const SocketTCP& Other) const;
    bool operator != (const SocketTCP& Other) const;
    bool operator < (const SocketTCP& Other) const;
    private:
    }
friend class Selector<SocketTCP>;

SocketTCP(SocketHelper::SocketType Descriptor);

void Create(SocketHelper::SocketType Descriptor = 0);

// Member data
SocketHelper::SocketType mySocket;
Uint32 myPendingHeader;
Uint32 myPendingHeaderSize;
std::vector<char> myPendingPacket;
Int32 myPendingPacketSize;
bool myIsBlocking;

};

} // namespace sf

<endif // SFML_SOCKETTCP_HPP>
<table>
<thead>
<tr>
<th>Main Page</th>
<th>Namespaces</th>
<th>Classes</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Class List</td>
<td>Class Index</td>
<td>Class Hierarchy</td>
</tr>
</tbody>
</table>

**SFML**
Simple and Fast Multimedia Library
sf::SocketTCP Member List

This is the complete list of members for sf::SocketTCP, including all inherited members.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accept(SocketTCP &amp;Connected, IPAddress *Address=NULL)</td>
<td>sf::SocketTCP</td>
</tr>
<tr>
<td>Close()</td>
<td>sf::SocketTCP</td>
</tr>
<tr>
<td>Connect(unsigned short Port, const IPAddress &amp;HostAddress, float Timeout=0.f)</td>
<td>sf::SocketTCP</td>
</tr>
<tr>
<td>IsValid() const</td>
<td>sf::SocketTCP</td>
</tr>
<tr>
<td>Listen(unsigned short Port)</td>
<td>sf::SocketTCP</td>
</tr>
<tr>
<td>operator!=(const SocketTCP &amp;Other) const</td>
<td>sf::SocketTCP</td>
</tr>
<tr>
<td>operator&lt;(const SocketTCP &amp;Other) const</td>
<td>sf::SocketTCP</td>
</tr>
<tr>
<td>operator==(const SocketTCP &amp;Other) const</td>
<td>sf::SocketTCP</td>
</tr>
<tr>
<td>Receive(char *Data, std::size_t MaxSize, std::size_t &amp;SizeReceived)</td>
<td>sf::SocketTCP</td>
</tr>
<tr>
<td>Receive(Packet &amp;PacketToReceive)</td>
<td>sf::SocketTCP</td>
</tr>
<tr>
<td>Selector&lt; SocketTCP &gt; (defined in sf::SocketTCP)</td>
<td>sf::SocketTCP [friend]</td>
</tr>
<tr>
<td>Send(const char *Data, std::size_t Size)</td>
<td>sf::SocketTCP</td>
</tr>
<tr>
<td>Send(Packet &amp;PacketToSend)</td>
<td>sf::SocketTCP</td>
</tr>
<tr>
<td>SetBlocking(bool Blocking)</td>
<td>sf::SocketTCP</td>
</tr>
<tr>
<td>SocketTCP()</td>
<td>sf::SocketTCP</td>
</tr>
</tbody>
</table>

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
// you must not claim that you wrote the original software.
// If you use this software in a product, an acknowledgment
// in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
// and must not be misrepresented as being the original software.

// 3. This notice may not be removed or altered

// Headers
#include <SFML/Network/SocketTCP.hpp>
#include <SFML/Network/IPAddress.hpp>
#include <SFML/Network/Packet.hpp>
#include <SFML/Network/SocketHelper.hpp>
#include <algorithm>
#include <iostream>
#include <cstring>

#ifndef _MSC_VER
#pragma warning(disable : 4127) // "conditional expression is constant" generated by the FD_SET macro

namespace sf
{

SocketTCP::SocketTCP()
{
  Create(SocketHelper::InvalidSocket());
}

void SocketTCP::SetBlocking(bool Blocking)
{
  // Make sure our socket is valid
  if (!IsValid())
  
  SocketHelper::SetBlocking(mySocket, Blocking);

  myIsBlocking = Blocking;
}

Socket::Status SocketTCP::Connect(unsigned short
{
  // Make sure our socket is valid
  if (!IsValid())
  
  // Build the host address
  sockaddr_in SockAddr;
  memset(SockAddr.sin_zero, 0, sizeof(SockAddr.sin_zero));
  SockAddr.sin_addr.s_addr = inet_addr(HostAddress.
  SockAddr.sin_family = AF_INET;
  SockAddr.sin_port = htons(Port);

  if (Timeout <= 0)
// ----- We're not using a timeout : just try to connect ----

if (connect(mySocket, reinterpret_cast<sockaddr*>(&SockAddr),
{  // Failed to connect
    return SocketHelper::GetErrorStatus();
}

// Connection succeeded
return Socket::Done;

else{
    // ----- We're using a timeout : we'll need a few tricks to make it work ----
    // Save the previous blocking state
    bool IsBlocking = myIsBlocking;

    // Switch to non-blocking to enable our connection timeout
    if (IsBlocking)
        SetBlocking(false);

    // Try to connect to host
    if (connect(mySocket, reinterpret_cast<sockaddr*>(&SockAddr),
    {  // We got instantly connected! (it may happen a lot...)
        return Socket::Done;
    }

    // Get the error status
    Socket::Status Status = SocketHelper::Get

    // If we were in non-blocking mode, return
    if (!IsBlocking)
        return Status;
// Otherwise, wait until something happens to our socket (success, timeout or error)
if (Status == Socket::NotReady)
{
    // Setup the selector
    fd_set Selector;
    FD_ZERO(&Selector);
    FD_SET(mySocket, &Selector);

    // Setup the timeout
    timeval Time;
    Time.tv_sec = static_cast<long>(Timeout);
    Time.tv_usec = (static_cast<long>(Timeout * 1000) % 1000) * 1000;

    // Wait for something to write on our socket (which means the connection request has returned)
    if (select(static_cast<int>(mySocket + 1), NULL, &Selector, NULL, &Time) > 0)
    {
        // At this point the connection may have been either accepted or refused.
        // To know whether it's a success or a failure, we try to retrieve the name of the connected peer
        SocketHelper::LengthType Size =
        if (getpeername(mySocket, reinterpret_cast<sockaddr*>(&SockAddr), &Size) != -1)
        {
            // Connection accepted
            Status = Socket::Done;
        }
        else
        {
            // Connection failed
            Status = SocketHelper::GetErrorStatus;
        }
    }
    else
    {
        // Failed to connect before timeout is over
        Status = SocketHelper::GetErrorStatus;
    }
}
// Switch back to blocking mode
SetBlocking(true);
return Status;

bool SocketTCP::Listen(unsigned short Port)
{
    // Make sure our socket is valid
    if (!IsValid())
    {
        Create();
    }

    // Build the address
    sockaddr_in SockAddr;
    memset(SockAddr.sin_zero, 0, sizeof(SockAddr.sin_zero));
    SockAddr.sin_addr.s_addr = htonl(INADDR_ANY);
    SockAddr.sin_family = AF_INET;
    SockAddr.sin_port = htons(Port);

    // Bind the socket to the specified port
    if (bind(mySocket, reinterpret_cast<sockaddr*>(&SockAddr),
    {  // Not likely to happen, but...
        std::cerr << "Failed to bind socket to port 
        return false;
    }

    // Listen to the bound port
    if (listen(mySocket, 0) == -1)
    {  // Oops, socket is deaf
        std::cerr << "Failed to listen to port 
        return false;
    }
    
}
return true;

Socket::Status SocketTCP::Accept(SocketTCP& Connected) {
    // Address that will be filled with client information
    sockaddr_in ClientAddress;
    SocketHelper::LengthType Length = sizeof(ClientAddress);

    // Accept a new connection
    Connected = accept(mySocket, reinterpret_cast<sockaddr*>(&ClientAddress), &Length);

    // Check errors
    if (!Connected.IsValid()) {
        if (Address)
            *Address = IPAddress();
        return SocketHelper::GetErrorStatus();
    }

    // Fill address if requested
    if (Address)
        *Address = IPAddress(inet_ntoa(ClientAddress.sin_addr));
    return Socket::Done;
}

Socket::Status SocketTCP::Send(const char* Data, std::size_t Size) {
    // First check that socket is valid
    if (!IsValid())
        return Socket::Error;

    // Check parameters
if (Data && Size) {
    // Loop until every byte has been sent
    int Sent = 0;
    int SizeToSend = static_cast<int>(Size);
    for (int Length = 0; Length < SizeToSend;)
    {
        // Send a chunk of data
        Sent = send(mySocket, Data + Length, SizeToSend - Length, 0);
        // Check if an error occurred
        if (Sent <= 0)
            return SocketHelper::GetErrorStatus;
    }
    return Socket::Done;
}
else
{
    // Error...
    std::cerr << "Cannot send data over the network (invalid parameters)";
    return Socket::Error;
}
Socket::Status SocketTCP::Receive(char* Data, std::size_t MaxSize, std::size_t& SizeReceived) {
    // First clear the size received
    SizeReceived = 0;
    // Check that socket is valid
    if (!isValid())
        return Socket::Error;
    // Check parameters
    if (Data && MaxSize)
{ // Receive a chunk of bytes
    int Received = recv(mySocket, Data, static_cast<int>(MaxSize), 0);

    // Check the number of bytes received
    if (Received > 0) {
        SizeReceived = static_cast<std::size_t>(Received);
        return Socket::Done;
    } else if (Received == 0) {
        return Socket::Disconnected;
    } else {
        return SocketHelper::GetErrorStatus();
    }
} else { // Error...
    std::cerr << "Cannot receive data from the network (invalid parameters)";
    return Socket::Error;
}

Socket::Status SocketTCP::Send(Packet& PacketToSend) {
    // Get the data to send from the packet
    std::size_t DataSize = 0;
    const char* Data = PacketToSend.OnSend(DataSize);

    // Send the packet size
    Uint32 PacketSize = htonl(static_cast<unsigned long>(DataSize));
    Send(reinterpret_cast<const char*>(&PacketSize));
// Send the packet data
if (PacketSize > 0)
{
    return Send(Data, DataSize);
}
else
{
    return Socket::Done;
}

Socket::Status SocketTCP::Receive(Packet& PacketToReceive)
{
    // We start by getting the size of the incoming packet
    Uint32 PacketSize = 0;
    std::size_t Received = 0;
    if (myPendingPacketSize < 0)
    {
        // Loop until we've received the entire size of the packet
        while (myPendingHeaderSize < sizeof(myPendingHeader))
        {
            char* Data = reinterpret_cast<char*>(myPendingHeader);
            Socket::Status Status = Receive(Data, Received);
            myPendingHeaderSize += Received;
            
            if (Status != Socket::Done)
                return Status;
        }
        
        PacketSize = ntohl(myPendingHeader);  
        myPendingHeaderSize = 0;
    }
    else
    {

// There is a pending packet : we already know its size
PacketSize = myPendingPacketSize;

// Then loop until we receive all the packet data
char Buffer[1024];
while (myPendingPacket.size() < PacketSize)
{
    // Receive a chunk of data
    std::size_t SizeToGet = std::min(static_cast<std::size_t>(PacketSize - myPendingPacket.size()),
    Socket::Status Status = Receive(Buffer,
    if (Status != Socket::Done)
    {
        // We must save the size of the pending packet
        if (Status == Socket::NotReady)
            myPendingPacketSize = PacketSize;
        return Status;
    }
    // Append it into the packet
    if (Received > 0)
    {
        myPendingPacket.resize(myPendingPacket.size() + Received);
        char* Begin = &myPendingPacket[0] + myPendingPacket.size() - Received;
        memcpy(Begin, Buffer, Received);
    }

    // We have received all the datas : we can copy it to the user packet
    if (!myPendingPacket.empty())
        PacketToReceive.OnReceive(&myPendingPacket[0], myPendingPacket.size());
    myPendingPacket.clear();
    myPendingPacketSize = -1;
}
return Socket::Done;
```cpp
bool SocketTCP::Close()
{
    if (IsValid())
    {
        if (!SocketHelper::Close(mySocket))
        {
            std::cerr << "Failed to close socket"
            return false;
        }
        mySocket = SocketHelper::InvalidSocket();
    }
    myIsBlocking = true;
    return true;
}

bool SocketTCP::IsValid() const
{
    return mySocket != SocketHelper::InvalidSocket();
}

bool SocketTCP::operator==(const SocketTCP& Other)
{
    return mySocket == Other.mySocket;
}

bool SocketTCP::operator!=(const SocketTCP& Other)
{
    return mySocket != Other.mySocket;
}
```
bool SocketTCP::operator <(const SocketTCP & Other)
{
    return mySocket < Other.mySocket;
}

SocketTCP::SocketTCP(SocketHelper::SocketType Descriptor)
{
    Create(Descriptor);
}

void SocketTCP::Create(SocketHelper::SocketType Descriptor)
{
    // Use the given socket descriptor, or get a new one
    mySocket = Descriptor ? Descriptor : socket(PF_INET, SOCK_STREAM, 0);
    myIsBlocking = true;

    // Reset the pending packet
    myPendingHeaderSize = 0;
    myPendingPacket.clear();
    myPendingPacketSize = -1;

    // Setup default options
    if (IsValid())
    {
        // To avoid the "Address already in use" error message when trying to bind to the same port
        int Yes = 1;
        if (setsockopt(mySocket, SOL_SOCKET, SO_REUSEADDR, reinterpret_cast<char*>(&Yes),
                       0)
        {
            std::cerr << "Failed to set socket option " <<
                       "SO_REUSEADDR; binding to a same port may fail if too fast"
        }
    }

    // Disable the Nagle algorithm (ie. remove

if (setsockopt(mySocket, IPPROTO_TCP, TCP_NODELAY,
	reinterpret_cast<char*>(&Yes), 0) == -1) {
    std::cerr << "Failed to set socket option " <<
    "all your TCP packets will be buffered" << std::endl;
}

// Set blocking by default (should always be the case)
SetBlocking(true);

} // namespace sf
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any expressed or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
// you must not claim that you wrote the original software.
// If you use this software in a product, an acknowledgment
// in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
// and must not be misrepresented as being the original software.

// 3. This notice may not be removed or altered.

#ifndef SFML_SOCKETUDP_HPP
#define SFML_SOCKETUDP_HPP

// Headers
#include <SFML/Network/SocketHelper.hpp>
#include <vector>

namespace sf
{
    class Packet;
}
class IPAddress;

template <typename> class Selector;

class SFML_API SocketUDP {

public:

SocketUDP();

void SetBlocking(bool Blocking);

bool Bind(unsigned short Port);

bool Unbind();

Socket::Status Send(const char* Data, std::size_t Size);

Socket::Status Receive(char* Data, std::size_t MaxSize, std::size_t& SizeReceived);

Socket::Status Send(Packet& PacketToSend, const);

Socket::Status Receive(Packet& PacketToReceive, const);

bool Close();

bool IsValid() const;

unsigned short GetPort() const;

bool operator==(const SocketUDP& Other) const;

bool operator!=(const SocketUDP& Other) const;

bool operator<(const SocketUDP& Other) const;

private:

private;
friend class Selector<SocketUDP>;

SocketUDP(SocketHelper::SocketType Descriptor);

void Create(SocketHelper::SocketType Descriptor);

// Member data
SocketHelper::SocketType mySocket;
unsigned short myPort;
Uint32 myPendingHeader;
Uint32 myPendingHeaderSize;
std::vector<char> myPendingPacket;
Int32 myPendingPacketSize;
bool myIsBlocking;

};

} // namespace sf

#endif // SFML_SOCKETUDP_HPP
sf::SocketUDP Member List

This is the complete list of members for sf::SocketUDP, including all inherited members.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bind( unsigned short Port )</td>
<td>sf::SocketUDP</td>
</tr>
<tr>
<td>Close()</td>
<td>sf::SocketUDP</td>
</tr>
<tr>
<td>GetPort() const</td>
<td>sf::SocketUDP</td>
</tr>
<tr>
<td>IsValid() const</td>
<td>sf::SocketUDP</td>
</tr>
<tr>
<td>operator!=(const SocketUDP &amp;Other) const</td>
<td>sf::SocketUDP</td>
</tr>
<tr>
<td>operator&lt;(const SocketUDP &amp;Other) const</td>
<td>sf::SocketUDP</td>
</tr>
<tr>
<td>operator==(const SocketUDP &amp;Other) const</td>
<td>sf::SocketUDP</td>
</tr>
<tr>
<td>Receive(char *Data, std::size_t MaxSize, std::size_t &amp;SizeReceived, IPAddress &amp;Address, unsigned short &amp;Port)</td>
<td>sf::SocketUDP</td>
</tr>
<tr>
<td>Receive(Packet &amp;PacketToReceive, IPAddress &amp;Address, unsigned short &amp;Port)</td>
<td>sf::SocketUDP</td>
</tr>
<tr>
<td>Selector&lt;SocketUDP&gt;</td>
<td>(defined in sf::SocketUDP) sf::SocketUDP [friend]</td>
</tr>
<tr>
<td>Send(const char *Data, std::size_t Size, const IPAddress &amp;Address, unsigned short Port)</td>
<td>sf::SocketUDP</td>
</tr>
<tr>
<td>Send(Packet &amp;PacketToSend, const IPAddress &amp;Address, unsigned short Port)</td>
<td>sf::SocketUDP</td>
</tr>
<tr>
<td>SetBlocking(bool Blocking)</td>
<td>sf::SocketUDP</td>
</tr>
<tr>
<td>SocketUDP()</td>
<td>sf::SocketUDP</td>
</tr>
<tr>
<td>Unbind()</td>
<td>sf::SocketUDP</td>
</tr>
</tbody>
</table>

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any accompanying warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.

// 3. This notice may not be removed or altered.

// Headers
#include <SFML/Network/SocketUDP.hpp>
#include <SFML/Network/IPAddress.hpp>
#include <SFML/Network/Packet.hpp>
#include <algorithm>
#include <iostream>
#include <string.h>

namespace sf {

    // Your code here

}
SocketUDP::SocketUDP() {
    Create();
}

void SocketUDP::SetBlocking(bool Blocking) {
    // Make sure our socket is valid
    if (!IsValid())
        Create();

    SocketHelper::SetBlocking(mySocket, Blocking);
    myIsBlocking = Blocking;
}

bool SocketUDP::Bind(unsigned short Port) {
    // Check if the socket is already bound to the specified port
    if (myPort != Port)
        { // If the socket was previously bound to another port,
            Unbind();
        }

    if (Port != 0)
        { // Build an address with the specified port
            sockaddr_in Addr;
            Addr.sin_family = AF_INET;
            Addr.sin_port = htons(Port);
            Addr.sin_addr.s_addr = INADDR_ANY;
            memset(Addr.sin_zero, 0, sizeof(Addr.sin_zero));

            // Bind the socket to the port
            if (bind(mySocket, reinterpret_cast<sockaddr*>(&Addr),
                      { }
std::cerr << "Failed to bind the socket to port" << myPort = 0;
return false;

// Save the new port
myPort = Port;
return true;

bool SocketUDP::Unbind()
{
// To unbind the socket, we just recreate it
if (myPort != 0)
{
Close();
Create();
myPort = 0;
}
return true;

Socket::Status SocketUDP::Send(const char* Data,
{ // Make sure the socket is valid
if (!IsValid())
Create();

// Check parameters
if (Data && Size)
{
// Build the target address
sockaddr_in Target;
Target.sin_family = AF_INET;
Target.sin_port = htons(Port);
Target.sin_addr.s_addr = inet_addr(Address);
memeSet(Target.sin_zero, 0, sizeof(Target.sin_zero));

// Loop until every byte has been sent
int Sent = 0;
int SizeToSend = static_cast<int>(Size);
for (int Length = 0; Length < SizeToSend;
    Length += Sent) {
    // Send a chunk of data
    Sent = sendto(mySocket, Data + Length,
                  SizeToSend - Length, 0,
                  reinterpret_cast<sockaddr*>(&Target),
                  0);
    // Check errors
    if (Sent <= 0)
        return SocketHelper::GetErrorStatus;
}
return Socket::Done;

else {
    // Error...
    std::cerr << "Cannot send data over the network (invalid parameters)"
    return Socket::Error;
}

Socket::Status SocketUDP::Receive(char* Data, std::size_t MaxSize, std::size_t& SizeReceived) {
    // First clear the size received
    SizeReceived = 0;
    // Make sure the socket is bound to a port
if (myPort == 0) {
    std::cerr << "Failed to receive data ; the UDP socket first needs to be bound to a port"
    return Socket::Error;
}

// Make sure the socket is valid
if (!IsValid())
    Create();

// Check parameters
if (Data && MaxSize)
{
    sockaddr_in Sender;
    Sender.sin_family = AF_INET;
    Sender.sin_port = 0;
    Sender.sin_addr.s_addr = INADDR_ANY;
    memset(Sender.sin_zero, 0, sizeof(Sender.sin_zero));
    SocketHelper::LengthType SenderSize = sizeof(Sender);

    // Receive a chunk of bytes
    int Received = recvfrom(mySocket, Data, static_cast<int>(MaxSize), 0, reinterpret_cast<sockaddr*>(&Sender), &SenderSize);

    // Check the number of bytes received
    if (Received > 0)
    {
        Address = IPAddress(inet_ntoa(Sender.sin_addr));
        Port = ntohs(Sender.sin_port);
        SizeReceived = static_cast<std::size_t>(Received);
        return Socket::Done;
    }
    else
    {
        Address = IPAddress();
        Port = 0;
        return Received == 0 ? Socket::Disconnected : Socket::Error;
    }
}
else
{
    // Error...
    std::cerr << "Cannot receive data from the network (invalid parameters)"
    return Socket::Error;
}

Socket::Status SocketUDP::Send(Packet& PacketToSend)
{
    // Get the data to send from the packet
    std::size_t DataSize = 0;
    const char* Data = PacketToSend.OnSend(DataSize);

    // Send the packet size
    Uint32 PacketSize = htonl(static_cast<unsigned long>(DataSize));
    Send(reinterpret_cast<const char*>(&PacketSize),

    // Send the packet data
    if (PacketSize > 0)
    {
        return Send(Data, DataSize, Address, Port);
    }
    else
    {
        return Socket::Done;
    }
}

Socket::Status SocketUDP::Receive(Packet& PacketToReceive)
{
    // We start by getting the size of the incoming packet
    Uint32 PacketSize = 0;

std::size_t Received = 0;

if (myPendingPacketSize < 0)
{
    // Loop until we've received the entire packet
    // (even a 4 bytes variable may be received in more than one call)
    while (myPendingHeaderSize < sizeof(myPendingHeader))
    {
        char* Data = reinterpret_cast<char*>(Socket::Status Status = Receive(Data);
        myPendingHeaderSize += Received;

        if (Status != Socket::Done)
            return Status;
    }

    PacketSize = ntohl(myPendingHeader);
    myPendingHeaderSize = 0;
}
else
{
    // There is a pending packet : we already know its size
    PacketSize = myPendingPacketSize;
}

// Use another address instance for receiving chunks of data coming from a different sender
IPAddress Sender;
unsigned short SenderPort;

// Then loop until we receive all the packet
char Buffer[1024];
while (myPendingPacket.size() < PacketSize)
{
    // Receive a chunk of data
    std::size_t SizeToGet = std::min(static_cast
    Socket::Status Status = Receive(Buffer,
    if (Status != Socket::Done)
{  // We must save the size of the pending packet
    if (Status == Socket::NotReady)
        myPendingPacketSize = PacketSize;
    return Status;
}

// Append it into the packet
if ((Sender == Address) && (SenderPort == Port) && (Received > 0))
{
    myPendingPacket.resize(myPendingPacket.size() + Received);
    char* Begin = &myPendingPacket[0] + myPendingPacket.size() - Received;
    memcpy(Begin, Buffer, Received);
}

// We have received all the datas : we can copy it to the user packet, and clear our internal packet
PacketToReceive.Clear();
if (!myPendingPacket.empty())
    PacketToReceive.OnReceive(&myPendingPacket[0], myPendingPacket.size());
myPendingPacket.clear();
myPendingPacketSize = -1;
return Socket::Done;

bool SocketUDP::Close()
{
    if (IsValid())
    {
        if (!SocketHelper::Close(mySocket))
        {
            std::cerr << "Failed to close socket
return false;
    }
}
00326   mySocket = SocketHelper::InvalidSocket();
00327   }
00328
00329   myPort = 0;
00330   myIsBlocking = true;
00331
00332   return true;
00333 }
00334
00335
00340   bool SocketUDP::IsValid() const
00341   { 
00342     return mySocket != SocketHelper::InvalidSocket;
00343   }
00344
00345
00349   unsigned short SocketUDP::GetPort() const
00350   { 
00351     return myPort;
00352   }
00353
00354
00358   bool SocketUDP::operator==(const SocketUDP& Other) {
00359     return mySocket == Other.mySocket;
00360   }
00361
00362
00367   bool SocketUDP::operator!=(const SocketUDP& Other) {
00368     return mySocket != Other.mySocket;
00369   }
00370
00371
00378   bool SocketUDP::operator<(const SocketUDP& Other) {
00379     return mySocket < Other.mySocket;
SocketUDP::SocketUDP(SocketHelper::SocketType Descriptor)
{
    Create(Descriptor);
}

void SocketUDP::Create(SocketHelper::SocketType Descriptor)
{
    // Use the given socket descriptor, or get a new one
    mySocket = Descriptor ? Descriptor : socket(PF_INET, SOCK_DGRAM, 0);
    myIsBlocking = true;

    // Clear the last port used
    myPort = 0;

    // Reset the pending packet
    myPendingHeaderSize = 0;
    myPendingPacket.clear();
    myPendingPacketSize = -1;

    // Setup default options
    if (IsValid())
    {
        // To avoid the "Address already in use" error message when trying to bind to the same port
        int Yes = 1;
        if (setsockopt(mySocket, SOL_SOCKET, SO_REUSEADDR, reinterpret_cast<char*>(&Yes),
            0
        
        // Enable broadcast by default
        if (setsockopt(mySocket, SOL_SOCKET, SO_BROADCAST, reinterpret_cast<char*>(&Yes),
            0
            
}
std::cerr << "Failed to enable broadcast on UDP socket"

// Set blocking by default (should always be the case anyway)
SetBlocking(true);

} } // namespace sf
sf::Sound Member List

This is the complete list of members for sf::Sound, including all inherited members.

<table>
<thead>
<tr>
<th>Function</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>AudioResource()</td>
<td>sf::AudioResource [protected]</td>
</tr>
<tr>
<td>AudioResource(const AudioResource &amp;)</td>
<td>sf::AudioResource [protected]</td>
</tr>
<tr>
<td>GetAttenuation() const</td>
<td>sf::Sound</td>
</tr>
<tr>
<td>GetBuffer() const</td>
<td>sf::Sound</td>
</tr>
<tr>
<td>GetLoop() const</td>
<td>sf::Sound</td>
</tr>
<tr>
<td>GetMinDistance() const</td>
<td>sf::Sound</td>
</tr>
<tr>
<td>GetPitch() const</td>
<td>sf::Sound</td>
</tr>
<tr>
<td>GetPlayingOffset() const</td>
<td>sf::Sound</td>
</tr>
<tr>
<td>GetPosition() const</td>
<td>sf::Sound</td>
</tr>
<tr>
<td>GetStatus() const</td>
<td>sf::Sound</td>
</tr>
<tr>
<td>GetVolume() const</td>
<td>sf::Sound</td>
</tr>
<tr>
<td>IsRelativeToListener() const</td>
<td>sf::Sound</td>
</tr>
<tr>
<td>operator=(const Sound &amp;Other)</td>
<td>sf::Sound</td>
</tr>
<tr>
<td>Pause()</td>
<td>sf::Sound</td>
</tr>
<tr>
<td>Paused enum value</td>
<td>sf::Sound</td>
</tr>
<tr>
<td>Play()</td>
<td>sf::Sound</td>
</tr>
<tr>
<td>Playing enum value</td>
<td>sf::Sound</td>
</tr>
<tr>
<td>ResetBuffer()</td>
<td>sf::Sound</td>
</tr>
<tr>
<td>SetAttenuation(float Attenuation)</td>
<td>sf::Sound</td>
</tr>
<tr>
<td>SetBuffer(const SoundBuffer &amp;Buffer)</td>
<td>sf::Sound</td>
</tr>
<tr>
<td>SetLoop(bool Loop)</td>
<td>sf::Sound</td>
</tr>
<tr>
<td>SetMinDistance(float MinDistance)</td>
<td>sf::Sound</td>
</tr>
<tr>
<td>SetPitch(float Pitch)</td>
<td>sf::Sound</td>
</tr>
<tr>
<td>SetPlayingOffset(float TimeOffset)</td>
<td>sf::Sound</td>
</tr>
<tr>
<td>SetPosition(float X, float Y, float Z)</td>
<td>sf::Sound</td>
</tr>
<tr>
<td>SetPosition(const Vector3f &amp;Position)</td>
<td>sf::Sound</td>
</tr>
<tr>
<td>SetRelativeToListener(bool Relative)</td>
<td>sf::Sound</td>
</tr>
<tr>
<td>SetVolume(float Volume)</td>
<td>sf::Sound</td>
</tr>
<tr>
<td>Sound()</td>
<td>sf::Sound</td>
</tr>
<tr>
<td>Sound(const SoundBuffer &amp;Buffer, bool)</td>
<td>sf::Sound</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td>---------------</td>
</tr>
<tr>
<td>Loop=false, float Pitch=1.f, float Volume=100.f, const Vector3f &amp;Position=Vector3f(0, 0, 0)</td>
<td>sf::Sound</td>
</tr>
<tr>
<td><strong>Sound</strong> (const Sound &amp;Copy)</td>
<td>sf::Sound</td>
</tr>
<tr>
<td><strong>SoundStream</strong> (defined in <strong>sf::Sound</strong>)</td>
<td>sf::Sound</td>
</tr>
<tr>
<td><strong>Status</strong> enum name</td>
<td>sf::Sound</td>
</tr>
<tr>
<td><strong>Stop</strong> ()</td>
<td>sf::Sound</td>
</tr>
<tr>
<td><strong>Stopped</strong> enum value</td>
<td>sf::Sound</td>
</tr>
<tr>
<td>~<strong>AudioResource</strong> ()</td>
<td>sf::AudioResource</td>
</tr>
<tr>
<td>~<strong>Sound</strong> ()</td>
<td>sf::Sound</td>
</tr>
</tbody>
</table>
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)
// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.
// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:
// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is not required.
// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.
// 3. This notice may not be removed or altered from any source distribution.

namespace sf
{
    Sound::Sound()
    {
        ALCheck(alGenSources(1, &mySource));
    }
}
ALCheck(alSourcei(mySource, AL_BUFFER, 0));
}

Sound::Sound(const SoundBuffer& Buffer, bool Loop)
myBuffer(NULL)
{
    ALCheck(alGenSources(1, &mySource));
    SetBuffer(Buffer);
    SetLoop(Loop);
    SetPitch(Pitch);
    SetVolume(Volume);
    setPosition(Position);
}

Sound::Sound(const Sound& Copy) :
AudioResource(Copy),
myBuffer(NULL)
{
    ALCheck(alGenSources(1, &mySource));
    if (Copy.myBuffer)
        SetBuffer(*Copy.myBuffer);
    SetLoop(Copy.GetLoop);
    SetPitch(Copy.GetPitch());
    SetVolume(Copy.GetVolume());
    SetPosition(Copy.GetPosition());
    SetRelativeToListener(Copy.IsRelativeToListener);
    SetMinDistance(Copy.GetMinDistance());
    SetAttenuation(Copy.GetAttenuation());
}

Sound::~Sound()
{
if (mySource) {
    if (myBuffer) {
        Stop();
        ALCheck(alSourcei(mySource, AL_BUFFER, myBuffer->DetachSound(this));
    }
    ALCheck(alDeleteSources(1, &mySource));
}

void Sound::Play() {
    ALCheck(alSourcePlay(mySource));
}

void Sound::Pause() {
    ALCheck(alSourcePause(mySource));
}

void Sound::Stop() {
    ALCheck(alSourceStop(mySource));
}

void Sound::SetBuffer(const SoundBuffer& Buffer) {
    // First detach from the previous buffer
    if (myBuffer) {
        Stop();
    }
}
myBuffer->DetachSound(this);

// Assign and use the new buffer
myBuffer = &Buffer;
myBuffer->AttachSound(this);
ALCheck(alSourcei(mySource, AL_BUFFER, myBuffer->myBuffer));

void Sound::SetLoop(bool Loop)
{
    ALCheck(alSourcei(mySource, AL_LOOPING, Loop));
}

void Sound::SetPitch(float Pitch)
{
    ALCheck(alSourcef(mySource, AL_PITCH, Pitch));
}

void Sound::SetVolume(float Volume)
{
    ALCheck(alSourcef(mySource, AL_GAIN, Volume));
}

void Sound::SetPosition(float X, float Y, float Z)
{
    ALCheck(alSource3f(mySource, AL_POSITION, X, Y, Z));
}

void Sound::SetPosition(const Vector3f& Position)
{
    SetPosition(Position.x, Position.y, Position.z);
}
void Sound::SetRelativeToListener(bool Relative) {
    ALCheck(alSourcei(mySource, AL_SOURCE_RELATIVE, Relative));
}

void Sound::SetMinDistance(float MinDistance) {
    ALCheck(alSourcef(mySource, AL_REFERENCE_DISTANCE, MinDistance));
}

void Sound::SetAttenuation(float Attenuation) {
    ALCheck(alSourcef(mySource, AL_ROLLOFF_FACTOR, Attenuation));
}

void Sound::SetPlayingOffset(float TimeOffset) {
    ALCheck(alSourcef(mySource, AL_SEC_OFFSET, TimeOffset));
}

const SoundBuffer* Sound::GetBuffer() const {
    return myBuffer;
}

bool Sound::GetLoop() const {
    ALint Loop;
    ALCheck(alGetSourcei(mySource, AL_LOOPING, &Loop));
    return Loop != 0;
}
float Sound::GetPitch() const
{
    ALfloat Pitch;
    ALCheck(alGetSourcef(mySource, AL_PITCH, &Pitch));
    return Pitch;
}

float Sound::GetVolume() const
{
    ALfloat Gain;
    ALCheck(alGetSourcef(mySource, AL_GAIN, &Gain));
    return Gain * 100.f;
}

Vector3f Sound::GetPosition() const
{
    Vector3f Position;
    ALCheck(alGetSource3f(mySource, AL_POSITION, &Position));
    return Position;
}

bool Sound::IsRelativeToListener() const
{
    ALint Relative;
    ALCheck(alGetSourcei(mySource, AL_SOURCE_RELATIVE, &Relative));
    return Relative != 0;
float Sound::GetMinDistance() const
{
    ALfloat MinDistance;
    ALCheck(alGetSourcef(mySource, AL_REFERENCE_DISTANCE, &MinDistance));
    return MinDistance;
}

float Sound::GetAttenuation() const
{
    ALfloat Attenuation;
    ALCheck(alGetSourcef(mySource, AL_ROLLOFF_FACTOR, &Attenuation));
    return Attenuation;
}

float Sound::GetPlayingOffset() const
{
    ALfloat Seconds = 0.f;
    ALCheck(alGetSourcef(mySource, AL_SEC_OFFSET, &Seconds));
    return Seconds;
}

Sound::Status Sound::GetStatus() const
{
    ALint State;
    ALCheck(alGetSourcei(mySource, AL_SOURCE_STATE, &State));
    switch (State)
    {

case AL_INITIAL :
    return Stopped;
    
    case AL_STOPPED :
        return Stopped;
    
    case AL_PAUSED :
        return Paused;
    
    case AL_PLAYING :
        return Playing;
    
    return Stopped;
}

Sound& Sound::operator =(const Sound& Other)
{
    // Here we don't use the copy-and-swap idiom,
    // because it would mess up the list of sound instances contained in
    // Detach the sound instance from the previous
    if (myBuffer)
    {
        Stop();
        myBuffer->DetachSound(this);
        myBuffer = NULL;
    }

    // Copy the sound attributes
    if (Other.myBuffer)
        SetBuffer(*Other.myBuffer);
    SetLoop(Other.GetLoop());
    SetPitch(Other.GetPitch());
    SetVolume(Other.GetVolume());
    SetPosition(Other.GetPosition());
    SetRelativeToListener(Other.IsRelativeToListener);
    SetMinDistance(Other.GetMinDistance());
    SetAttenuation(Other.GetAttenuation());
    return *this;
void Sound::ResetBuffer()
{
    // First stop the sound in case it is playing
    Stop();

    // Detach the buffer
    ALCheck(alSourcei(mySource, AL_BUFFER, 0));
    myBuffer = NULL;
}

// namespace sf
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gomila@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
//     you must not claim that you wrote the original software.
//     If you use this software in a product, an acknowledgment
//     in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
//     and must not be misrepresented as being the original software.

// 3. This notice may not be removed or altered.

#ifndef SFML_SOUNDBUFFER_HPP
#define SFML_SOUNDBUFFER_HPP

// Headers
#include <SFML/System/Resource.hpp>
#include <SFML/Audio/AudioResource.hpp>
#include <string>
#include <vector>
#include <set>

#endif
namespace sf
{
class Sound;

class SFML_API SoundBuffer : public AudioResource
{
published:
  SoundBuffer();
  SoundBuffer(const SoundBuffer& Copy);
  ~SoundBuffer();

  bool LoadFromFile(const std::string& Filename);
  bool LoadFromMemory(const char* Data, std::size_t SizeInBytes);
  bool LoadFromSamples(const Int16* Samples, std::size_t SamplesCount);
  bool SaveToFile(const std::string& Filename);

  Int16* GetSamples() const;
  size_t GetSamplesCount() const;
  unsigned int GetSampleRate() const;
  unsigned int GetChannelsCount() const;
  float GetDuration() const;

  SoundBuffer& operator =(const SoundBuffer& Other);

private:
  friend class Sound;
bool Update(unsigned int ChannelsCount, unsigned int ChannelsCount);

void AttachSound(Sound* Instance) const;

void DetachSound(Sound* Instance) const;

// Types
typedef std::set<Sound*> SoundList;

// Member data
unsigned int myBuffer;
std::vector<Int16> mySamples;
float myDuration;
mutable SoundList mySounds;

};

} // namespace sf

#endif // SFML_SOUNDBUFFER_HPP
sf::SoundBuffer Member List

This is the complete list of members for sf::SoundBuffer, including all inherited members.

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AudioResource()</td>
<td>sf::AudioResource [protected]</td>
</tr>
<tr>
<td>AudioResource(const AudioResource &amp;)</td>
<td>sf::AudioResource [protected]</td>
</tr>
<tr>
<td>GetChannelsCount() const</td>
<td>sf::SoundBuffer</td>
</tr>
<tr>
<td>GetDuration() const</td>
<td>sf::SoundBuffer</td>
</tr>
<tr>
<td>GetSampleRate() const</td>
<td>sf::SoundBuffer</td>
</tr>
<tr>
<td>GetSamples() const</td>
<td>sf::SoundBuffer</td>
</tr>
<tr>
<td>GetSamplesCount() const</td>
<td>sf::SoundBuffer</td>
</tr>
<tr>
<td>LoadFromFile(const std::string &amp;Filename)</td>
<td>sf::SoundBuffer</td>
</tr>
<tr>
<td>LoadFromMemory(const char *Data, std::size_t SizeInBytes)</td>
<td>sf::SoundBuffer</td>
</tr>
<tr>
<td>LoadFromSamples(const Int16 *Samples, std::size_t SamplesCount, unsigned int ChannelsCount, unsigned int SampleRate)</td>
<td>sf::SoundBuffer</td>
</tr>
<tr>
<td>operator=(const SoundBuffer &amp;Other)</td>
<td>sf::SoundBuffer</td>
</tr>
<tr>
<td>Resource&lt; SoundBuffer &gt;::operator=(const Resource&lt; SoundBuffer &gt; &amp;Other)</td>
<td>sf::Resource&lt; SoundBuffer &gt; [protected]</td>
</tr>
<tr>
<td>Resource()</td>
<td>sf::Resource&lt; SoundBuffer &gt; [protected]</td>
</tr>
<tr>
<td>Resource(const Resource&lt; SoundBuffer &gt; &amp;Copy)</td>
<td>sf::Resource&lt; SoundBuffer &gt; [protected]</td>
</tr>
<tr>
<td>SaveToFile(const std::string &amp;Filename) const</td>
<td>sf::SoundBuffer</td>
</tr>
<tr>
<td>Sound (defined in sf::SoundBuffer)</td>
<td>sf::SoundBuffer [friend]</td>
</tr>
<tr>
<td>SoundBuffer()</td>
<td>sf::SoundBuffer</td>
</tr>
<tr>
<td>SoundBuffer(const SoundBuffer &amp;Copy)</td>
<td>sf::SoundBuffer</td>
</tr>
<tr>
<td>~AudioResource()</td>
<td>sf::AudioResource [protected, virtual]</td>
</tr>
<tr>
<td>~Resource()</td>
<td>sf::Resource&lt; SoundBuffer &gt; [protected]</td>
</tr>
<tr>
<td>~SoundBuffer()</td>
<td>sf::SoundBuffer</td>
</tr>
</tbody>
</table>
Documentation generated by doxygen 1.5.2 ::
Main Page          Namespaces          Classes          Files

File List
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
// you must not claim that you wrote the original software.
// If you use this software in a product, an acknowledgment
// in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
// and must not be misrepresented as being the original software.

// 3. This notice may not be removed or altered.

// Headers
#include <SFML/Audio/SoundBuffer.hpp>
#include <SFML/Audio/SoundFile.hpp>
#include <SFML/Audio/Sound.hpp>
#include <SFML/Audio/OpenAL.hpp>
#include <iostream>
#include <memory>

namespace sf
SoundBuffer::SoundBuffer() :
  myBuffer (0),
  myDuration(0.f)
{
  // Create the buffer
  ALCheck(alGenBuffers(1, &myBuffer));
}

SoundBuffer::SoundBuffer(const SoundBuffer& Copy) :
  AudioResource (Copy),
  Resource<SoundBuffer>(Copy),
  myBuffer (0),
  mySamples (Copy.mySamples),
  myDuration (Copy.myDuration),
  mySounds () // don't copy the attached sounds
{
  // Create the buffer
  ALCheck(alGenBuffers(1, &myBuffer));
  // Update the internal buffer with the new samples
  Update(Copy.GetChannelsCount(), Copy.GetSampleRate());
}

~SoundBuffer()
{
  // First detach the buffer from the sounds that use it
  for (SoundList::const_iterator it = mySounds.begin(); it != mySounds.end(); ++it)
    (*it)->ResetBuffer();
  // Destroy the buffer
  if (myBuffer)
    ALCheck(alDeleteBuffers(1, &myBuffer));
}
bool SoundBuffer::LoadFromFile(const std::string& Filename) {
    // Create the sound file
    std::auto_ptr<priv::SoundFile> File(priv::SoundFile::CreateRead(Filename));

    // Open the sound file
    if (File.get()) {
        // Get the sound parameters
        std::size_t NbSamples = File->GetSamplesCount();
        unsigned int ChannelsCount = File->GetChannelsCount();
        unsigned int SampleRate = File->GetSampleRate();

        // Read the samples from the opened file
        mySamples.resize(NbSamples);
        if (File->Read(&mySamples[0], NbSamples) == NbSamples) {
            // Update the internal buffer with the new samples
            return Update(ChannelsCount, SampleRate);
        } else {
            // Error...
            std::cerr << "Failed to read audio data from file " << Filename << std::endl;
            return false;
        }
    } else {
        // Error...
        std::cerr << "Failed to load sound buffer from file " << Filename << std::endl;
        return false;
    }
}
bool SoundBuffer::LoadFromMemory(const char* Data) {
    // Create the sound file
    std::auto_ptr<priv::SoundFile> File(priv::SoundFile::CreateRead(Data, SizeInBytes));

    // Open the sound file
    if (File.get()) {
        // Get the sound parameters
        std::size_t NbSamples = File->GetSamplesCount();
        unsigned int ChannelsCount = File->GetChannelsCount();
        unsigned int SampleRate = File->GetSampleRate();

        // Read the samples from the opened file
        mySamples.resize(NbSamples);
        if (File->Read(&mySamples[0], NbSamples)) {
            // Update the internal buffer with the new samples
            return Update(ChannelsCount, SampleRate);
        } else {
            // Error...
            std::cerr << "Failed to read audio data from file in memory";
            return false;
        }
    } else {
        // Error...
        std::cerr << "Failed to load sound buffer from file in memory";
        return false;
    }
}
bool SoundBuffer::LoadFromSamples(const Int16* Samples, std::size_t SamplesCount, ChannelsCount, SampleRate)
{
    if (Samples && SamplesCount && ChannelsCount && SampleRate)
    {
        // Copy the new audio samples
        mySamples.assign(Samples, Samples + SamplesCount);

        // Update the internal buffer with the new samples
        return Update(ChannelsCount, SampleRate);
    }
    else
    {
        // Error...
        std::cerr << "Failed to load sound buffer from memory (Samples: " << Samples << "Samples count: " << SamplesCount
                 << "Channels count: " << ChannelsCount << "Sample rate: " << SampleRate << std::endl;

        return false;
    }
}

bool SoundBuffer::SaveToFile(const std::string& Filename)
{
    // Create the sound file in write mode
    std::auto_ptr<priv::SoundFile> File(priv::SoundFile::CreateWrite(Filename,
    if (File.get())
    {
        // Write the samples to the opened file
        File->Write(&mySamples[0], mySamples.size());
    }
    return false;
}
return true;

else
{
    // Error...
    std::cerr << "Failed to save sound buffer to file "
    return false;
}

const Int16* SoundBuffer::GetSamples() const
{
    return mySamples.empty() ? NULL : &mySamples;
}

std::size_t SoundBuffer::GetSamplesCount() const
{
    return mySamples.size();
}

unsigned int SoundBuffer::GetSampleRate() const
{
    ALint SampleRate;
    ALCheck(alGetBufferi(myBuffer, AL_FREQUENCY, &SampleRate));
    return SampleRate;
}

unsigned int SoundBuffer::GetChannelsCount() const
{
    ALint ChannelsCount;
    ALCheck(alGetBufferi(myBuffer, AL_CHANNELS, &ChannelsCount));
return ChannelsCount;
}

float SoundBuffer::GetDuration() const
{
    return myDuration;
}

SoundBuffer& SoundBuffer::operator =(const SoundBuffer& Other)
{
    SoundBuffer Temp(Other);
    std::swap(mySamples, Temp.mySamples);
    std::swap(myBuffer, Temp.myBuffer);
    std::swap(myDuration, Temp.myDuration);
    std::swap(mySounds, Temp.mySounds); // swap sounds too, so that they are detached when Temp is destroyed
    return *this;
}

bool SoundBuffer::Update(unsigned int ChannelsCount)
{
    // Check parameters
    if (!SampleRate || !ChannelsCount || mySamples.empty())
        return false;
    // Find the good format according to the number of channels
    ALenum Format = priv::AudioDevice::GetInstance().GetFormatFromChannelsCount(ChannelsCount);
    // Check if the format is valid
    if (Format == 0)
    {
        std::cerr << "Unsupported number of channels: " << ChannelsCount << std::endl;
        return false;
    }
    return true;
}
return false;

// Fill the buffer
ALsizei Size = static_cast<ALsizei>(mySamples.size())
ALCheck(alBufferData(myBuffer, Format, &mySamples[0], Size, SampleRate));

// Compute the duration
myDuration = static_cast<float>(mySamples.size()) / SampleRate / ChannelsCount;
return true;

void SoundBuffer::AttachSound(Sound* Instance) const {
    mySounds.insert(Instance);
}

void SoundBuffer::DetachSound(Sound* Instance) const {
    mySounds.erase(Instance);
}

} // namespace sf
SoundBufferRecorder.hpp

// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gomila@gmail.com)
//
// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.
//
// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:
// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is not required.
// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.
// 3. This notice may not be removed or altered from any source distribution.

#ifndef SFML_SOUNDBUFFERRECORDER_HPP
#define SFML_SOUNDBUFFERRECORDER_HPP

namespace sf
{

#include <SFML/Audio/SoundBuffer.hpp>
#include <SFML/Audio/SoundRecorder.hpp>
#include <vector>

namespace sf
{


class SFML_API SoundBufferRecorder : public SoundRecorder
{
public :

const SoundBuffer& GetBuffer() const;

private :

virtual bool OnStart();

virtual bool OnProcessSamples(const Int16* Samples, std::size_t SamplesCount);

virtual void OnStop();

// Member data
std::vector<Int16> mySamples;
SoundBuffer myBuffer;
};

// namespace sf

#endif // SFML_SOUNDBUFFERRECORDER_HPP
## sf::SoundBufferRecorder Member List

This is the complete list of members for **sf::SoundBufferRecorder**, including all inherited members.

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanCapture()</td>
<td>sf::SoundRecorder</td>
<td>[static]</td>
</tr>
<tr>
<td>GetBuffer() const</td>
<td>sf::SoundBufferRecorder</td>
<td></td>
</tr>
<tr>
<td>GetSampleRate() const</td>
<td>sf::SoundRecorder</td>
<td></td>
</tr>
<tr>
<td>SoundRecorder()</td>
<td>sf::SoundRecorder</td>
<td>[protected]</td>
</tr>
<tr>
<td>Start(unsigned int SampleRate=44100)</td>
<td>sf::SoundRecorder</td>
<td></td>
</tr>
<tr>
<td>Stop()</td>
<td>sf::SoundRecorder</td>
<td></td>
</tr>
<tr>
<td>~SoundRecorder()</td>
<td>sf::SoundRecorder</td>
<td>[virtual]</td>
</tr>
</tbody>
</table>

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::

Documentation generated by doxygen 1.5.2 ::
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)
//
// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.
//
// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:
//
// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is not required.
// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.
// 3. This notice may not be removed or altered from any source distribution.

#include <SFML/Audio/SoundRecorder.hpp>
#include <SFML/Audio/AudioDevice.hpp>
#include <SFML/Audio/OpenAL.hpp>
#include <SFML/System/Sleep.hpp>
#include <iostream>

// Private data
namespace
{
    // Headers
    #include <SFML/Audio/SoundRecorder.hpp>
    #include <SFML/Audio/AudioDevice.hpp>
    #include <SFML/Audio/OpenAL.hpp>
    #include <SFML/System/Sleep.hpp>
    #include <iostream>

    // Private data
    namespace
    {
    
    }}
ALCdevice* CaptureDevice = NULL;
}
namespace sf
{
SoundRecorder::SoundRecorder() :
mySampleRate (0),
myIsCapturing(false)
{

SoundRecorder::~SoundRecorder()
{
  // Nothing to do
}

void SoundRecorder::Start(unsigned int SampleRate)
{
  // Check if the device can do audio capture
  if (!CanCapture())
  {
    std::cerr << "Failed to start capture : 
    return;
  }
  // Check that another capture is not already
  if (CaptureDevice)
  {
    std::cerr << "Trying to start audio capture
    return;
  }
  // Open the capture device for capturing 16
  CaptureDevice = alcCaptureOpenDevice(NULL, &

if (!CaptureDevice)
{
    std::cerr << "Failed to open the audio device.
    return;
}

// Clear the sample array
mySamples.clear();

// Store the sample rate
mySampleRate = SampleRate;

// Notify derived class
if (OnStart())
{
    // Start the capture
    alcCaptureStart(CaptureDevice);
    myIsCapturing = true;
    Launch();
}

void SoundRecorder::Stop()
{
    // Stop the capturing thread
    myIsCapturing = false;
    Wait();
}

unsigned int SoundRecorder::GetSampleRate() const
{
    return mySampleRate;
}
bool SoundRecorder::CanCapture()
{
    ALCdevice* Device = priv::AudioDevice::GetInstance().GetDevice();
    return (alcIsExtensionPresent(Device, "ALC_"
        (alcIsExtensionPresent(Device, "ALC_"
    }
}

bool SoundRecorder::OnStart()
{
    // Nothing to do
    return true;
}

void SoundRecorder::OnStop()
{
    // Nothing to do
}

void SoundRecorder::Run()
{
    while (myIsCapturing)
    {
        // Process available samples
        ProcessCapturedSamples();
        // Don't bother the CPU while waiting for
        Sleep(0.1f);
    }

    // Capture is finished : clean up everything
    CleanUp();
void SoundRecorder::ProcessCapturedSamples()
{
    // Get the number of samples available
    ALCint SamplesAvailable;
    alcGetIntegerv(CaptureDevice, ALC_CAPTURE_SAMPLES, 1, &SamplesAvailable);
    if (SamplesAvailable > 0)
    {
        // Get the recorded samples
        mySamples.resize(SamplesAvailable);
        alcCaptureSamples(CaptureDevice, &mySamples[0], SamplesAvailable);

        // Forward them to the derived class
        if (!OnProcessSamples(&mySamples[0], mySamples.size()))
        {
            // The user wants to stop the capture
            myIsCapturing = false;
        }
    }
}

void SoundRecorder::CleanUp()
{
    // Stop the capture
    alcCaptureStop(CaptureDevice);

    // Get the samples left in the buffer
    ProcessCapturedSamples();

    // Close the device
00223    alcCaptureCloseDevice(CaptureDevice);
00224    CaptureDevice = NULL;
00225 }
00226
00227 } // namespace sf
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.

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

// Headers
#include <SFML/Audio/SoundBufferRecorder.hpp>
#include <algorithm>
#include <iterator>

namespace sf {

bool SoundBufferRecorder::OnStart()
{
    mySamples.clear();
}
return true;
}

bool SoundBufferRecorder::OnProcessSamples(const
{
    std::copy(Samples, Samples + SamplesCount, s
    return true;
}

void SoundBufferRecorder::OnStop()
{
    if (!mySamples.empty())
        myBuffer.LoadFromSamples(&mySamples[0],
    }
}

const SoundBuffer& SoundBufferRecorder::GetBuffer
{
    return myBuffer;
}

} // namespace sf

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
soundrecorder.hpp

// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.

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

#ifndef SFML_SOUNDRECORDER_HPP
#define SFML_SOUNDRECORDER_HPP

// Headers
#include <SFML/System/Thread.hpp>
#include <vector>

namespace sf
{

class SFML_API SoundRecorder : private Thread

}
```cpp
{ 
    public:
    virtual ~SoundRecorder();
    void Start(unsigned int SampleRate = 44100);
    void Stop();
    unsigned int GetSampleRate() const;
    static bool CanCapture();

    protected:
    SoundRecorder();

    private:
    virtual bool OnStart();
    virtual bool OnProcessSamples(const Int16* Samples, std::size_t SamplesCount) = 0;
    virtual void OnStop();
    virtual void Run();
    void ProcessCapturedSamples();
    void CleanUp();

    // Member data
    std::vector<Int16> mySamples;
    unsigned int mySampleRate;
    bool myIsCapturing;
};
```
} // namespace sf
#endif // SFML_SOUNDERORDER_HPP

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
<table>
<thead>
<tr>
<th>Main Page</th>
<th>Namespaces</th>
<th>Classes</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class List</td>
<td>Class Index</td>
<td>Class Hierarchy</td>
<td>Class Members</td>
</tr>
</tbody>
</table>
sf::SoundRecorder Member List

This is the complete list of members for **sf::SoundRecorder**, including all inherited members.

<table>
<thead>
<tr>
<th>Method</th>
<th>sf::SoundRecorder</th>
<th>sf::Thread</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanCapture()</td>
<td>[static]</td>
<td></td>
</tr>
<tr>
<td>FuncType typedef (defined in sf::Thread)</td>
<td>sf::Thread [private]</td>
<td></td>
</tr>
<tr>
<td>GetSampleRate() const</td>
<td>sf::SoundRecorder</td>
<td></td>
</tr>
<tr>
<td>Launch()</td>
<td>sf::Thread [private]</td>
<td></td>
</tr>
<tr>
<td>SoundRecorder()</td>
<td>sf::SoundRecorder [protected]</td>
<td></td>
</tr>
<tr>
<td>Start(unsigned int SampleRate=44100)</td>
<td>sf::SoundRecorder</td>
<td></td>
</tr>
<tr>
<td>Stop()</td>
<td>sf::SoundRecorder</td>
<td></td>
</tr>
<tr>
<td>Terminate()</td>
<td>sf::Thread [private]</td>
<td></td>
</tr>
<tr>
<td>Thread(FuncType Function, void *UserData=NULL)</td>
<td>sf::Thread [private]</td>
<td></td>
</tr>
<tr>
<td>Thread()</td>
<td>sf::Thread [private]</td>
<td></td>
</tr>
<tr>
<td>Wait()</td>
<td>sf::Thread [private]</td>
<td></td>
</tr>
<tr>
<td>~SoundRecorder()</td>
<td>sf::SoundRecorder [virtual]</td>
<td></td>
</tr>
<tr>
<td>~Thread()</td>
<td>sf::Thread [private, virtual]</td>
<td></td>
</tr>
<tr>
<td>Main Page</td>
<td>Namespaces</td>
<td>Classes</td>
</tr>
<tr>
<td>-----------</td>
<td>------------</td>
<td>---------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.

// 3. This notice may not be removed or altered.

#ifndef SFML_SOUNDSTREAM_HPP
#define SFML_SOUNDSTREAM_HPP

// Headers
#include <SFML/Audio/Sound.hpp>
#include <SFML/System/Thread.hpp>
#include <cstdlib>

namespace sf {

#define SFML_SOUNDSTREAM_HPP

# ifndef SFML_SOUNDSTREAM_HPP
#define SFML_SOUNDSTREAM_HPP

// Headers
#include <SFML/Audio/Sound.hpp>
#include <SFML/System/Thread.hpp>
#include <cstdlib>

namespace sf {

{
class SFML_API SoundStream : private Thread, private Thread
{
public:

using Sound::Status;
using Sound::Stopped;
using Sound::Paused;
using Sound::Playing;
using Sound::Pause;
using Sound::SetPitch;
using Sound::SetVolume;
using Sound::SetPosition;
using Sound::SetRelativeToListener;
using Sound::SetMinDistance;
using Sound::SetAttenuation;
using Sound::GetPitch;
using Sound::GetVolume;
using Sound::GetPosition;
using Sound::IsRelativeToListener;
using Sound::GetMinDistance;
using Sound::GetAttenuation;

struct Chunk
{
    const Int16* Samples;
    std::size_t NbSamples;
};

virtual ~SoundStream();

void Play();

void Stop();

unsigned int GetChannelsCount() const;

unsigned int GetSampleRate() const;
Status GetStatus() const;

float GetPlayingOffset() const;

void SetLoop(bool Loop);

bool GetLoop() const;

protected :

SoundStream();

void Initialize(unsigned int ChannelsCount,

private :

virtual void Run();

virtual bool OnStart();

virtual bool OnGetData(Chunk& Data) = 0;

bool FillAndPushBuffer(unsigned int BufferNum);

bool FillQueue();

void ClearQueue();

enum {BuffersCount = 3};

// Member data

bool myIsStreaming;

unsigned int myBuffers[BuffersCount];

unsigned int myChannelsCount;

unsigned int mySampleRate;

unsigned long myFormat;
bool myLoop;
unsigned int mySamplesProcessed;
bool myEndBuffers[BuffersCount];
}
};

// namespace sf

#endif  // SFML_SOUNDSTREAM_HPP
## sf::SoundStream Member List

This is the complete list of members for `sf::SoundStream`, including all inherited members.

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>AudioResource()</td>
<td>sf::AudioResource</td>
<td>private</td>
</tr>
<tr>
<td>AudioResource(const AudioResource&amp;)</td>
<td>sf::AudioResource</td>
<td>private</td>
</tr>
<tr>
<td>FuncType typedef (defined in sf::Thread)</td>
<td>sf::Thread</td>
<td>private</td>
</tr>
<tr>
<td>GetAttenuation() const</td>
<td>sf::Sound</td>
<td>private</td>
</tr>
<tr>
<td>GetBuffer() const</td>
<td>sf::Sound</td>
<td>private</td>
</tr>
<tr>
<td>GetChannelsCount() const</td>
<td>sf::SoundStream</td>
<td></td>
</tr>
<tr>
<td>GetLoop() const</td>
<td>sf::SoundStream</td>
<td></td>
</tr>
<tr>
<td>GetMinDistance() const</td>
<td>sf::Sound</td>
<td>private</td>
</tr>
<tr>
<td>GetPitch() const</td>
<td>sf::Sound</td>
<td>private</td>
</tr>
<tr>
<td>GetPlayingOffset() const</td>
<td>sf::SoundStream</td>
<td></td>
</tr>
<tr>
<td>GetPosition() const</td>
<td>sf::Sound</td>
<td>private</td>
</tr>
<tr>
<td>GetSampleRate() const</td>
<td>sf::SoundStream</td>
<td></td>
</tr>
<tr>
<td>GetStatus() const</td>
<td>sf::SoundStream</td>
<td></td>
</tr>
<tr>
<td>GetVolume() const</td>
<td>sf::Sound</td>
<td>private</td>
</tr>
<tr>
<td>Initialize(unsigned int ChannelsCount, unsigned int SampleRate)</td>
<td>sf::SoundStream</td>
<td>protected</td>
</tr>
<tr>
<td>IsRelativeToListener() const</td>
<td>sf::Sound</td>
<td>private</td>
</tr>
<tr>
<td>Launch()</td>
<td>sf::Thread</td>
<td>private</td>
</tr>
<tr>
<td>operator=(const Sound &amp;Other)</td>
<td>sf::Sound</td>
<td>private</td>
</tr>
<tr>
<td>Pause()</td>
<td>sf::Sound</td>
<td>private</td>
</tr>
<tr>
<td>Paused enum value</td>
<td>sf::Sound</td>
<td>private</td>
</tr>
<tr>
<td>Play()</td>
<td>sf::SoundStream</td>
<td></td>
</tr>
<tr>
<td>Playing enum value</td>
<td>sf::Sound</td>
<td>private</td>
</tr>
<tr>
<td>ResetBuffer()</td>
<td>sf::Sound</td>
<td>private</td>
</tr>
<tr>
<td>SetAttenuation(float Attenuation)</td>
<td>sf::Sound</td>
<td>private</td>
</tr>
<tr>
<td>SetBuffer(const SoundBuffer &amp;Buffer)</td>
<td>sf::Sound</td>
<td>private</td>
</tr>
<tr>
<td>SetLoop(bool Loop)</td>
<td>sf::SoundStream</td>
<td></td>
</tr>
<tr>
<td>SetMinDistance(float MinDistance)</td>
<td>sf::Sound</td>
<td>private</td>
</tr>
<tr>
<td>SetPitch(float Pitch)</td>
<td>sf::Sound</td>
<td>private</td>
</tr>
<tr>
<td>SetPlayingOffset(float TimeOffset)</td>
<td>sf::Sound</td>
<td>private</td>
</tr>
<tr>
<td>Function</td>
<td>Type</td>
<td>Access</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>---------------</td>
<td>---------</td>
</tr>
<tr>
<td>SetPosition(float X, float Y, float Z)</td>
<td>sf::Sound</td>
<td>[private]</td>
</tr>
<tr>
<td>SetPosition(const Vector3f &amp;Position)</td>
<td>sf::Sound</td>
<td>[private]</td>
</tr>
<tr>
<td>SetRelativeToListener(bool Relative)</td>
<td>sf::Sound</td>
<td>[private]</td>
</tr>
<tr>
<td>SetVolume(float Volume)</td>
<td>sf::Sound</td>
<td>[private]</td>
</tr>
<tr>
<td>Sound()</td>
<td>sf::Sound</td>
<td>[private]</td>
</tr>
<tr>
<td>Sound(const SoundBuffer &amp;Buffer, bool Loop=false, float Pitch=1.f, float Volume=100.f, const Vector3f &amp;Position=Vector3f(0, 0, 0))</td>
<td>sf::Sound</td>
<td>[explicit, private]</td>
</tr>
<tr>
<td>Sound(const Sound &amp;Copy)</td>
<td>sf::Sound</td>
<td>[private]</td>
</tr>
<tr>
<td>SoundStream()</td>
<td>sf::SoundStream</td>
<td>[protected]</td>
</tr>
<tr>
<td>Status enum name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stop()</td>
<td>sf::SoundStream</td>
<td></td>
</tr>
<tr>
<td>Stopped enum value</td>
<td>sf::Sound</td>
<td>[private]</td>
</tr>
<tr>
<td>Terminate()</td>
<td>sf::Thread</td>
<td>[private]</td>
</tr>
<tr>
<td>Thread(FuncType Function, void *UserData=NULL)</td>
<td>sf::Thread</td>
<td>[private]</td>
</tr>
<tr>
<td>Thread()</td>
<td>sf::Thread</td>
<td>[private]</td>
</tr>
<tr>
<td>Wait()</td>
<td>sf::Thread</td>
<td>[private]</td>
</tr>
<tr>
<td>~AudioResource()</td>
<td>sf::AudioResource</td>
<td>[private, virtual]</td>
</tr>
<tr>
<td>~Sound()</td>
<td>sf::Sound</td>
<td>[private]</td>
</tr>
<tr>
<td>~SoundStream()</td>
<td>sf::SoundStream</td>
<td>[virtual]</td>
</tr>
<tr>
<td>~Thread()</td>
<td>sf::Thread</td>
<td>[private, virtual]</td>
</tr>
</tbody>
</table>
### sf::SoundStream::SoundStream::Chunk Member List

This is the complete list of members for `sf::SoundStream::SoundStream::Chunk`, including all inherited members.

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NbSamples</td>
<td>sf::SoundStream::SoundStream::Chunk</td>
</tr>
<tr>
<td>Samples</td>
<td>sf::SoundStream::SoundStream::Chunk</td>
</tr>
</tbody>
</table>

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
<table>
<thead>
<tr>
<th>Main Page</th>
<th>Namespaces</th>
<th>Classes</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>File List</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Sprite.hpp

#ifndef SFML_SPRITE_HPP
#define SFML_SPRITE_HPP

// Headers
#include <SFML/System/Resource.hpp>
#include <SFML/Graphics/Drawable.hpp>
#include <SFML/Graphics/Rect.hpp>

namespace sf {


class Image;

class SFML_API Sprite : public Drawable
{
    public :
        Sprite();

    explicit Sprite(const Image& Img, const Vector2f& Size);

    void SetImage(const Image& Img);

    void SetSubRect(const IntRect& SubRect);

    void Resize(float Width, float Height);

    void Resize(const Vector2f& Size);

    void FlipX(bool Flipped);

    void FlipY(bool Flipped);

    const Image* GetImage() const;

    const IntRect& GetSubRect() const;

    Vector2f GetSize() const;

    Color GetPixel(unsigned int X, unsigned int Y);

    protected :
        virtual void Render(RenderTarget& Target) const;

    private :
        // Member data
ResourcePtr<Image> myImage;
IntRect mySubRect;
bool myIsFlippedX;
bool myIsFlippedY;
};

// namespace sf

#endif // SFML_SPRITE_HPP
sf::Sprite Member List

This is the complete list of members for **sf::Sprite**, including all inherited members.

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Drawable</strong> (const Vector2f &amp;Position=Vector2f(0, 0), const Vector2f &amp;Scale=Vector2f(1, 1), float Rotation=0.f, const Color &amp;Col=Color(255, 255, 255, 255))</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td><strong>FlipX</strong> (bool Flipped)</td>
<td>sf::Sprite</td>
</tr>
<tr>
<td><strong>FlipY</strong> (bool Flipped)</td>
<td>sf::Sprite</td>
</tr>
<tr>
<td><strong>GetBlendMode</strong> () const</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td><strong>GetCenter</strong> () const</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td><strong>GetColor</strong> () const</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td><strong>GetImage</strong> () const</td>
<td>sf::Sprite</td>
</tr>
<tr>
<td><strong>GetInverseMatrix</strong> () const</td>
<td>sf::Drawable [protected]</td>
</tr>
<tr>
<td><strong>GetMatrix</strong> () const</td>
<td>sf::Drawable [protected]</td>
</tr>
<tr>
<td><strong>GetPixel</strong> (unsigned int X, unsigned int Y) const</td>
<td>sf::Sprite</td>
</tr>
<tr>
<td><strong>GetPosition</strong> () const</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td><strong>GetRotation</strong> () const</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td><strong>GetScale</strong> () const</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td><strong>GetSize</strong> () const</td>
<td>sf::Sprite</td>
</tr>
<tr>
<td><strong>GetSubRect</strong> () const</td>
<td>sf::Sprite</td>
</tr>
<tr>
<td><strong>Move</strong> (float OffsetX, float OffsetY)</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td><strong>Move</strong> (const Vector2f &amp;Offset)</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td><strong>Render</strong> (RenderTarget &amp;Target) const</td>
<td>sf::Sprite [protected, virtual]</td>
</tr>
<tr>
<td><strong>Resize</strong> (float Width, float Height)</td>
<td>sf::Sprite</td>
</tr>
<tr>
<td><strong>Resize</strong> (const Vector2f &amp;Size)</td>
<td>sf::Sprite</td>
</tr>
<tr>
<td><strong>Rotate</strong> (float Angle)</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td><strong>Scale</strong> (float FactorX, float FactorY)</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td><strong>Scale</strong> (const Vector2f &amp;Factor)</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td><strong>SetBlendMode</strong> (Blend::Mode Mode)</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td><strong>SetCenter</strong> (float centerX, float centerY)</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td><strong>SetCenter</strong> (const Vector2f &amp;Center)</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td><strong>SetColor</strong> (const Color &amp;Col)</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td>Method</td>
<td>Class</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td><code>SetImage(const Image &amp;Img)</code></td>
<td><code>sf::Sprite</code></td>
</tr>
<tr>
<td><code>setPosition(float X, float Y)</code></td>
<td><code>sf::Drawable</code></td>
</tr>
<tr>
<td><code>setPosition(const Vector2f &amp;Position)</code></td>
<td><code>sf::Drawable</code></td>
</tr>
<tr>
<td><code>setRotation(float Rotation)</code></td>
<td><code>sf::Drawable</code></td>
</tr>
<tr>
<td><code>setScale(float ScaleX, float ScaleY)</code></td>
<td><code>sf::Drawable</code></td>
</tr>
<tr>
<td><code>setScale(const Vector2f &amp;Scale)</code></td>
<td><code>sf::Drawable</code></td>
</tr>
<tr>
<td><code>setScaleX(float FactorX)</code></td>
<td><code>sf::Drawable</code></td>
</tr>
<tr>
<td><code>setScaleY(float FactorY)</code></td>
<td><code>sf::Drawable</code></td>
</tr>
<tr>
<td><code>setSubRect(const IntRect &amp;SubRect)</code></td>
<td><code>sf::Sprite</code></td>
</tr>
<tr>
<td><code>setX(float X)</code></td>
<td><code>sf::Drawable</code></td>
</tr>
<tr>
<td><code>setY(float Y)</code></td>
<td><code>sf::Drawable</code></td>
</tr>
<tr>
<td><code>sprite()</code></td>
<td><code>sf::Drawable</code></td>
</tr>
<tr>
<td><code>sprite(const Image &amp;Img, const Vector2f &amp;Position=Vector2f(0, 0), const Vector2f &amp;Scale=Vector2f(1, 1), float Rotation=0.f, const Color &amp;Col=Color(255, 255, 255, 255))</code></td>
<td><code>sf::Sprite [explicit]</code></td>
</tr>
<tr>
<td><code>transformToGlobal(const sf::Vector2f &amp;Point) const</code></td>
<td><code>sf::Drawable</code></td>
</tr>
<tr>
<td><code>transformToLocal(const sf::Vector2f &amp;Point) const</code></td>
<td><code>sf::Drawable</code></td>
</tr>
<tr>
<td><code>~drawable()</code></td>
<td><code>sf::Drawable [virtual]</code></td>
</tr>
</tbody>
</table>
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)
//
// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.
//
// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:
//
// 1. The origin of this software must not be misrepresented;
// you must not claim that you wrote the original software.
// If you use this software in a product, an acknowledgment
// in the product documentation would be appreciated but is not required.
//
// 2. Altered source versions must be plainly marked as such,
// and must not be misrepresented as being the original software.
//
// 3. This notice may not be removed or altered.

// Headers
#include <SFML/Graphics/Sprite.hpp>
#include <SFML/Graphics/Image.hpp>
#include <SFML/Graphics/GraphicsContext.hpp>

namespace sf {

Sprite::Sprite() : mySubRect(0, 0, 1, 1), myIsFlippedX(false),
Sprite::Sprite(const Image& Img, const Vector2f& Position, Drawable(Position, Scale, Rotation, Col), mySubRect(0, 0, 1, 1), myIsFlippedX(false), myIsFlippedY(false) {
    SetImage(Img);
}

void Sprite::SetImage(const Image& Img) {
    // If there was no source image before and the new image is valid, adjust the source rectangle
    if (!myImage && (ImgGetWidth() > 0) && (ImgGetHeight() > 0)) {
        SetSubRect(IntRect(0, 0, ImgGetWidth(), ImgGetHeight()),
    }
    // Assign the new image
    myImage = &Img;
}

void Sprite::SetSubRect(const IntRect& SubRect) {
    mySubRect = SubRect;
}

void Sprite::Resize(float Width, float Height) {

int LocalWidth = mySubRect.GetWidth();
int LocalHeight = mySubRect.GetHeight();

if ((LocalWidth > 0) && (LocalHeight > 0))
    SetScale(Width / LocalWidth, Height / LocalHeight);

void Sprite::Resize(const Vector2f& Size)
{
    Resize(Size.x, Size.y);
}

void Sprite::FlipX(bool Flipped)
{
    myIsFlippedX = Flipped;
}

void Sprite::FlipY(bool Flipped)
{
    myIsFlippedY = Flipped;
}

const Image* Sprite::GetImage() const
{
    return myImage;
}

const IntRect& Sprite::GetSubRect() const
{
    return mySubRect;
}
Vector2f Sprite::GetSize() const
{
    return Vector2f(mySubRect.GetWidth()) * GetScale();
}

Color Sprite::GetPixel(unsigned int X, unsigned int Y)
{
    if (myImage)
    {
        unsigned int ImageX = mySubRect.Left + X;
        unsigned int ImageY = mySubRect.Top + Y;
        if (myIsFlippedX) ImageX = mySubRectGetWidth();
        if (myIsFlippedY) ImageY = mySubRect.GetHeight();
        return myImage->GetPixel(ImageX, ImageY);
    }
    else
    {
        return GetColor();
    }
}

void Sprite::Render(RenderTarget&) const
{
    // Get the sprite size
    float Width = static_cast<float>(mySubRectGetWidth());
    float Height = static_cast<float>(mySubRect.GetHeight());
    // Check if the image is valid
    if (myImage && (myImage->GetWidth() > 0) &&
    {
        // Use the "offset trick" to get pixel-perfect rendering
        // see http://www.opengl.org/resources/faq/technical/transformations.htm#tran0030
GLCheck(glTranslatef(0.375f, 0.375f, 0.f));

// Bind the texture
myImage->Bind();

// Calculate the texture coordinates
FloatRect TexCoords = myImage->GetTexCoords(mySubRect);
FloatRect Rect(myIsFlippedX ? TexCoords.
myIsFlippedY ? TexCoords.
myIsFlippedX ? TexCoords.
myIsFlippedY ? TexCoords.

// Draw the sprite's triangles
glBegin(GL_QUADS);
   glTexCoord2f(Rect.Left, Rect.Top);
   glTexCoord2f(Rect.Left, Rect.Bottom);
   glTexCoord2f(Rect.Right, Rect.Bottom);
   glTexCoord2f(Rect.Right, Rect.Top);
   glEnd();
else {
   // Disable texturing
   GLCheck(glDisable(GL_TEXTURE_2D));

   // Draw the sprite's triangles
   glBegin(GL_QUADS);
      glVertex2f(0, 0);
      glVertex2f(0, Height);
      glVertex2f(Width, Height);
      glVertex2f(Width, 0);
   glEnd();
}
} // namespace sf
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.

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

#ifndef SFML_STRING_HPP
#define SFML_STRING_HPP

// Headers
#include <SFML/System/Resource.hpp>
#include <SFML/System/Unicode.hpp>
#include <SFML/Graphics/Drawable.hpp>
#include <SFML/Graphics/Font.hpp>
#include <SFML/Graphics/Rect.hpp>
#include <string>
#endif
namespace sf
{
class SFML_API String : public Drawable
{
public:

    enum Style
    {
        Regular = 0,
        Bold = 1 << 0,
        Italic = 1 << 1,
        Underlined = 1 << 2
    }

    String();

    explicit String(const Unicode::Text& Text);

    void SetText(const Unicode::Text& Text);

    void SetFont(const Font& CharFont);

    void SetSize(float Size);

    void SetStyle(unsigned long TextStyle);

    const Unicode::Text& GetText() const;

    const Font& GetFont() const;

    float GetSize() const;

    unsigned long GetStyle() const;

    sf::Vector2f GetCharacterPos(std::size_t Index);
FloatRect GetRect() const;

protected :

virtual void Render(RenderTarget &Target) const;

private :

void RecomputeRect();

// Member data
Unicode::Text myText;
ResourcePtr<Font> myFont;
float mySize;
unsigned long myStyle;
FloatRect myBaseRect;
bool myNeedRectUpdate;

};

} // namespace sf

#endif // SFML_STRING_HPP
sf::String Member List

This is the complete list of members for sf::String, including all inherited members.

<table>
<thead>
<tr>
<th>Bold enum value</th>
<th>sf::String</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Drawable</strong> (const Vector2f &amp;Position=Vector2f(0, 0), const Vector2f &amp;Scale=Vector2f(1, 1), float Rotation=0.f, const Color &amp;Col=Color(255, 255, 255))</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td><strong>GetBlendMode</strong>() const</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td><strong>GetCenter</strong>() const</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td><strong>GetCharacterPos</strong>(std::size_t Index) const</td>
<td>sf::String</td>
</tr>
<tr>
<td><strong>GetColor</strong>() const</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td><strong>GetFont</strong>() const</td>
<td>sf::String</td>
</tr>
<tr>
<td><strong>GetInverseMatrix</strong>() const</td>
<td>sf::Drawable [protected]</td>
</tr>
<tr>
<td><strong>GetPosition</strong>() const</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td><strong>GetPosition</strong>() const</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td><strong>GetRect</strong>() const</td>
<td>sf::String</td>
</tr>
<tr>
<td><strong>GetRotation</strong>() const</td>
<td>sf::String</td>
</tr>
<tr>
<td><strong>GetScale</strong>() const</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td><strong>GetSize</strong>() const</td>
<td>sf::String</td>
</tr>
<tr>
<td><strong>GetStyle</strong>() const</td>
<td>sf::String</td>
</tr>
<tr>
<td><strong>GetText</strong>() const</td>
<td>sf::String</td>
</tr>
<tr>
<td><strong>Italic</strong> enum value</td>
<td>sf::String</td>
</tr>
<tr>
<td><strong>Move</strong>(float OffsetX, float OffsetY)</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td><strong>Move</strong>(const Vector2f &amp;Offset)</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td><strong>Regular</strong> enum value</td>
<td>sf::String</td>
</tr>
<tr>
<td><strong>Render</strong>(RenderTarget &amp;Target) const</td>
<td>sf::String [protected, virtual]</td>
</tr>
<tr>
<td><strong>Rotate</strong>(float Angle)</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td><strong>Scale</strong>(float FactorX, float FactorY)</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td><strong>Scale</strong>(const Vector2f &amp;Factor)</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td><strong>SetBlendMode</strong>(Blend::Mode Mode)</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td><strong>SetCenter</strong>(float CenterX, float CenterY)</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td><strong>SetCenter</strong>(const Vector2f &amp;Center)</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td>Function</td>
<td>Type</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>SetColor(const Color &amp;Col)</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td>SetFont(const Font &amp;CharFont)</td>
<td>sf::String</td>
</tr>
<tr>
<td>SetPosition(float X, float Y)</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td>SetPosition(const Vector2f &amp;Position)</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td>SetRotation(float Rotation)</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td>SetScale(float ScaleX, float ScaleY)</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td>SetScale(const Vector2f &amp;Scale)</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td>SetScaleX(float FactorX)</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td>SetScaleY(float FactorY)</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td>SetSize(float Size)</td>
<td>sf::String</td>
</tr>
<tr>
<td>SetStyle(unsigned long TextStyle)</td>
<td>sf::String</td>
</tr>
<tr>
<td>SetText(const Unicode::Text &amp;Text)</td>
<td>sf::String</td>
</tr>
<tr>
<td>SetX(float X)</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td>SetY(float Y)</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td>String()</td>
<td>sf::String</td>
</tr>
<tr>
<td>String(const Unicode::Text &amp;Text, const Font &amp;CharFont=Font::GetDefaultFont(), float Size=30.f)</td>
<td>sf::String [explicit]</td>
</tr>
<tr>
<td>Style enum name</td>
<td>sf::String</td>
</tr>
<tr>
<td>TransformToGlobal(const sf::Vector2f &amp;Point) const</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td>TransformToLocal(const sf::Vector2f &amp;Point) const</td>
<td>sf::Drawable</td>
</tr>
<tr>
<td>Underlined enum value</td>
<td>sf::String</td>
</tr>
<tr>
<td>~Drawable()</td>
<td>sf::Drawable [virtual]</td>
</tr>
</tbody>
</table>
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
In no event will the authors be held liable for any damages arising from
the use of this software.

Permission is granted to anyone to use this software for any purpose,
including commercial applications, and to alter it and redistribute it freely,
subject to the following restrictions:

1. The origin of this software must not be misrepresented;
you must not claim that you wrote the original software.
If you use this software in a product, an acknowledgment
in the product documentation would be appreciated but is not
required.

2. Altered source versions must be plainly marked as such,
and must not be misrepresented as being the original software.

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

// Headers
#include <SFML/Graphics/String.hpp>
#include <SFML/Graphics/Image.hpp>
#include <SFML/Graphics/GraphicsContext.hpp>
#include <locale>

namespace sf
{
  String::String() :
    myFont(&Font::GetDefaultFont()),

mySize (30.f),
myStyle (Regular),
myNeedRectUpdate(true)
{
}

String::String(const Unicode::Text& Text, const
myFont (&CharFont),
mySize (Size),
myStyle (Regular),
myNeedRectUpdate(true)
{
   SetText(Text);
}

void String::SetText(const Unicode::Text& Text)
{
    myNeedRectUpdate = true;
    myText = Text;
}

void String::SetFont(const Font& CharFont)
{
    if (myFont != &CharFont)
    {
        myNeedRectUpdate = true;
        myFont = &CharFont;
    }
}

void String::SetSize(float Size)
{
if (mySize != Size) {
    myNeedRectUpdate = true;
    mySize = Size;
}

void String::SetStyle(unsigned long TextStyle) {
    if (myStyle != TextStyle) {
        myNeedRectUpdate = true;
        myStyle = TextStyle;
    }
}

const Unicode::Text& String::GetText() const {
    return myText;
}

const Font& String::GetFont() const {
    return *myFont;
}

float String::GetSize() const {
    return mySize;
}

unsigned long String::GetStyle() const
00143
00144
00145
00146
00147
00153
00154
00155
00156
00157
00158
00159
00160
00161
00162
00163
00164
00165
00166
00167
00168
00169
00170
00171
00172
00173
00174
00175
00176
00177
00178
00179
00180
00181
00182
00183
00184

{
return myStyle;
}

sf::Vector2f String::GetCharacterPos(std::size_t Index)
{
// First get the UTF32 representation of the text
const Unicode::UTF32String& Text = myText;
// Adjust the index if it's out of range
if (Index > Text.length())
Index = Text.length();

// The final size is based on the text size
float FactorX = mySize / myFont->GetCharacterSize()
float AdvanceY = mySize;

// Compute the position
sf::Vector2f Position;
for (std::size_t i = 0; i < Index; ++i)
{
// Get the current character and its correspondi
Uint32
CurChar = Text[i];
const Glyph& CurGlyph = myFont->GetGlyph(CurChar
float
AdvanceX = CurGlyph.Advance

switch (CurChar)
{
// Handle special characters
case L' ' : Position.x += AdvanceX;
case L'\t' : Position.x += AdvanceX * 4;
case L'\v' : Position.y += AdvanceY * 4;
case L'\n' : Position.y += AdvanceY; Positio
// Regular character : just add its advance
default : Position.x += AdvanceX; break


return Position;

FloatRect String::GetRect() const
{
    if (myNeedRectUpdate)
        const_cast<String*>(this)->RecomputeRect();

    FloatRect Rect;
    Rect.Left = (myBaseRect.Left - GetCenter());
    Rect.Top = (myBaseRect.Top - GetCenter());
    Rect.Right = (myBaseRect.Right - GetCenter());
    Rect.Bottom = (myBaseRect.Bottom - GetCenter());

    return Rect;
}

void String::Render(RenderTarget&) const
{
    // First get the internal UTF-32 string of the text
    const Unicode::UTF32String& Text = myText;

    // No text, no rendering :)  
    if (Text.empty())
        return;

    // Set the scaling factor to get the actual size
    float CharSize = static_cast<float>(myFont->GetCharacterSize());
    float Factor = mySize / CharSize;
    GLCheck(glScalef(Factor, Factor, 1.f));

    // Bind the font texture
myFont->GetImage().Bind();

// Initialize the rendering coordinates
float X = 0.f;
float Y = CharSize;

// Holds the lines to draw later, for underlined style
std::vector<float> UnderlineCoords;
UnderlineCoords.reserve(16);

// Compute the shearing to apply if we're using the italic style
float ItalicCoeff = (myStyle & Italic) ? 0.2f : 0.f;

// Draw one quad for each character
glBegin(GL_QUADS);
for (std::size_t i = 0; i < Text.size(); ++i)
{
    // Get the current character and its corresponding glyph
    Uint32 CurChar = Text[i];
    const Glyph& CurGlyph = myFont->GetGlyph(CurChar);
    int Advance = CurGlyph.Advance;
    IntRect Rect = CurGlyph.Rectangle;
    FloatRect Coord = CurGlyph.TexCoords;

    // If we're using the underlined style and there's a new line,
    // we keep track of the previous line to draw it later
    if ((CurChar == L'\n') && (myStyle & Underlined))
    {
        UnderlineCoords.push_back(X);
        UnderlineCoords.push_back(Y + 2);
    }
}

// Handle special characters
switch (CurChar)
{
    case L' ' : X += Advance; continue;
    case L'\n' : Y += CharSize; X = 0; continue;
}
case L'\t': X += Advance * 4;
    break;
  case L'\v': Y += CharSize * 4;
    break;
  }

  // Draw a textured quad for the current character
  glTexCoord2f(Coord.Left, Coord.Top);
  glTexCoord2f(Coord.Left, Coord.Bottom);
  glTexCoord2f(Coord.Right, Coord.Bottom);
  glTexCoord2f(Coord.Right, Coord.Top);
  
  // If we're using the bold style, we must render the character 4 more times,
  // slightly offseted, to simulate a higher weight
  if (myStyle & Bold)
  {
    static const float OffsetsX[] = {-0.5f, 0.5f, 0.f, 0.f};
    static const float OffsetsY[] = {0.f, 0.f, -0.5f, 0.5f};

    for (int j = 0; j < 4; ++j)
    {
      glTexCoord2f(Coord.Left, Coord.Top);
      glTexCoord2f(Coord.Left, Coord.Bottom);
      glTexCoord2f(Coord.Right, Coord.Bottom);
      glTexCoord2f(Coord.Right, Coord.Top);
    }
  }

  // Advance to the next character
  X += Advance;

  glEnd();

  // Draw the underlines if needed
  if (myStyle & Underlined)
  {
    // Compute the line thickness
    float Thickness = (myStyle & Bold) ? 3.1

// Add the last line (which was not finished with a
UnderlineCoords.push_back(X);
UnderlineCoords.push_back(Y + 2);

// Draw the underlines as quads
GLCheck(glDisable(GL_TEXTURE_2D));
glBegin(GL_QUADS);
for (std::size_t i = 0; i < UnderlineCoords.size(); i += 2)
{
    glVertex2f(0, UnderlineCoords[i + 1]);
    glVertex2f(0, UnderlineCoords[i + 1] + Thickness);
    glVertex2f(UnderlineCoords[i], UnderlineCoords[i + 1] + Thickness);
    glVertex2f(UnderlineCoords[i], UnderlineCoords[i + 1]);
}
glEnd();

void String::RecomputeRect()
{
    // First get the internal UTF-32 string of the text
    const Unicode::UTF32String& Text = myText;
    // Reset the "need update" state
    myNeedRectUpdate = false;
    // No text, empty box :) 
    if (Text.empty())
    {
        myBaseRect = FloatRect(0, 0, 0, 0);
        return;
    }
    // Initial values
    float CurWidth = 0;
    float CurHeight = 0;
00342
00343
00344
00345
00346
00347
00348
00349
00350
00351
00352
00353
00354
00355
00356
00357
00358
00359
00360
00361
00362
00363
00364
00365
00366
00367
00368
00369
00370
00371
00372
00373
00374
00375
00376
00377
00378

float Width
float Height
float Factor

= 0;
= 0;
= mySize / myFont->GetCharacterSize(

// Go through each character
for (std::size_t i = 0; i < Text.size(); ++i)
{
// Get the current character and its correspondi
Uint32
CurChar = Text[i];
const Glyph&
CurGlyph = myFont->GetGlyph(CurCh
float
Advance = CurGlyph.Advance * Fac
const IntRect& Rect
= CurGlyph.Rectangle;

// Handle special characters
switch (CurChar)
{
case L' ' : CurWidth += Advance;
case L'\t' : CurWidth += Advance * 4;
case L'\v' : Height
+= mySize * 4; CurHei
case L'\n' :
Height += mySize;
CurHeight = 0;
if (CurWidth > Width)
Width = CurWidth;
CurWidth = 0;
continue;
}
// Advance to the next character
CurWidth += Advance;

// Update the maximum height
float CharHeight = (myFont->GetCharacterSize() +
if (CharHeight > CurHeight)
CurHeight = CharHeight;
}


// Update the last line
if (CurWidth > Width)
    Width = CurWidth;
Height += CurHeight;

// Add a slight width / height if we're using the bold style
if (myStyle & Bold)
    { 
        Width += 1 * Factor;
        Height += 1 * Factor;
    }

// Add a slight width if we're using the italic style
if (myStyle & Italic)
    { 
        Width += 0.208f * mySize;
    }

// Add a slight height if we're using the underlined style
if (myStyle & Underlined)
    { 
        if (CurHeight < mySize + 4 * Factor)
            Height += 4 * Factor;
    }

// Finally update the rectangle
myBaseRect.Left = 0;
myBaseRect.Top = 0;
myBaseRect.Right = Width;
myBaseRect.Bottom = Height;

} } // namespace sf

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gomila@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
// you must not claim that you wrote the original software.
// If you use this software in a product, an acknowledgment
// in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
// and must not be misrepresented as being the original software.

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

#ifndef SFML_THREADWIN32_HPP
#define SFML_THREADWIN32_HPP

// Headers
#include <SFML/System/NonCopyable.hpp>
#include <windows.h>

namespace sf
{
  class SFML_API Thread : NonCopyable
{  
public:

typedef void (*FuncType)(void*);

Thread(FuncType Function, void* UserData = NULL);

virtual ~Thread();

void Launch();

void Wait();

void Terminate();

protected:

Thread();

private:

virtual void Run();

static unsigned int __stdcall ThreadFunc(void*);

// Member data
HANDLE myHandle;
FuncType myFunction;
void* myUserData;

};

} // namespace sf

#endif // SFML_THREADWIN32_HPP
sf::Thread Member List

This is the complete list of members for sf::Thread, including all inherited members.

<table>
<thead>
<tr>
<th>Member</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>FuncType typedef (defined in sf::Thread)</td>
<td>sf::Thread</td>
<td></td>
</tr>
<tr>
<td>Launch()</td>
<td>sf::Thread</td>
<td></td>
</tr>
<tr>
<td>NonCopyable()</td>
<td>sf::NonCopyable [inline, private]</td>
<td></td>
</tr>
<tr>
<td>Terminate()</td>
<td>sf::Thread</td>
<td></td>
</tr>
<tr>
<td>Thread(FuncType Function, void *UserData=NULL)</td>
<td>sf::Thread</td>
<td></td>
</tr>
<tr>
<td>Thread()</td>
<td>sf::Thread</td>
<td>[protected]</td>
</tr>
<tr>
<td>Wait()</td>
<td>sf::Thread</td>
<td></td>
</tr>
<tr>
<td>~Thread()</td>
<td>sf::Thread</td>
<td>[virtual]</td>
</tr>
</tbody>
</table>

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.

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

// Headers
#include <SFML/System/Win32/Thread.hpp>
#include <process.h>
#include <iostream>

namespace sf {

Thread::Thread()
{  
    myHandle (NULL),
    myFunction(NULL),
}
myUserData(NULL) {
    }

Thread::Thread(Thread::FuncType Function, void* UserData) {
    myHandle = NULL,
    myFunction(Function),
    myUserData(UserData) {
    }

Thread::~Thread() {
    }

void Thread::Launch() {
    // Wait for the thread to finish, in case it
    Wait();
}

void Thread::Wait() {
    // Error?
    if (myHandle == NULL)
        std::cerr << "Failed to create thread" <
    }

void Thread::Wait()
if (myHandle) {
    // Wait for the thread to finish, no timeout
    WaitForSingleObject(myHandle, INFINITE);

    // Don't forget to close the thread handle (__endthreadex doesn't do it)
    CloseHandle(myHandle);
    myHandle = NULL;
}

void Thread::Terminate()
{
    if (myHandle) {
        TerminateThread(myHandle, 0);
        myHandle = NULL;
    }
}

void Thread::Run()
{
    if (myFunction) {
        myFunction(myUserData);
    }
}

unsigned int __stdcall Thread::ThreadFunc(void* UserData)
{
    // The Thread instance is stored in the user data
    Thread* ThreadInstance = reinterpret_cast<br>
    // Forward to the instance
    ThreadInstance->Run();
// Optional, but it is cleaner
_endthreadex(0);

return 0;
}
}
// namespace sf
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.

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

#ifndef SFML_UNICODE_HPP
#define SFML_UNICODE_HPP

// Headers
#include <SFML/Config.hpp>
#include <iterator>
#include <locale>
#include <string>
#include <stdlib.h>

#endif // SFML_UNICODE_HPP
namespace sf {
    class SFML_API Unicode {
        public:
            typedef std::basic_string<Uint8> UTF8String;
            typedef std::basic_string<Uint16> UTF16String;
            typedef std::basic_string<Uint32> UTF32String;

            class SFML_API Text {
                public:
                    Text();
                    Text(const char* Str);
                    Text(const wchar_t* Str);
                    Text(const Uint8* Str);
                    Text(const Uint16* Str);
                    Text(const Uint32* Str);
                    Text(const std::string& Str);
                    Text(const std::wstring& Str);
                    Text(const Unicode::UTF8String& Str);
                    Text(const Unicode::UTF16String& Str);
                    Text(const Unicode::UTF32String& Str);

                    operator std::string () const;
                    operator std::wstring () const;
                    operator Unicode::UTF8String () const;
                    operator Unicode::UTF16String () const;
                    operator const Unicode::UTF32String&() const;

                    private:
                        // Data member
                        sf::Unicode::UTF32String myUTF32String;
template <typename In, typename Out>
static Out UTF32ToANSI(In Begin, In End, Out Output);

template <typename In, typename Out>
static Out ANSIToUTF32(In Begin, In End, Out Output);

template <typename In, typename Out>
static Out UTF8ToUTF16(In Begin, In End, Out Output, Uint16 Replacement = 0);

template <typename In, typename Out>
static Out UTF8ToUTF32(In Begin, In End, Out Output, Uint32 Replacement = 0);

template <typename In, typename Out>
static Out UTF16ToUTF8(In Begin, In End, Out Output, Uint8 Replacement = 0);

template <typename In, typename Out>
static Out UTF16ToUTF32(In Begin, In End, Out Output, Uint16 Replacement = 0);

template <typename In, typename Out>
static std::size_t GetUTF8Length(In Begin, In End);

template <typename In>
static std::size_t GetUTF16Length(In Begin, In End);

template <typename In>
static std::size_t GetUTF32Length(In Begin, In End);

private:
static const std::locale& GetDefaultLocale();

// Static member data
static const int UTF8TrailingBytes[256];
static const Uint32 UTF8Offsets[6];
static const Uint8 UTF8FirstBytes[7];
};

#include <SFML/System/Unicode.inl>

} // namespace sf

#endif // SFML_UNICODE_HPP
# sf::Unicode Member List

This is the complete list of members for sf::Unicode, including all inherited members.

<table>
<thead>
<tr>
<th>Member</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSIToUTF32 (In Begin, In End, Out Output, const std::locale &amp;Locale=GetDefaultLocale())</td>
<td>sf::Unicode [inline, static]</td>
<td>sf::Unicode [inline, static]</td>
</tr>
<tr>
<td>GetUTF16Length (In Begin, In End)</td>
<td>sf::Unicode [inline, static]</td>
<td>sf::Unicode [inline, static]</td>
</tr>
<tr>
<td>GetUTF32Length (In Begin, In End)</td>
<td>sf::Unicode [inline, static]</td>
<td>sf::Unicode [inline, static]</td>
</tr>
<tr>
<td>GetUTF8Length (In Begin, In End)</td>
<td>sf::Unicode [inline, static]</td>
<td>sf::Unicode [inline, static]</td>
</tr>
<tr>
<td>UTF16String typedef (defined in sf::Unicode)</td>
<td>sf::Unicode [inline]</td>
<td>sf::Unicode [inline]</td>
</tr>
<tr>
<td>UTF16ToUTF32 (In Begin, In End, Out Output, Uint32 Replacement= '?')</td>
<td>sf::Unicode [inline, static]</td>
<td>sf::Unicode [inline, static]</td>
</tr>
<tr>
<td>UTF16ToUTF32 (In Begin, In End, Out Output, Uint32 Replacement)</td>
<td>sf::Unicode [inline]</td>
<td>sf::Unicode [inline]</td>
</tr>
<tr>
<td>UTF16ToUTF8 (In Begin, In End, Out Output, Uint8 Replacement= '?')</td>
<td>sf::Unicode [inline, static]</td>
<td>sf::Unicode [inline, static]</td>
</tr>
<tr>
<td>UTF16ToUTF8 (In Begin, In End, Out Output, Uint8 Replacement)</td>
<td>sf::Unicode [inline]</td>
<td>sf::Unicode [inline]</td>
</tr>
<tr>
<td>UTF32String typedef (defined in sf::Unicode)</td>
<td>sf::Unicode [inline]</td>
<td>sf::Unicode [inline]</td>
</tr>
<tr>
<td>UTF32ToANSI (In Begin, In End, Out Output, char Replacement= '?', const std::locale &amp;Locale=GetDefaultLocale())</td>
<td>sf::Unicode [inline, static]</td>
<td>sf::Unicode [inline, static]</td>
</tr>
<tr>
<td>UTF32ToANSI (In Begin, In End, Out Output, char Replacement, const std::locale &amp;Locale)</td>
<td>sf::Unicode [inline]</td>
<td>sf::Unicode [inline]</td>
</tr>
<tr>
<td>UTF32ToUTF16 (In Begin, In End, Out Output, Uint16 Replacement= '?')</td>
<td>sf::Unicode [inline, static]</td>
<td>sf::Unicode [inline, static]</td>
</tr>
<tr>
<td>UTF32ToUTF16 (In Begin, In End, Out Output, Uint16 Replacement)</td>
<td>sf::Unicode [inline]</td>
<td>sf::Unicode [inline]</td>
</tr>
<tr>
<td>UTF32ToUTF8 (In Begin, In End, Out Output, Uint8 Replacement= '?')</td>
<td>sf::Unicode [inline, static]</td>
<td>sf::Unicode [inline, static]</td>
</tr>
<tr>
<td>Function</td>
<td>Parameters</td>
<td>Return Type</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>UTF32ToUTF8</td>
<td>(In Begin, In End, Out Output, Uint8 Replacement)</td>
<td>sf::Unicode</td>
</tr>
<tr>
<td>UTF8String typedef</td>
<td></td>
<td>sf::Unicode</td>
</tr>
<tr>
<td>UTF8ToUTF16</td>
<td>(In Begin, In End, Out Output, Uint16 Replacement= '?')</td>
<td>sf::Unicode</td>
</tr>
<tr>
<td>UTF8ToUTF16</td>
<td>(In Begin, In End, Out Output, Uint16 Replacement)</td>
<td>sf::Unicode</td>
</tr>
<tr>
<td>UTF8ToUTF32</td>
<td>(In Begin, In End, Out Output, Uint32 Replacement= '?')</td>
<td>sf::Unicode</td>
</tr>
<tr>
<td>UTF8ToUTF32</td>
<td>(In Begin, In End, Out Output, Uint32 Replacement)</td>
<td>sf::Unicode</td>
</tr>
</tbody>
</table>

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
<table>
<thead>
<tr>
<th>Main Page</th>
<th>Namespaces</th>
<th>Classes</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>File List</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
//   you must not claim that you wrote the original software.
//   If you use this software in a product, an acknowledgment
//   in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
//   and must not be misrepresented as being the original software.

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

template <typename In, typename Out>
inline Out Unicode::UTF32ToANSI(In Begin, In End, Out Output, {
    #ifdef __MINGW32__
        // MinGW has a almost no support for unicode stuff
        // As a consequence, the MinGW version of this function can only use the default locale
        // and ignores the one passed as parameter while (Begin < End)
        {
            
        }
    
}
char Char = 0;

if (wctomb(&Char, static_cast<wchar_t>(*Begin++)) >= 0)
    *Output++ = Char;
else if (Replacement)
    *Output++ = Replacement;

#else
    // Get the facet of the locale which deals
    // with character conversion
    const std::ctype<wchar_t>& Facet = std::use_facet<
        std::ctype<wchar_t>>(Locale);

    // Use the facet to convert each character
    while (Begin < End)
        *Output++ = Facet.narrow(static_cast<wchar_t>(*Begin++), Replacement);
#endif

return Output;

}
#else

// Get the facet of the locale which deals with character conversion
const std::ctype<wchar_t>& Facet = std::use_facet<std::ctype<wchar_t>>(Locale);

// Use the facet to convert each character
while (Begin < End)
  *Output++ = static_cast<Uint32>(Facet.widen(*Begin++));

#endif

return Output;

}
// Then encode it in UTF-16
if (c < 0xFFFF) {
    // Character can be converted directly to 16 bits, just need to check it's in the valid range
    if ((c >= 0xD800) && (c <= 0xDFFF)) {
        // Invalid character (this range is reserved)
        if (Replacement)
            *Output++ = Replacement;
    } else {
        // Valid character directly convertible to 16 bits
        *Output++ = static_cast<Uint16>(c);
    }
} else if (c > 0x0010FFFF) {
    // Invalid character (greater than the maximum Unicode value)
    if (Replacement)
        *Output++ = Replacement;
} else {
    // Character will be converted to 2 UTF-16 elements
    c -= 0x0010000;
    *Output++ = static_cast<Uint16>(c >> 10 + 0xD800);
    *Output++ = static_cast<Uint16>((c & 0x3FFUL) + 0xDC00);
}

return Output;

template <typename In, typename Out>
inline Out Unicode::UTF8ToUTF32(In Begin, In End, Out Output, Uint32 Replacement) {
  while (Begin < End) {
    Uint32 c = 0;
    int TrailingBytes = UTF8TrailingBytes[static_cast<int>(Begin[0])];
    if (Begin + TrailingBytes < End) {
      // First decode the UTF-8 character (TrailingBytes)
      switch (TrailingBytes) {
        case 5 : c += *Begin++; c <<= 6;
        case 4 : c += *Begin++; c <<= 6;
        case 3 : c += *Begin++; c <<= 6;
        case 2 : c += *Begin++; c <<= 6;
        case 1 : c += *Begin++; c <<= 6;
        case 0 : c += *Begin++;
      }
      c -= UTF8Offsets[TrailingBytes];
    }
    if ((c < 0xD800) || (c > 0xDFFF)) {
      // Valid UTF-32 character
      *Output++ = c;
    } else {
      // Invalid UTF-32 character
      if (Replacement) {
        *Output++ = Replacement;
      }
    }
  }
  return Output;
}
template <typename In, typename Out>
inline Out Unicode::UTF16ToUTF8(In Begin, In End, Out Output, Uint8 Replacement) {
    while (Begin < End) {
        Uint32 c = *Begin++;

        // If it's a surrogate pair, first convert
        if ((c >= 0xD800) && (c <= 0xDBFF)) {
            if (Begin < End) {
                // The second element is valid :
                Uint32 d = *Begin++;
                if ((d >= 0xDC00) && (d <= 0xDFFF))
                    c = static_cast<Uint32>((c - 0xD800) << 10) + (d - 0xDC00) + 0x10000;
            } else {
                // Invalid second element
                if (Replacement) *Output++ = Replacement;
            }
        } else {
            // Valid character
            if (c > 0x0010FFFF) {
                // Invalid character (greater than the maximum unicode value)
                if (Replacement) *Output++ = Replacement;
            } else {
                // Valid character
            }
        }
    }
}

Get number of bytes to write

```c++
int BytesToWrite = 1;
if (c < 0x80) BytesToWrite = 1;
else if (c < 0x800) BytesToWrite = 2;
else if (c < 0x10000) BytesToWrite = 3;
else if (c <= 0x0010FFFF) BytesToWrite = 4;

// Extract bytes to write
Uint8 Bytes[4];
switch (BytesToWrite) {
    case 4 : Bytes[3] = static_cast<
    case 2 : Bytes[1] = static_cast<
    case 1 : Bytes[0] = static_cast<
}

// Add them to the output
const Uint8* CurByte = Bytes;
switch (BytesToWrite) {
    case 4 : *Output++ = *CurByte++;
    case 3 : *Output++ = *CurByte++;
    case 2 : *Output++ = *CurByte++;
    case 1 : *Output++ = *CurByte++;
}
return Output;
```

template <typename In, typename Out>
inline Out Unicode::UTF16ToUTF32(In Begin, In End) {

while (Begin < End) {
    Uint16 c = *Begin++;
    if ((c >= 0xD800) && (c <= 0xDBFF)) {
        // We have a surrogate pair, ie. a character composed of two elements
        if (Begin < End) {
            Uint16 d = *Begin++;
            if ((d >= 0xDC00) && (d <= 0xDFFF)) {
                // The second element is valid: convert the two elements to a UTF-32 character
                *Output++ = static_cast<Uint32>(c); // Valid character directly convertible to UTF-32
            } else {
                // Invalid second element
                if (Replacement) {
                    *Output++ = Replacement;
                }
            }
        } else {
            // Invalid character
            if (Replacement) {
                *Output++ = Replacement;
            }
        }
    } else if ((c >= 0xDC00) && (c <= 0xDFFF)) {
        // Invalid character
        if (Replacement) {
            *Output++ = Replacement;
        }
    } else {
        // Valid character directly convertible to UTF-32
        *Output++ = static_cast<Uint32>(c);
    }
}
return Output;
template<typename In, typename Out>
inline Out Unicode::UTF32ToUTF8(In Begin, In End, Out Output, Uint8 Replacement) {
  while (Begin < End) {
    Uint32 c = *Begin++;
    if (c > 0x0010FFFF) {
      // Invalid character (greater than the maximum unicode value)
      if (Replacement) {
        *Output++ = Replacement;
      } else {
        // Valid character
        // Get number of bytes to write
        int BytesToWrite = 1;
        if (c < 0x80) BytesToWrite = 1;
        else if (c < 0x800) BytesToWrite = 2;
        else if (c < 0x10000) BytesToWrite = 3;
        else if (c <= 0x0010FFFF) BytesToWrite = 4;
        // Extract bytes to write
        Uint8 Bytes[4];
        switch (BytesToWrite) {
          case 4: Bytes[3] = static_cast<uint8_t>(c);
          case 3: Bytes[2] = static_cast<uint8_t>((c >> 24) & 0xFF);
          case 2: Bytes[1] = static_cast<uint8_t>((c >> 16) & 0xFF);
          case 1: Bytes[0] = static_cast<uint8_t>((c >> 8) & 0xFF);
        }
        // Add them to the output
        const Uint8* CurByte = Bytes;
      }  // end of invalid character
    } else {  // Valid character
      // Get number of bytes to write
      int BytesToWrite = 1;
      if (c < 0x80) BytesToWrite = 1;
      else if (c < 0x800) BytesToWrite = 2;
      else if (c < 0x10000) BytesToWrite = 3;
      else if (c <= 0x0010FFFF) BytesToWrite = 4;
      // Extract bytes to write
      Uint8 Bytes[4];
      switch (BytesToWrite) {
        case 4: Bytes[3] = static_cast<uint8_t>(c);
        case 3: Bytes[2] = static_cast<uint8_t>((c >> 24) & 0xFF);
        case 2: Bytes[1] = static_cast<uint8_t>((c >> 16) & 0xFF);
        case 1: Bytes[0] = static_cast<uint8_t>((c >> 8) & 0xFF);
      }
      // Add them to the output
      const Uint8* CurByte = Bytes;
    }  // end of valid character
  }  // end of while
}  // end of Unicode::UTF32ToUTF8
switch (BytesToWrite) {
  case 4 : *Output++ = *CurByte++;
  case 3 : *Output++ = *CurByte++;
  case 2 : *Output++ = *CurByte++;
  case 1 : *Output++ = *CurByte++;
}
return Output;

template <typename In, typename Out>
inline Out Unicode::UTF32ToUTF16(In Begin, In End, Out Output, Uint16 Replacement) {
  while (Begin < End) {
    Uint32 c = *Begin++;
    if (c < 0xFFFF) {
      // Character can be converted directly to 16 bits,
      // just need to check it's in the valid range
      if ((c >= 0xD800) && (c <= 0xDFFF)) {
        // Invalid character (this range is reserved)
        if (Replacement) *Output++ = Replacement;
      } else {
        // Valid character directly convertible to 16 bits
        *Output++ = static_cast<Uint16>(c);
      }
    } else if (c > 0x10FFFF) {
    } else if (c > 0x0010FFFF) {
    } else if (c > 0x00010FFFF) {
    }
// Invalid character (greater than the maximum unicode value)
if (Replacement)
    *Output++ = Replacement;
else
    // Character will be converted to 2 UTF-16 elements
    c -= 0x0010000;
    *Output++ = static_cast<Uint16>((c >> 10) + 0xD800);
    *Output++ = static_cast<Uint16>((c & 0x3FFUL) + 0xDC00);
return Output;

template <typename In>
inline std::size_t Unicode::GetUTF8Length(In Begin, In End)
{
    std::size_t Length = 0;
    while (Begin < End)
    {
        int NbBytes = UTF8TrailingBytes[static_cast<int>(c)];
        if (Begin + NbBytes < End)
            ++Length;
        Begin += NbBytes + 1;
    }
    return Length;
}

template <typename In>
inline std::size_t Unicode::GetUTF16Length(In Begin, In End)
{
std::size_t Length = 0;
while (Begin < End)
{
    if (*Begin >= 0xD800 && *Begin <= 0xDBFF)
    {
        ++Begin;
        if ((Begin < End) && (*Begin >= 0xDC00 && *Begin <= 0xDFFF))
            ++Length;
    }
    else
    {
        ++Length;
    }
    ++Begin;
}
return Length;

template <typename In>
inline std::size_t Unicode::GetUTF32Length(In Begin, In End)
{
    return End - Begin;
}
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
// you must not claim that you wrote the original software.
// If you use this software in a product, an acknowledgment
// in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
// and must not be misrepresented as being the original software.

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

// Headers
#include <SFML/System/Unicode.hpp>
#include <stdexcept>
#include <string.h>

// References :

// http://www.unicode.org/
// http://www.unicode.org/Public/PROGRAMS/CVTUTF
namespace

// Generic utility function to compute the number of characters in a null-terminated string

template <typename T>
std::size_t StrLen(const T* Str)
{
    std::size_t Length = 0;
    while (*Str++) Length++;
    return Length;
}

// Get the current system locale

std::locale GetCurrentLocale()
{
    try
    {
        return std::locale("");  // It seems some implementations don't care
    // (Mac OS, MinGW)
    }
    catch (std::runtime_error&)  // (Mac OS, MinGW)
    {
        return std::locale();
    }
}

namespace sf

// Static member data

const int Unicode::UTF8TrailingBytes[256] =

const Uint32 Unicode::UTF8Offsets[6] =
{
    0x00000000, 0x00003080, 0x000E2080, 0x03C82080, 0xFA082080, 0x82082080
};

const Uint8 Unicode::UTF8FirstBytes[7] =
{
    0x00, 0x00, 0xC0, 0xE0, 0xF0, 0xF8, 0xFC
};

Unicode::Text::Text()
{
    // Nothing to do
}

Unicode::Text::Text(const char* Str)
{
    if (Str)
    {
        std::size_t Length = StrLen(Str);
        if (Length > 0)
        {
            myUTF32String.reserve(Length + 1);
            Unicode::ANSIToUTF32(Str, Str + Length, std::back_inserter(myUTF32String));
        }
    }
}
Unicode::Text::Text(const wchar_t* Str) {
    if (Str) {
        std::size_t Length = StrLen(Str);
        if (Length > 0) {
            // See comments below, in Unicode:
            myUTF32String.reserve(Length + 1);
            switch (sizeof(wchar_t)) {
            case 2:
                Unicode::UTF16ToUTF32(Str, Str + Length,
                                      std::back_inserter(myUTF32String), 0);
                break;
            case 4:
                std::copy(Str, Str + Length,
                           std::back_inserter(myUTF32String));
                break;
            default:
                break;
            }
        }
    }
}

Unicode::Text::Text(const Uint8* Str) {
    if (Str) {
        std::size_t Length = StrLen(Str);
        if (Length > 0) {
            myUTF32String.reserve(Length + 1);
            Unicode::UTF8ToUTF32(Str, Str + Length,
                                  std::back_inserter(myUTF32String), 0);
        }
    }
}

Unicode::Text::Text(const Uint16* Str) {
    if (Str) {
        std::size_t Length = StrLen(Str);
        if (Length > 0) {
            myUTF32String.reserve(Length + 1);
            Unicode::UTF8ToUTF32(Str, Str + Length,
                                  std::back_inserter(myUTF32String), 0);
        }
    }
}

Unicode::Text::Text(const Uint16* Str) {
    if (Str) {
        std::size_t Length = StrLen(Str);
        if (Length > 0) {
            myUTF32String.reserve(Length + 1);
            Unicode::UTF8ToUTF32(Str, Str + Length,
                                  std::back_inserter(myUTF32String), 0);
        }
    }
}
myUTF32String.reserve(Length + 1);
Unicode::UTF16ToUTF32(Str, Str + Length, std::back_inserter(myUTF32String), 0);

Unicode::Text::Text(const Uint32* Str)
{
    if (Str)
    {
        myUTF32String = Str;
    }
}

Unicode::Text::Text(const std::string& Str)
{
    myUTF32String.reserve(Str.length() + 1);
    Unicode::ANSIToUTF32(Str.begin(), Str.end(), std::back_inserter(myUTF32String));
}

Unicode::Text::Text(const std::wstring& Str)
{
    // This function assumes that 2-byte large wchar_t are encoded using UTF-16
    // 4-byte large wchar_t are encoded using UTF-32
    // Is that always true? (some platforms may differ)
    // The macro __STDC_ISO_10646__ should help
    myUTF32String.reserve(Str.length() + 1);

    switch (sizeof(wchar_t))
    {
        // wchar_t uses UTF-16 -- need a conversion
        case 2 :
            { Unicode::UTF16ToUTF32(Str.begin(), Str.end(), std::back_inserter(myUTF32String)); break; }
        // wchar_t uses UTF-32 -- direct copy
        case 4 :
            { }
std::copy(Str.begin(), Str.end(), std::back_inserter(myUTF32String));

break;
}

// This should never happen
default : break;
}

Unicode::Text::Text(const Unicode::UTF8String& Str) {
    myUTF32String.reserve(Str.length() + 1);
    Unicode::UTF8ToUTF32(Str.begin(), Str.end(), std::back_inserter(myUTF32String), 0);
}

Unicode::Text::Text(const Unicode::UTF16String& Str) {
    myUTF32String.reserve(Str.length() + 1);
    Unicode::UTF16ToUTF32(Str.begin(), Str.end(), std::back_inserter(myUTF32String), 0);
}

Unicode::Text::Text(const Unicode::UTF32String& Str) {
    myUTF32String = Str;
}

Unicode::Text::operator std::string() const {
    std::string Output;
    Output.reserve(myUTF32String.length() + 1);
    Unicode::UTF32ToANSI(myUTF32String.begin(), std::back_inserter(Output), 0, Unicode::GetDefaultLocale());
    return Output;
}

Unicode::Text::operator std::wstring() const {
    // This function assumes that 2-byte large wchar_t are encoded using UTF-16
    // 4-byte large wchar_t are encoded using UTF-32
    // Is that always true? (some platforms may use JIS Japanese encoding)
    // The macro __STDC_ISO_10646__ should help
    
}
std::wstring Output;
Output.reserve(myUTF32String.length() + 1);

// Select the proper function according to the (supposed) wchar_t system encoding
switch (sizeof(wchar_t))
{
    // wchar_t uses UTF-16 -- need a conversion
    case 2 :
        UTF32ToUTF16(myUTF32String.begin(), myUTF32String.end(), std::back_inserter(Output), 0);
        break;

    // wchar_t uses UTF-32 -- direct copy
    case 4 :
        std::copy(myUTF32String.begin(), myUTF32String.end(), std::back_inserter(Output));
        break;

    // This should never happen
    default : break;

    return Output;
}

Unicode::Text::operator sf::Unicode::UTF8String() const
{
    Unicode::UTF8String Output;
    Output.reserve(myUTF32String.length() * 4 + 1);
    Unicode::UTF32ToUTF8(myUTF32String.begin(), myUTF32String.end(), std::back_inserter(Output), 0);
    return Output;
}

Unicode::Text::operator sf::Unicode::UTF16String() const
{
    Unicode::UTF16String Output;
    Output.reserve(myUTF32String.length() * 2 + 1);
Unicode::UTF32ToUTF16(myUTF32String.begin(),
    return Output;
Unicode::Text::operator const sf::Unicode::UTF32String&()
    return myUTF32String;
const std::locale& Unicode::GetDefaultLocale()
} // namespace sf
**sf::Unicode::Unicode::Text Member List**

This is the complete list of members for **sf::Unicode::Unicode::Text**, including all inherited members.

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>operator const Unicode::UTF32String &amp;()</code> const</td>
<td><strong>sf::Unicode::Unicode::Text</strong></td>
</tr>
<tr>
<td><code>operator std::string()</code> const</td>
<td><strong>sf::Unicode::Unicode::Text</strong></td>
</tr>
<tr>
<td><code>operator std::wstring()</code> const</td>
<td><strong>sf::Unicode::Unicode::Text</strong></td>
</tr>
<tr>
<td><code>operator Unicode::UTF16String()</code> const</td>
<td><strong>sf::Unicode::Unicode::Text</strong></td>
</tr>
<tr>
<td><code>operator Unicode::UTF8String()</code> const</td>
<td><strong>sf::Unicode::Unicode::Text</strong></td>
</tr>
<tr>
<td><code>Text()</code></td>
<td><strong>sf::Unicode::Unicode::Text</strong></td>
</tr>
<tr>
<td><code>Text(const char *Str)</code></td>
<td><strong>sf::Unicode::Unicode::Text</strong></td>
</tr>
<tr>
<td><code>Text(const wchar_t *Str)</code> (defined in <strong>sf::Unicode::Unicode::Text</strong>)</td>
<td><strong>sf::Unicode::Unicode::Text</strong></td>
</tr>
<tr>
<td><code>Text(const Uint8 *Str)</code> (defined in <strong>sf::Unicode::Unicode::Text</strong>)</td>
<td><strong>sf::Unicode::Unicode::Text</strong></td>
</tr>
<tr>
<td><code>Text(const Uint16 *Str)</code> (defined in <strong>sf::Unicode::Unicode::Text</strong>)</td>
<td><strong>sf::Unicode::Unicode::Text</strong></td>
</tr>
<tr>
<td><code>Text(const Uint32 *Str)</code> (defined in <strong>sf::Unicode::Unicode::Text</strong>)</td>
<td><strong>sf::Unicode::Unicode::Text</strong></td>
</tr>
<tr>
<td><code>Text(const std::string &amp;Str)</code> (defined in <strong>sf::Unicode::Unicode::Text</strong>)</td>
<td><strong>sf::Unicode::Unicode::Text</strong></td>
</tr>
<tr>
<td><code>Text(const std::wstring &amp;Str)</code> (defined in <strong>sf::Unicode::Unicode::Text</strong>)</td>
<td><strong>sf::Unicode::Unicode::Text</strong></td>
</tr>
<tr>
<td><code>Text(const Unicode::UTF8String &amp;Str)</code> (defined in <strong>sf::Unicode::Unicode::Text</strong>)</td>
<td><strong>sf::Unicode::Unicode::Text</strong></td>
</tr>
<tr>
<td><code>Text(const Unicode::UTF16String &amp;Str)</code> (defined in <strong>sf::Unicode::Unicode::Text</strong>)</td>
<td><strong>sf::Unicode::Unicode::Text</strong></td>
</tr>
<tr>
<td><code>Text(const Unicode::UTF32String &amp;Str)</code> (defined in <strong>sf::Unicode::Unicode::Text</strong>)</td>
<td><strong>sf::Unicode::Unicode::Text</strong></td>
</tr>
</tbody>
</table>

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
| File List |
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)
//
// This software is provided 'as-is', without any expressed or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.
//
// Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:
//
// 1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.
// 2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.
// 3. This notice may not be removed or altered from any source distribution.

#ifndef SFML_VECTOR2_HPP
#define SFML_VECTOR2_HPP

namespace sf
{
    template<typename T>
    class Vector2
    {
        public:

    #ifdef SFML_VECTOR2_HPP
    #define SFML_VECTOR2_HPP
    #endif
}
Vector2();

Vector2(T X, T Y);

// Member data
T x;
T y;
};

// Template methods

template <typename T>
Vector2<T> operator -(const Vector2<T>& V);

// Increment and decrement operators

template <typename T>
Vector2<T>& operator +=(Vector2<T>& V1, const Vector2<T>& V);

template <typename T>
Vector2<T>& operator -=(Vector2<T>& V1, const Vector2<T>& V);

// Arithmetic operators

template <typename T>
Vector2<T> operator +(const Vector2<T>& V1, const Vector2<T>& V);

template <typename T>
Vector2<T> operator -(const Vector2<T>& V1, const Vector2<T>& V);

template <typename T>
Vector2<T> operator *(const Vector2<T>& V, T X);

template <typename T>
Vector2<T> operator *(T X, const Vector2<T>& V);

template <typename T>
Vector2<T>& operator *=(Vector2<T>& V, T X);

template <typename T>
Vector2<T>& operator /=(Vector2<T>& V, T X);

// Divides a Vector2 by another Vector2 and T

// Multiplication operator

Vector2<T>& operator /=(Vector2<T>& V, T X);

template <typename T>
tbool operator ==((const Vector2<T>& V1, const Vector2<T>& V2);

ttemplate <typename T>
tbool operator !=((const Vector2<T>& V1, const Vector2<T>& V2);

#include <SFML/System/Vector2.inl>

// Define the most common types
typedef Vector2<int> Vector2i;
ttypedef Vector2<float> Vector2f;

} // namespace sf

# endif // SFML_VECTOR2_HPP

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::

Documentation generated by doxygen 1.5.2 ::
sf::Vector2< T > Member List

This is the complete list of members for sf::Vector2< T >, including all inherited members.

<table>
<thead>
<tr>
<th>Member</th>
<th>sf::Vector2&lt; T &gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vector2()</td>
<td>sf::Vector2&lt; T &gt;</td>
</tr>
<tr>
<td>Vector2(T X, T Y)</td>
<td>sf::Vector2&lt; T &gt;</td>
</tr>
<tr>
<td>x</td>
<td>sf::Vector2&lt; T &gt;</td>
</tr>
<tr>
<td>y</td>
<td>sf::Vector2&lt; T &gt;</td>
</tr>
</tbody>
</table>

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any
// express or implied warranty. In no event will the authors be held liable for
// any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any
// purpose, including commercial applications, and to alter it and
// redistribute it freely, subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an
//    acknowledgment in the product documentation would be appreciated but
//    is not required.

// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.

// 3. This notice may not be removed or altered.

#ifndef SFML VECTOR3_HPP
#define SFML VECTOR3_HPP

namespace sf
{

    //...
}
#endif
Vector3();
Vector3(T X, T Y, T Z);

// Member data
T x;
T y;
T z;
};

template<typename T>
Vector3<T> operator -(const Vector3<T>& V);

template<typename T>
Vector3<T>& operator +=(Vector3<T>& V1, const Vector3<T>& V);

template<typename T>
Vector3<T>& operator -==(Vector3<T>& V1, const Vector3<T>& V);

template<typename T>
Vector3<T> operator +(const Vector3<T>& V1, const Vector3<T>& V);

template<typename T>
Vector3<T> operator *(const Vector3<T>& V, T X);

template<typename T>
Vector3<T> operator *(T X, const Vector3<T>& V);

template<typename T>
Vector3<T>& operator *=(Vector3<T>& V, T X);

template<typename T>
Vector3<T> operator /(const Vector3<T>& V, T X);
template <typename T>
Vector3<T>& operator /=(Vector3<T>& V, T X);

template <typename T>
bool operator %(const Vector3<T>& V1, const Vector3<T>& V2);

#include <SFML/System/Vector3.inl>

// Define the most common types
typedef Vector3<int> Vector3i;
typedef Vector3<float> Vector3f;

} // namespace sf

#endif // SFML_VECTOR3_HPP
This is the complete list of members for `sf::Vector3< T >`, including all inherited members.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Vector3()</code></td>
<td><code>sf::Vector3&lt; T &gt;</code></td>
</tr>
<tr>
<td><code>Vector3(T X, T Y, T Z)</code></td>
<td><code>sf::Vector3&lt; T &gt;</code></td>
</tr>
<tr>
<td><code>x</code></td>
<td><code>sf::Vector3&lt; T &gt;</code></td>
</tr>
<tr>
<td><code>y</code></td>
<td><code>sf::Vector3&lt; T &gt;</code></td>
</tr>
<tr>
<td><code>z</code></td>
<td><code>sf::Vector3&lt; T &gt;</code></td>
</tr>
</tbody>
</table>
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)
//
// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.
//
// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:
//
// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is not required.
//
// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.
//
// 3. This notice may not be removed or altered.

#ifndef SFML_VIDEOMODE_HPP
#define SFML_VIDEOMODE_HPP

namespace sf
{

class SFML_API VideoMode

public:

VideoMode();

VideoMode(unsigned int ModeWidth, unsigned int ModeHeight);

static VideoMode GetDesktopMode();

static VideoMode GetMode(std::size_t Index);

static std::size_t GetModesCount();

bool IsValid() const;

bool operator ==(const VideoMode& Other) const;

bool operator !=(const VideoMode& Other) const;

// Member data

unsigned int Width;

unsigned int Height;

unsigned int BitsPerPixel;

private:

static void InitializeModes();

};

} // namespace sf

#endif // SFML_VIDEOMODE_HPP
This is the complete list of members for `sf::VideoMode`, including all inherited members.

<table>
<thead>
<tr>
<th>Member</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>BitsPerPixel</code></td>
<td><code>sf::VideoMode</code></td>
</tr>
<tr>
<td><code>GetDesktopMode()</code></td>
<td><code>sf::VideoMode</code> [static]</td>
</tr>
<tr>
<td><code>GetMode(std::size_t Index)</code></td>
<td><code>sf::VideoMode</code> [static]</td>
</tr>
<tr>
<td><code>GetModesCount()</code></td>
<td><code>sf::VideoMode</code> [static]</td>
</tr>
<tr>
<td><code>Height</code></td>
<td><code>sf::VideoMode</code></td>
</tr>
<tr>
<td><code>IsValid()</code> const</td>
<td><code>sf::VideoMode</code></td>
</tr>
<tr>
<td><code>operator!=(const VideoMode &amp;Other)</code> const</td>
<td><code>sf::VideoMode</code></td>
</tr>
<tr>
<td><code>operator==(const VideoMode &amp;Other)</code> const</td>
<td><code>sf::VideoMode</code></td>
</tr>
<tr>
<td><code>VideoMode()</code></td>
<td><code>sf::VideoMode</code></td>
</tr>
<tr>
<td><code>VideoMode(unsigned int ModeWidth, unsigned int ModeHeight, unsigned int ModeBpp=32)</code></td>
<td><code>sf::VideoMode</code></td>
</tr>
<tr>
<td><code>Width</code></td>
<td><code>sf::VideoMode</code></td>
</tr>
</tbody>
</table>
Main Page
Namespaces
Classes
Files

File List
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)
//
// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.
//
// Permission is granted to anyone to use this software for any purpose, including commercial applications,
// and to alter it and redistribute it freely, subject to the following restrictions:
// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is not required.
// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.
// 3. This notice may not be removed or altered.

// Headers
#include <SFML/Window/VideoMode.hpp>
#include <SFML/Window/VideoModeSupport.hpp>
#include <algorithm>
#include <vector>

namespace
{
    // Global array of supported video modes
    std::vector<sf::VideoMode> SupportedModes;
// Functor for sorting modes from highest to lowest
struct CompareModes {
    bool operator()(const sf::VideoMode& v1, const sf::VideoMode& v2) {
        if (v1.BitsPerPixel > v2.BitsPerPixel)
            return true;
        else if (v1.BitsPerPixel < v2.BitsPerPixel)
            return false;
        else if (v1.Width > v2.Width)
            return true;
        else if (v1.Width < v2.Width)
            return false;
        else
            return (v1.Height > v2.Height);
    }
};

namespace sf {
    VideoMode::VideoMode() :
        Width(0),
        Height(0),
        BitsPerPixel(0) {
    }
    VideoMode::VideoMode(unsigned int ModeWidth, unsigned int ModeHeight, unsigned int ModeBpp) {

VideoMode VideoMode::GetDesktopMode()
{
    // Directly forward to the video mode support
    return priv::VideoModeSupport::GetDesktopVideoMode();
}

VideoMode VideoMode::GetMode(std::size_t Index)
{
    if (SupportedModes.empty())
        InitializeModes();
    if (Index < GetModesCount())
        return SupportedModes[Index];
    else
        return VideoMode();
}

std::size_t VideoMode::GetModesCount()
{
    if (SupportedModes.empty())
        InitializeModes();
    return SupportedModes.size();
}

bool VideoMode::IsValid() const
{
    if (SupportedModes.empty())
        InitializeModes();
    return true;
}
bool VideoMode::operator == (const VideoMode& Other) {
    return (Width == Other.Width) && (Height == Other.Height) && (BitsPerPixel == Other.BitsPerPixel);
}

bool VideoMode::operator != (const VideoMode& Other) {
    return !(*this == Other);
}

void VideoMode::InitializeModes() {
    // We request the array of valid modes
    priv::VideoModeSupport::GetSupportedVideoModes(SupportedModes);

    // And we sort them from highest to lowest (so that number 0 is the best)
    std::sort(SupportedModes.begin(), SupportedModes.end(), CompareModes());
}

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
// you must not claim that you wrote the original software.
// If you use this software in a product, an acknowledgment
// in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
// and must not be misrepresented as being the original software.

// 3. This notice may not be removed or altered.

#ifndef SFML_VIEW_HPP
#define SFML_VIEW_HPP

// Headers
#include <SFML/Config.hpp>
#include <SFML/Graphics/Rect.hpp>
#include <SFML/Graphics/Matrix3.hpp>
#include <SFML/System/Vector2.hpp>

namespace sf {
class RenderTarget;

class SFML_API View
{
  public :
    explicit View(const FloatRect& ViewRect = FloatRect(),
    View(const sf::Vector2f& Center, const sf::Vector2f& Center);
    void SetCenter(float X, float Y);
    void SetCenter(const sf::Vector2f& Center);
    void SetHalfSize(float HalfWidth, float HalfHeight);
    void SetHalfSize(const sf::Vector2f& HalfSize);
    void SetFromRect(const FloatRect& ViewRect);
    const sf::Vector2f& GetCenter() const;
    const sf::Vector2f& GetHalfSize() const;
    const sf::FloatRect& GetRect() const;
    void Move(float OffsetX, float OffsetY);
    void Move(const sf::Vector2f& Offset);
    void Zoom(float Factor);

  private :
    friend class RenderTarget;
const Matrix3& GetMatrix() const;

void RecomputeMatrix();

// Member data
sf::Vector2f myCenter;
sf::Vector2f myHalfSize;
FloatRect myRect;
Matrix3 myMatrix;
bool myNeedUpdate;

} // namespace sf

#endif // SFML_VIEW_HPP
<table>
<thead>
<tr>
<th>Main Page</th>
<th>Namespaces</th>
<th>Classes</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class List</td>
<td>Class Index</td>
<td>Class Hierarchy</td>
<td>Class Members</td>
</tr>
</tbody>
</table>
sf::View Member List

This is the complete list of members for sf::View, including all inherited members.

<table>
<thead>
<tr>
<th>Function</th>
<th>sf::View</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetCenter() const</td>
<td>sf::View</td>
</tr>
<tr>
<td>GetHalfSize() const</td>
<td>sf::View</td>
</tr>
<tr>
<td>GetRect() const</td>
<td>sf::View</td>
</tr>
<tr>
<td>Move(float OffsetX, float OffsetY)</td>
<td>sf::View</td>
</tr>
<tr>
<td>Move(const sf::Vector2f &amp;Offset)</td>
<td>sf::View</td>
</tr>
<tr>
<td>RenderTarget (defined in sf::View)</td>
<td>sf::View [friend]</td>
</tr>
<tr>
<td>SetCenter(float X, float Y)</td>
<td>sf::View</td>
</tr>
<tr>
<td>SetCenter(const sf::Vector2f &amp;Center)</td>
<td>sf::View</td>
</tr>
<tr>
<td>SetFromRect(const FloatRect &amp;ViewRect)</td>
<td>sf::View</td>
</tr>
<tr>
<td>SetHalfSize(float HalfWidth, float HalfHeight)</td>
<td>sf::View</td>
</tr>
<tr>
<td>SetHalfSize(const sf::Vector2f &amp;HalfSize)</td>
<td>sf::View</td>
</tr>
<tr>
<td>View(const FloatRect &amp;ViewRect=FloatRect(0, 0, 1000, 1000))</td>
<td>sf::View [explicit]</td>
</tr>
<tr>
<td>View(const sf::Vector2f &amp;Center, const sf::Vector2f &amp;HalfSize)</td>
<td>sf::View</td>
</tr>
<tr>
<td>Zoom(float Factor)</td>
<td>sf::View</td>
</tr>
</tbody>
</table>
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.

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

// Headers
#include <SFML/Graphics/View.hpp>
#include <algorithm>

namespace sf
{
    View::View(const FloatRect& ViewRect)
    {
        SetFromRect(ViewRect);
    }
}
View::View(const sf::Vector2f& Center, const sf::Vector2f& myCenter (Center),
myHalfSize (HalfSize), myNeedUpdate(true)
{
}
void View::SetCenter(float X, float Y)
{
    myCenter.x = X;
    myCenter.y = Y;
    myNeedUpdate = true;
}
void View::SetCenter(const sf::Vector2f& Center)
{
    SetCenter(Center.x, Center.y);
}
void View::SetHalfSize(float HalfWidth, float HalfHeight)
{
    myHalfSize.x = HalfWidth;
    myHalfSize.y = HalfHeight;
    myNeedUpdate = true;
}
void View::SetHalfSize(const sf::Vector2f& HalfSize)
{
    SetHalfSize(HalfSize.x, HalfSize.y);
}
void View::SetFromRect(const FloatRect& ViewRect)
{
    SetCenter( (ViewRect.Right + ViewRect.Left) / 2,
              (ViewRect.Right - ViewRect.Left) / 2);
}

const sf::Vector2f& View::GetCenter() const
{
    return myCenter;
}

const sf::Vector2f& View::GetHalfSize() const
{
    return myHalfSize;
}

const sf::FloatRect& View::GetRect() const
{
    // Recompute it if needed
    if (myNeedUpdate)
    {
        const_cast<sf::View*>(this)->RecomputeMatrix();
    }
    return myRect;
}

void View::Move(float OffsetX, float OffsetY)
{
    myCenter.x += OffsetX;
    myCenter.y += OffsetY;
    myNeedUpdate = true;
}
void View::Move(const sf::Vector2f& Offset)
{
    Move(Offset.x, Offset.y);
}

void View::Zoom(float Factor)
{
    if (Factor != 0)
    {
        myHalfSize /= Factor;
        myNeedUpdate = true;
    }
}

const Matrix3& View::GetMatrix() const
{
    // Recompute the matrix if needed
    if (myNeedUpdate)
        const_cast<View*>(this)->RecomputeMatrix();
    return myMatrix;
}

void View::RecomputeMatrix()
{
    // Compute the 4 corners of the view
    float Left = myCenter.x - myHalfSize.x;
    float Top = myCenter.y - myHalfSize.y;
    float Right = myCenter.x + myHalfSize.x;
    float Bottom = myCenter.y + myHalfSize.y;

    // Update the view rectangle - be careful, reversed views are allowed!
}
myRect.Left = std::min(Left, Right);
myRect.Top = std::min(Top, Bottom);
myRect.Right = std::max(Left, Right);
myRect.Bottom = std::max(Top, Bottom);

// Update the projection matrix
myMatrix(0, 0) = 2.f / (Right - Left);
myMatrix(1, 1) = 2.f / (Top - Bottom);
myMatrix(0, 2) = (Left + Right) / (Left - Right);
myMatrix(1, 2) = (Bottom + Top) / (Bottom - Top);
myNeedUpdate = false;
}
}
namespace sf // namespace sf
SFML: Simple and Fast Multimedia Library

Main Page  Namespaces  Classes  Files

File List
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an
//    acknowledgment in the product documentation would be appreciated but is
//    not required.

// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the
//    original software.

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

#ifndef SFML_WINDOW_HPP
#define SFML_WINDOW_HPP

// Headers
#include <SFML/Window/Event.hpp>
#include <SFML/Window/Input.hpp>
#include <SFML/Window/VideoMode.hpp>
#include <SFML/Window/WindowHandle.hpp>
#include <SFML/Window/WindowListener.hpp>
#include <SFML/Window/WindowSettings.hpp>
#include <SFML/Window/WindowStyle.hpp>
#endif
```cpp
#include <SFML/System/Clock.hpp>
#include <SFML/System/NonCopyable.hpp>
#include <queue>
#include <string>

namespace sf
{
    namespace priv
    {
        class WindowImpl;
    }

    class SFML_API Window : public WindowListener, NonCopyable
    {
        public:
            Window();
            Window(VideoMode Mode, const std::string& Title);
            Window(WindowHandle Handle, const WindowSettings&);
            virtual ~Window();
            void Create(VideoMode Mode, const std::string& Title);
            void Create(WindowHandle Handle, const WindowSettings&);
            void Close();

            bool isOpened() const;
            unsigned int getWidth() const;
            unsigned int getHeight() const;
    }
} // namespace sf
```
const WindowSettings& GetSettings() const;
bool GetEvent(Event& EventReceived);
void UseVerticalSync(bool Enabled);
void ShowMouseCursor(bool Show);
void SetCursorPosition(unsigned int Left, unsigned int Top);
void SetPosition(int Left, int Top);
void SetSize(unsigned int Width, unsigned int Height);
void Show(bool State);
void EnableKeyRepeat(bool Enabled);
void SetIcon(unsigned int Width, unsigned int Height);
bool SetActive(bool Active = true) const;
void Display();
const Input& GetInput() const;
void SetFramerateLimit(unsigned int Limit);
float GetFrameTime() const;
void SetJoystickThreshold(float Threshold);
private :
virtual void OnCreate();
virtual void OnEvent(const Event& EventReceived);
void Initialize(priv::WindowImpl* Impl);

// Member data
priv::WindowImpl* myWindow;
std::queue<Event> myEvents;
Input myInput;
Clock myClock;
WindowSettings mySettings;
float myLastFrameTime;
bool myIsExternal;
unsigned int myFramerateLimit;
int mySetCursorPosX;
int mySetCursorPosY;
};
namespace sf

#endif // SFML_WINDOW_HPP

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
This is the complete list of members for `sf::Window`, including all inherited members.

<table>
<thead>
<tr>
<th>Member</th>
<th>Return Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Close()</td>
<td><code>sf::Window</code></td>
</tr>
<tr>
<td><code>Create</code> (VideoMode Mode, const std::string &amp;Title, unsigned long WindowStyle=Style::Resize</td>
<td>Style::Close, const WindowSettings &amp;Params=WindowSettings())</td>
</tr>
<tr>
<td><code>Create</code> (WindowHandle Handle, const WindowSettings &amp;Params=WindowSettings())</td>
<td><code>sf::Window</code></td>
</tr>
<tr>
<td><code>Display</code></td>
<td><code>sf::Window</code></td>
</tr>
<tr>
<td>EnableKeyRepeat (bool Enabled)</td>
<td><code>sf::Window</code></td>
</tr>
<tr>
<td>GetEvent (Event &amp;EventReceived)</td>
<td><code>sf::Window</code></td>
</tr>
<tr>
<td>GetFrameTime() const</td>
<td><code>sf::Window</code></td>
</tr>
<tr>
<td>GetHeight() const</td>
<td><code>sf::Window</code></td>
</tr>
<tr>
<td>GetInput() const</td>
<td><code>sf::Window</code></td>
</tr>
<tr>
<td>GetSettings() const</td>
<td><code>sf::Window</code></td>
</tr>
<tr>
<td>GetWidth() const</td>
<td><code>sf::Window</code></td>
</tr>
<tr>
<td>IsOpened() const</td>
<td><code>sf::Window</code></td>
</tr>
<tr>
<td>NonCopyable ()</td>
<td><code>sf::NonCopyable</code> [inline, private]</td>
</tr>
<tr>
<td>SetActive (bool Active=true) const</td>
<td><code>sf::Window</code></td>
</tr>
<tr>
<td>SetCursorPosition (unsigned int Left, unsigned int Top)</td>
<td><code>sf::Window</code></td>
</tr>
<tr>
<td>SetFramerateLimit (unsigned int Limit)</td>
<td><code>sf::Window</code></td>
</tr>
<tr>
<td>SetIcon (unsigned int Width, unsigned int Height, const Uint8 *Pixels)</td>
<td><code>sf::Window</code></td>
</tr>
<tr>
<td>SetJoystickThreshold (float Threshold)</td>
<td><code>sf::Window</code></td>
</tr>
<tr>
<td>setPosition (int Left, int Top)</td>
<td><code>sf::Window</code></td>
</tr>
<tr>
<td>setSize (unsigned int Width, unsigned int Height)</td>
<td><code>sf::Window</code></td>
</tr>
<tr>
<td>Show (bool State)</td>
<td><code>sf::Window</code></td>
</tr>
<tr>
<td>ShowMouseCursor (bool Show)</td>
<td><code>sf::Window</code></td>
</tr>
<tr>
<td>UseVerticalSync (bool Enabled)</td>
<td><code>sf::Window</code></td>
</tr>
<tr>
<td>Window ()</td>
<td><code>sf::Window</code></td>
</tr>
<tr>
<td>Class Function</td>
<td>Signature</td>
</tr>
<tr>
<td>----------------</td>
<td>----------------------------------------------------</td>
</tr>
<tr>
<td>Window</td>
<td>Window(VideoMode Mode, const std::string &amp;Title, unsigned long WindowStyle=Style::Resize</td>
</tr>
<tr>
<td></td>
<td>Style::Close, const WindowSettings &amp;Params=WindowSettings())</td>
</tr>
<tr>
<td>Window</td>
<td>Window(WindowHandle Handle, const WindowSettings &amp;Params=WindowSettings())</td>
</tr>
<tr>
<td>~Window</td>
<td>~Window()</td>
</tr>
<tr>
<td>~WindowListener</td>
<td>~WindowListener()</td>
</tr>
</tbody>
</table>
# SFML

*Simple and Fast Multimedia Library*

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Namespaces</th>
<th>Classes</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>File List</strong></td>
</tr>
</tbody>
</table>
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)
//
// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.
//
// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:
//
// 1. The origin of this software must not be misrepresented;
// you must not claim that you wrote the original software.
// If you use this software in a product, an acknowledgment
// in the product documentation would be appreciated but is not required.
//
// 2. Altered source versions must be plainly marked as such,
// and must not be misrepresented as being the original software.
//
// 3. This notice may not be removed or altered
#
#ifndef SFML_WINDOWLISTENER_HPP
#define SFML_WINDOWLISTENER_HPP

// Headers
#include <SFML/Config.hpp>

namespace sf
{
    class Event;
}
class SFML_API WindowListener
{
  public :
  
  virtual void OnEvent(const Event& EventReceived) = 0;

  protected :
  
  virtual ~WindowListener() {};

} // namespace sf

#endif // SFML_WINDOWLISTENER_HPP
sf::WindowListener Member List

This is the complete list of members for sf::WindowListener, including all inherited members.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Inheritance</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnEvent(const Event &amp;EventReceived)=0</td>
<td>sf::WindowListener</td>
<td>[pure virtual]</td>
</tr>
<tr>
<td>~WindowListener()</td>
<td>sf::WindowListener</td>
<td>[inline, protected, virtual]</td>
</tr>
</tbody>
</table>

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any
// express or implied warranty.
// In no event will the authors be held liable for
// any damages arising from the use of this software.

// Permission is granted to anyone to use this software
// for any purpose, including commercial applications,
// and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be
// misrepresented; you must not claim that you wrote
// the original software.
// If you use this software in a product, an
// acknowledgment in the product documentation
// would be appreciated but is not required.

// 2. Altered source versions must be plainly marked
// as such, and must not be misrepresented as being
// the original software.

// 3. This notice may not be removed or altered.

#ifndef SFML_WINDOWSETTINGS_HPP
#define SFML_WINDOWSETTINGS_HPP

namespace sf {

struct WindowSettings {

    explicit WindowSettings(unsigned int Depth = DepthBits(Depth),
                            unsigned int StencilBits = Stencil);

    unsigned int DepthBits;
    unsigned int StencilBits;

};

#endif // SFML_WINDOWSETTINGS_HPP
AntialiasingLevel(Antialiasing)
{
}

// Member data
unsigned int DepthBits;
unsigned int StencilBits;
unsigned int AntialiasingLevel;
};

} // namespace sf

#endif // SFML_WINDOWSETTINGS_HPP
**sf::WindowSettings Member List**

This is the complete list of members for *sf::WindowSettings*, including all inherited members.

<table>
<thead>
<tr>
<th>Member</th>
<th>sf::WindowSettings</th>
</tr>
</thead>
<tbody>
<tr>
<td>AntialiasingLevel</td>
<td></td>
</tr>
<tr>
<td>DepthBits</td>
<td></td>
</tr>
<tr>
<td>StencilBits</td>
<td></td>
</tr>
<tr>
<td><strong>WindowSettings</strong> (unsigned int Depth=24, unsigned int Stencil=8, unsigned int Antialiasing=0)</td>
<td>[inline, explicit]</td>
</tr>
</tbody>
</table>
- a -

- Accept() : `sf::SocketTCP`
- Add() : `sf::Selector< Type >`, `sf::SelectorBase`
- AddPoint() : `sf::Shape`
- ANSIToUTF32() : `sf::Unicode`
- Append() : `sf::Packet`
- AudioResource() : `sf::AudioResource`
### Class List

#### Class Index

<table>
<thead>
<tr>
<th>Class</th>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main</td>
<td>Page</td>
</tr>
<tr>
<td>Namespaces</td>
<td></td>
</tr>
<tr>
<td>Classes</td>
<td></td>
</tr>
<tr>
<td>Files</td>
<td></td>
</tr>
</tbody>
</table>

#### Class Hierarchy

- **Main**: Page
- **Namespaces**: Classes, Files

#### Class Members

<table>
<thead>
<tr>
<th>All</th>
<th>Functions</th>
<th>Variables</th>
<th>TypeDefs</th>
<th>Enumerations</th>
<th>Enumerator</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td>g</td>
<td>h</td>
</tr>
</tbody>
</table>

### Main Page

**SFML Simple and Fast Multimedia Library**

<table>
<thead>
<tr>
<th>Button</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main Page</td>
</tr>
<tr>
<td>Class List</td>
</tr>
</tbody>
</table>

---

### a

- **a**: `sf::Color`
- **Advance**: `sf::Glyph`
- **AntialiasingLevel**: `sf::WindowSettings`

### b

- **b**: `sf::Color`
- **BitsPerPixel**: `sf::VideoMode`
- **Black**: `sf::Color`
- **Blue**: `sf::Color`
- **Bottom**: `sf::Rect< T >`

### c

- **Cyan**: `sf::Color`

### d

- **d**
• DepthBits: `sf::WindowSettings`

- g -
  • g: `sf::Color`
  • Green: `sf::Color`

- h -
  • Height: `sf::VideoMode`

- i -
  • Identity: `sf::Matrix3`

- l -
  • Left: `sf::Rect<T>`
  • LocalHost: `sf::IPAddress`

- m -
  • Magenta: `sf::Color`

- n -
  • NbSamples: `sf::SoundStream::SoundStream::Chunk`

- r -
  • r: `sf::Color`
  • Rectangle: `sf::Glyph`
  • Red: `sf::Color`
  • Right: `sf::Rect<T>`
- s -
  
  - Samples : sf::SoundStream::SoundStream::Chunk
  - StencilBits : sf::WindowSettings

- t -
  
  - TexCoords : sf::Glyph
  - Top : sf::Rect< T >
  - Type : sf::Event

- w -
  
  - White : sf::Color
  - Width : sf::VideoMode

- x -
  
  - x : sf::Vector2< T >, sf::Vector3< T >

- y -
  
  - y : sf::Vector2< T >, sf::Vector3< T >
  - Yellow : sf::Color

- z -
  
  - z : sf::Vector3< T >
UTF8String : \texttt{sf::Unicode}
- **EventType**: `sf::Event`
- **Method**: `sf::Http::Http::Request`
- **Status**: `sf::Sound`, `sf::Ftp::Ftp::Response`, `sf::Http::Http::Response`
- **Style**: `sf::String`
- **TransferMode**: `sf::Ftp`
- a -

- Accepted: `sf::Http::Http::Response`
- Ascii: `sf::Ftp`

- b -

- BadCommandSequence: `sf::Ftp::Ftp::Response`
- BadGateway: `sf::Http::Http::Response`
- BadRequest: `sf::Http::Http::Response`
- Binary: `sf::Ftp`
- Bold: `sf::String`

- c -

- ClosingConnection: `sf::Ftp::Ftp::Response`
- ClosingDataConnection: `sf::Ftp::Ftp::Response`
- CommandNotImplemented: `sf::Ftp::Ftp::Response`
- CommandUnknown: `sf::Ftp::Ftp::Response`
- d -

- e -

- f -

- g -

- h -
- i -

- InsufficientStorageSpace : \texttt{sf::Ftp::Ftp::Response}
- InternalServerError : \texttt{sf::Http::Http::Response}
- InvalidFile : \texttt{sf::Ftp::Ftp::Response}
- InvalidResponse : \texttt{sf::Ftp::Ftp::Response, sf::Http::Http::Response}
- Italic : \texttt{sf::String}

- l -

- LocalError : \texttt{sf::Ftp::Ftp::Response}
- LoggedIn : \texttt{sf::Ftp::Ftp::Response}

- m -

- MovedPermanently : \texttt{sf::Http::Http::Response}
- MovedTemporarily : \texttt{sf::Http::Http::Response}
- MultipleChoices : \texttt{sf::Http::Http::Response}

- n -

- NeedAccountToLogIn : \texttt{sf::Ftp::Ftp::Response}
- NeedAccountToStore : \texttt{sf::Ftp::Ftp::Response}
- NeedInformation : \texttt{sf::Ftp::Ftp::Response}
- NeedPassword : \texttt{sf::Ftp::Ftp::Response}
- NoContent : \texttt{sf::Http::Http::Response}
- NotEnoughMemory : \texttt{sf::Ftp::Ftp::Response}
- NotFound : \texttt{sf::Http::Http::Response}
- NotImplemented : \texttt{sf::Http::Http::Response}
- NotLoggedIn : \texttt{sf::Ftp::Ftp::Response}
- NotModified : \texttt{sf::Http::Http::Response}
- o -

- Ok : sf::Ftp::Ftp::Response, sf::Http::Http::Response
- OpeningDataConnection : sf::Ftp::Ftp::Response

- p -

- PageTypeUnknown : sf::Ftp::Ftp::Response
- ParameterNotImplemented : sf::Ftp::Ftp::Response
- ParametersUnknown : sf::Ftp::Ftp::Response
- Paused : sf::Sound
- Playing : sf::Sound
- PointlessCommand : sf::Ftp::Ftp::Response
- Post : sf::Http::Http::Request

- r -

- Regular : sf::String
- RestartMarkerReply : sf::Ftp::Ftp::Response

- s -

- ServiceNotAvailable : sf::Http::Http::Response
- ServiceReady : sf::Ftp::Ftp::Response
- ServiceReadySoon : sf::Ftp::Ftp::Response
- ServiceUnavailable : sf::Ftp::Ftp::Response
- Stopped : sf::Sound
- SystemStatus : sf::Ftp::Ftp::Response
- SystemType : sf::Ftp::Ftp::Response

- t -

- TransferAborted : sf::Ftp::Ftp::Response

- u -
• Unauthorized: **sf::Http::Http::Response**
• Underlined: **sf::String**

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
Here is a list of all documented class members with links to the class documentation for each member:

- **b** -

  - b : [sf::Color](#)
  - BadCommandSequence : [sf::Ftp::Ftp::Response](#)
  - BadGateway : [sf::Http::Http::Response](#)
  - BadRequest : [sf::Http::Http::Response](#)
  - Binary : [sf::Ftp](#)
  - Bind() : [sf::Image](#) , [sf::SocketUDP](#)
  - BitsPerPixel : [sf::VideoMode](#)
  - Black : [sf::Color](#)
  - Blue : [sf::Color](#)
  - Bold : [sf::String](#)
  - Bottom : [sf::Rect< T >](#)
Here is a list of all documented class members with links to the class documentation for each member:

- C -

- CanCapture() : [sf::SoundRecorder](#)
- CanUsePostFX() : [sf::PostFX](#)
- Capture() : [sf::RenderWindow](#)
- ChangeDirectory() : [sf::Ftp](#)
- Circle() : [sf::Shape](#)
- Clear() : [sf::Packet](#), [sf::Selector<Type>](#), [sf::SelectorBase](#), [sf::RenderTarget](#)
- Clock() : [sf::Clock](#)
- Close() : [sf::SocketTCP](#), [sf::SocketUDP](#), [sf::SocketHelper](#), [sf::Window](#)
- ClosingConnection : [sf::Ftp::Ftp::Response](#)
- ClosingDataConnection : [sf::Ftp::Ftp::Response](#)
- Color() : [sf::Color](#)
- CommandNotImplemented : [sf::Ftp::Ftp::Response](#)
- CommandUnknown : sf::Ftp::Ftp::Response
- Connect() : sf::SocketTCP, sf::Ftp
- ConnectionClosed : sf::Ftp::Ftp::Response
- ConnectionFailed : sf::Ftp::Ftp::Response, sf::Http::Http::Response
- Contains() : sf::Rect< T >
- Context() : sf::Context
- ConvertCoords() : sf::RenderWindow
- Copy() : sf::Image
- CopyScreen() : sf::Image
- Create() : sf::Window, sf::Image, sf::Window
- Created : sf::Http::Http::Response
- CreateMaskFromColor() : sf::Image
- Cyan : sf::Color

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
Here is a list of all documented class members with links to the class documentation for each member:

- d -

- DataConnectionAlreadyOpened : sf::Ftp::Ftp::Response
- DataConnectionOpened : sf::Ftp::Ftp::Response
- DataConnectionUnavailable : sf::Ftp::Ftp::Response
- DeleteDirectory() : sf::Ftp
- DeleteFile() : sf::Ftp
- DepthBits : sf::WindowSettings
- DirectoryOk : sf::Ftp::Ftp::Response
- DirectoryResponse() : sf::Ftp::Ftp::DirectoryResponse
- DirectoryStatus : sf::Ftp::Ftp::Response
- Disconnect() : sf::Ftp
- Display() : sf::Window
- Download() : sf::Ftp
- Draw() : sf::RenderTarget
- Drawable() : sf::Drawable
Here is a list of all documented class members with links to the class documentation for each member:

- e -

- Ebcdic : sf::Ftp
- EnableFill() : sf::Shape
- EnableKeyRepeat() : sf::Window
- EnableOutline() : sf::Shape
- EndOfPacket() : sf::Packet
- EnteringPassiveMode : sf::Ftp::Ftp::Response
- EventType : sf::Event
Here is a list of all documented class members with links to the class documentation for each member:

- f -

- FileActionAborted : sf::Ftp::Ftp::Response
- FileActionOk : sf::Ftp::Ftp::Response
- FilenameNotAllowed : sf::Ftp::Ftp::Response
- FileStatus : sf::Ftp::Ftp::Response
- FileUnavailable : sf::Ftp::Ftp::Response
- FlipX() : sf::Sprite
- FlipY() : sf::Sprite
- Font() : sf::Font
- Forbidden : sf::Http::Http::Response

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
Here is a list of all documented class members with links to the class documentation for each member:

- **g** -

- **g** : **sf::Color**
- **Get** : **sf::Http::Http::Request**
- **Get4x4Elements()** : **sf::Matrix3**
- **GetAttenuation()** : **sf::Sound**
- **GetBlendMode()** : **sf::Drawable**
- **GetBody()** : **sf::Http::Http::Response**
- **GetBuffer()** : **sf::SoundBufferRecoder , sf::Sound**
- **GetCenter()** : **sf::Drawable , sf::View**
- **GetChannelsCount()** : **sf::SoundBuffer , sf::SoundStream**
- **GetCharacterPos()** : **sf::String**
- **GetCharacterSize()** : **sf::Font**
- **GetColor()** : **sf::Drawable**
- **GetCount()** : **sf::Ftp::Ftp::ListingResponse**
- **GetData()** : **sf::Packet**
- GetDataSize() : sf::Packet
- GetDefaultFont() : sf::Font
- GetDefaultView() : sf::RenderTarget
- GetDesktopMode() : sf::VideoMode
- GetDirectory() : sf::Ftp::Ftp::DirectoryResponse
- GetDirectoryListing() : sf::Ftp
- GetDuration() : sf::Music, sf::SoundBuffer
- GetElapsedTime() : sf::Clock
- GetErrorStatus() : sf::SocketHelper
- GetEvent() : sf::Window
- GetField() : sf::Http::Http::Response
- GetFilename() : sf::Ftp::Ftp::ListingResponse
- GetFont() : sf::String
- GetFrameTime() : sf::Window
- GetGlobal() : sf::Context
- GetGlobalVolume() : sf::Listener
- GetGlyph() : sf::Font
- GetHalfSize() : sf::View
- GetHeight() : sf::Image, sf::Rect< T >, sf::RenderTarget, sf::RenderWindow, sf::Window
- GetImage() : sf::Font, sf::Sprite
- GetInput() : sf::Window
- GetInverse() : sf::Matrix3
- GetInverseMatrix() : sf::Drawable
- GetJoystickAxis() : sf::Input
- GetLocalAddress() : sf::IPAddress
- GetLoop() : sf::Sound, sf::SoundStream
- GetMajorHttpVersion() : sf::Http::Http::Response
- GetMatrix() : sf::Drawable
- GetMessage() : sf::Ftp::Ftp::Response
- GetMinDistance() : sf::Sound
- GetMinorHttpVersion() : sf::Http::Http::Response
- GetMode() : sf::VideoMode
- GetModesCount() : sf::VideoMode
- GetMouseX() : sf::Input
- GetMouseY() : sf::Input
- GetNbPoints() : sf::Shape
- GetOutlineWidth() : sf::Shape
- GetPitch() : sf::Sound
- GetPixel() : sf::Sprite, sf::Image
- GetPixelsPtr() : sf::Image
- GetPlayingOffset() : sf::Sound, sf::SoundStream
- GetPointColor() : sf::Shape
- GetPointOutlineColor() : sf::Shape
- GetPointPosition() : sf::Shape
- GetPort() : sf::SocketUDP
- GetPosition() : sf::Listener, sf::Sound, sf::Drawable
- GetPublicAddress() : sf::IPAddress
- GetRect() : sf::String, sf::View
- GetRotation() : sf::Drawable
- GetSampleRate() : sf::SoundRecorder, sf::SoundStream, sf::SoundBuffer
- GetSamples() : sf::SoundBuffer
- GetSamplesCount() : sf::SoundBuffer
- GetScale() : sf::Drawable
- GetSeed() : sf::Randomizer
- GetSettings() : sf::Window
- GetSize() : sf::String, sf::Sprite
- GetSocketReady() : sf::SelectorBase, sf::Selector< Type >
- GetStatus() : sf::SoundStream, sf::Ftp::Ftp::Response, sf::Http::Http::Response, sf::Sound
- GetStyle() : sf::String
- GetSubRect() : sf::Sprite
- GetTarget() : sf::Listener
- GetTexCoords() : sf::Image
- `GetText()`: `sf::String`
- `GetUTF16Length()`: `sf::Unicode`
- `GetUTF32Length()`: `sf::Unicode`
- `GetUTF8Length()`: `sf::Unicode`
- `GetValidTextureSize()`: `sf::Image`
- `GetView()`: `sf::RenderTarget`
- `GetVolume()`: `sf::Sound`
- `GetWidth()`: `sf::Window`, `sf::Rect< T >`, `sf::Image`, `sf::RenderWindow`, `sf::RenderTarget`
- `GetWorkingDirectory()`: `sf::Ftp`
- `Glyph()`: `sf::Glyph`
- `Green`: `sf::Color`
Here is a list of all documented class members with links to the class documentation for each member:

- h -

- Head : `sf::Http::Http::Request`
- Height : `sf::VideoMode`
- HelpMessage : `sf::Ftp::Ftp::Response`
- Http() : `sf::Http`
Here is a list of all documented class members with links to the class documentation for each member:

- i -

- Identity : [sf::Matrix3](#)
- Image() : [sf::Image](#)
- Initialize() : [sf::SoundStream](#), [sf::RenderTarget](#)
- Input() : [sf::Input](#)
- InsufficientStorageSpace : [sf::Ftp::Ftp::Response](#)
- InternalServerError : [sf::Http::Http::Response](#)
- Intersects() : [sf::Rect<T>](#)
- InvalidFile : [sf::Ftp::Ftp::Response](#)
- InvalidResponse : [sf::Ftp::Ftp::Response](#), [sf::Http::Http::Response](#)
- InvalidSocket() : [sf::SocketHelper](#)
- IPAddress() : [sf::IpAddress](#)
- IsContextActive() : [sf::Context](#)
- IsJoystickButtonDown() : [sf::Input](#)
- IsKeyDown() : `sf::Input`
- IsMouseButtonDown() : `sf::Input`
- IsOk() : `sf::Ftp::Ftp::Response`
- IsOpened() : `sf::Window`
- IsRelativeToListener() : `sf::Sound`
- IsSmooth() : `sf::Image`
- IsValid() : `sf::IPAddress`, `sf::SocketTCP`, `sf::SocketUDP`, `sf::VideoMode`
- Italic : `sf::String`
Here is a list of all documented class members with links to the class documentation for each member:

- **k** -

  - KeepAlive() : [sf::Ftp](#)
Here is a list of all documented class members with links to the class documentation for each member:

- L -

- Launch() : sf::Thread
- Left : sf::Rect< T >
- Line() : sf::Shape
- Listen() : sf::SocketTCP
- ListingResponse() : sf::Ftp::Ftp::ListingResponse
- LoadFromFile() : sf::Font, sf::Image, sf::PostFX, sf::SoundBuffer
- LoadFromMemory() : sf::SoundBuffer, sf::Font, sf::Image, sf::PostFX
- LoadFromPixels() : sf::Image
- LoadFromSamples() : sf::SoundBuffer
- LocalError : sf::Ftp::Ftp::Response
- LocalHost : sf::IPAddress
- Lock() : sf::Lock, sf::Mutex
• LoggedIn : `sf::Ftp::Ftp::Response`
• Login() : `sf::Ftp`
Here is a list of all documented class members with links to the class documentation for each member:

- m -

- Magenta : sf::Color
- MakeDirectory() : sf::Ftp
- Matrix3() : sf::Matrix3
- Method : sf::Http::Http::Request
- Move() : sf::View , sf::Drawable
- MovedPermanently : sf::Http::Http::Response
- MovedTemporarily : sf::Http::Http::Response
- MultipleChoices : sf::Http::Http::Response
- Music() : sf::Music
- Mutex() : sf::Mutex
Here is a list of all documented class members with links to the class documentation for each member:

- n -

- NbSamples : [sf::SoundStream::SoundStream::Chunk](#)
- NeedAccountToLogIn : [sf::Ftp::Ftp::Response](#)
- NeedAccountToStore : [sf::Ftp::Ftp::Response](#)
- NeedInformation : [sf::Ftp::Ftp::Response](#)
- NeedPassword : [sf::Ftp::Ftp::Response](#)
- NoContent : [sf::Http::Http::Response](#)
- NonCopyable() : [sf::NonCopyable](#)
- NotEnoughMemory : [sf::Ftp::Ftp::Response](#)
- NotFound : [sf::Http::Http::Response](#)
- NotImplemented : [sf::Http::Http::Response](#)
- NotLoggedln : [sf::Ftp::Ftp::Response](#)
- NotModified : [sf::Http::Http::Response](#)
Here is a list of all documented class members with links to the class documentation for each member:

- o -

- Offset() : sf::Rect< T >
- Ok : sf::Ftp::Ftp::Response , sf::Http::Http::Response
- OnEvent() : sf::WindowListener
- OnResourceDestroyed() : sf::ResourcePtr< T >
- OpenFromFile() : sf::Music
- OpenFromMemory() : sf::Music
- OpeningDataConnection : sf::Ftp::Ftp::Response
- operator bool() : sf::Packet
- operator const T *() : sf::ResourcePtr< T >
- operator std::string() : sf::Unicode::Unicode::Text
- operator!=() : sf::VideoMode , sf::Color , sf::IPAddress , sf::SocketTCP , sf::SocketUDP
- operator() : sf::Matrix3
- operator*() : sf::Matrix3 , sf::ResourcePtr< T >
- operator*=() : sf::Color, sf::Matrix3
- operator+=() : sf::Color
- operator->() : sf::ResourcePtr< T >
- operator<() : sf::IPAddress, sf::SocketTCP, sf::SocketUDP
- operator<<() : sf::Packet
- operator<=() : sf::IPAddress
- operator=() : sf::PostFX, sf::Image, sf::Resource< T >, sf::ResourcePtr< T >, sf::Sound, sf::SoundBuffer
- operator==() : sf::IPAddress, sf::SocketUDP, sf::SocketTCP, sf::Color, sf::VideoMode
- operator>() : sf::IPAddress
- operator>=() : sf::IPAddress
- operator>>() : sf::Packet
Here is a list of all documented class members with links to the class documentation for each member:

- p -

- Packet() : sf::Packet
- PageTypeUnknown : sf::Ftp::Ftp::Response
- ParameterNotImplemented : sf::Ftp::Ftp::Response
- ParametersUnknown : sf::Ftp::Ftp::Response
- ParentDirectory() : sf::Ftp
- Pause() : sf::Sound
- Paused : sf::Sound
- Play() : sf::Sound, sf::SoundStream
- Playing : sf::Sound
- PointlessCommand : sf::Ftp::Ftp::Response
- Post : sf::Http::Http::Request
- PostFX() : sf::PostFX
- PreserveOpenGLStates() : sf::RenderTarget
Here is a list of all documented class members with links to the class documentation for each member:

- r -

- r : sf::Color
- Random() : sf::Randomizer
- Receive() : sf::SocketTCP, sf::SocketUDP
- Rect() : sf::Rect< T >
- Rectangle : sf::Glyph, sf::Shape
- Red : sf::Color
- Regular : sf::String
- Remove() : sf::SelectorBase, sf::Selector< Type >
- RenameFile() : sf::Ftp
- Render() : sf::PostFX, sf::Shape, sf::Sprite, sf::String
- RenderTarget() : sf::RenderTarget
- RenderWindow() : sf::RenderWindow
- Request() : sf::Http::Http::Request
- Reset() : sf::Clock
- `ResetBuffer()` : `sf::Sound`
- `Resize()` : `sf::Sprite`
- `Resource()` : `sf::Resource< T >`
- `ResourcePtr()` : `sf::ResourcePtr< T >`
- `Response()` : `sf::Ftp::Ftp::Response`, `sf::Http::Http::Response`
- `RestartMarkerReply` : `sf::Ftp::Ftp::Response`
- `Right` : `sf::Rect< T >`
- `Rotate()` : `sf::Drawable`
Here is a list of all documented class members with links to the class documentation for each member:

- S -

- Samples : sf::SoundStream::SoundStream::Chunk
- SaveToFile() : sf::SoundBuffer , sf::Image
- Scale() : sf::Drawable
- SelectorBase() : sf::SelectorBase
- Send() : sf::SocketTCP , sf::SocketUDP , sf::SocketTCP
- SendRequest() : sf::Http
- ServiceNotAvailable : sf::Http::Http::Response
- ServiceReady : sf::Ftp::Ftp::Response
- ServiceReadySoon : sf::Ftp::Ftp::Response
- ServiceUnavailable : sf::Ftp::Ftp::Response
-SetActive() : sf::Context , sf::Window
- SetAttenuation() : sf::Sound
- SetBlendMode() : sf::Drawable
- SetBlocking() : sf::SocketTCP , sf::SocketUDP ,
sf::SocketHelper
- SetBody() : sf::Http::Http::Request
- SetBuffer() : sf::Sound
- SetCenter() : sf::Drawable, sf::View
- SetColor() : sf::Drawable
- SetCursorPosition() : sf::Window
- SetField() : sf::Http::Http::Request
-SetFont() : sf::String
- SetFramerateLimit() : sf::Window
- SetFromRect() : sf::View
- SetFromTransformations() : sf::Matrix3
- SetGlobalVolume() : sf::Listener
- SetHalfSize() : sf::View
- SetHost() : sf::Http
- SetHttpVersion() : sf::Http::Http::Request
- SetIcon() : sf::Window
- SetImage() : sf::Sprite
- SetJoystickThreshold() : sf::Window
- SetLoop() : sf::Sound, sf::SoundStream
- SetMethod() : sf::Http::Http::Request
- SetMinDistance() : sf::Sound
- SetOutlineWidth() : sf::Shape
- SetParameter() : sf::PostFX
- SetPitch() : sf::Sound
- SetPixel() : sf::Image
- SetPlayingOffset() : sf::Sound
- SetPointColor() : sf::Shape
- SetPointOutlineColor() : sf::Shape
- SetPointPosition() : sf::Shape
- SetPosition() : sf::Drawable, sf::Window, sf::Sound, sf::Listener, sf::Sound
- SetRelativeToListener() : sf::Sound
- SetRotation() : sf::Drawable
• SetScale() : sf::Drawable
• SetScaleX() : sf::Drawable
• SetScaleY() : sf::Drawable
• SetSeed() : sf::Randomizer
• SetSize() : sf::String, sf::Window
• SetSmooth() : sf::Image
• SetStyle() : sf::String
• SetSubRect() : sf::Sprite
• SetTarget() : sf::Listener
• SetText() : sf::String
• SetTexture() : sf::PostFX
• SetURI() : sf::Http::Http::Request
• SetView() : sf::RenderTarget
• SetVolume() : sf::RenderTarget
• SetX() : sf::Drawable
• SetY() : sf::Drawable
• Shape() : sf::Shape
• Show() : sf::Window
• ShowMouseCursor() : sf::Window
• SocketTCP() : sf::SocketTCP
• SocketUDP() : sf::SocketUDP
• Sound() : sf::Sound
• SoundBuffer() : sf::SoundBuffer
• SoundRecorder() : sf::SoundRecorder
• SoundStream() : sf::SoundStream
• Sprite() : sf::Sprite
• Start() : sf::SoundRecorder
• Status : sf::Http::Http::Response, sf::Sound, sf::Ftp::Ftp::Response
• StencilBits : sf::WindowSettings
• Stop() : sf::SoundStream, sf::SoundRecorder, sf::Sound
• Stopped : sf::Sound
• String() : sf::String
• Style : sf::String
• SystemStatus : sf::Ftp::Ftp::Response
• SystemType : sf::Ftp::Ftp::Response
Here is a list of all documented class members with links to the class documentation for each member:

- t -

- **Terminate()**: `sf::Thread`
- **TexCoords**: `sf::Glyph`
- **Text()**: `sf::Unicode::Unicode::Text`
- **Thread()**: `sf::Thread`
- **ToInteger()**: `sf::IPAddress`
- **Top**: `sf::Rect< T >`
- **ToString()**: `sf::IPAddress`
- **TransferAborted**: `sf::Ftp::Ftp::Response`
- **TransferMode**: `sf::Ftp`
- **Transform()**: `sf::Matrix3`
- **TransformToWorld()**: `sf::Drawable`
- **TransformToLocal()**: `sf::Drawable`
- **Type**: `sf::Event`
Here is a list of all documented class members with links to the class documentation for each member:

- **u** -

- Unauthorized : [sf::Http::Http::Response](#)
- Unbind() : [sf::SocketUDP](#)
- Underlined : [sf::String](#)
- Unlock() : [sf::Mutex](#)
- Upload() : [sf::Ftp](#)
- UseVerticalSync() : [sf::Window](#)
- UTF16ToUTF32() : [sf::Unicode](#)
- UTF16ToUTF8() : [sf::Unicode](#)
- UTF32ToANSI() : [sf::Unicode](#)
- UTF32ToUTF16() : [sf::Unicode](#)
- UTF32ToUTF8() : [sf::Unicode](#)
- UTF8String : [sf::Unicode](#)
- UTF8ToUTF16() : [sf::Unicode](#)
- UTF8ToUTF32() : [sf::Unicode](#)
Here is a list of all documented class members with links to the class documentation for each member:

- **V** -

- Vector2() : sf::Vector2< T >
- Vector3() : sf::Vector3< T >
- VideoMode() : sf::VideoMode
- View() : sf::View
Here is a list of all documented class members with links to the class documentation for each member:

- **W** -
  - Wait() : `sf::Selector< Type >`, `sf::SelectorBase`, `sf::Thread`
  - White : `sf::Color`
  - Width : `sf::VideoMode`
  - Window() : `sf::Window`
  - WindowSettings() : `sf::WindowSettings`
Here is a list of all documented class members with links to the class documentation for each member:

- X -

- `x : sf::Vector2< T > , sf::Vector3< T >`

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
Here is a list of all documented class members with links to the class documentation for each member:

- y -

- y : `sf::Vector2< T >`, `sf::Vector3< T >`
- Yellow : `sf::Color`

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
Here is a list of all documented class members with links to the class documentation for each member:

- **Z** -

  - \texttt{sf::Vector3< T >}
  - Zoom() : \texttt{sf::View}

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by \texttt{doxygen 1.5.2} ::
Here is a list of all documented class members with links to the class documentation for each member:

- ~AudioResource() : sf::AudioResource
- ~Context() : sf::Context
- ~Drawable() : sf::Drawable
- ~Ftp() : sf::Ftp
- ~Image() : sf::Image
- ~Lock() : sf::Lock
- ~Music() : sf::Music
- ~Mutex() : sf::Mutex
- ~Packet() : sf::Packet
- ~PostFX() : sf::PostFX
- ~RenderTarget() : sf::RenderTarget
- ~RenderWindow() : sf::RenderWindow
- ~Resource() : sf::Resource< T >
- ~ResourcePtr() : sf::ResourcePtr< T >
- ~Sound() : sf::Sound
- ~SoundBuffer() : sf::SoundBuffer
- ~SoundRecorder() : sf::SoundRecorder
- ~SoundStream() : sf::SoundStream
- ~Thread() : sf::Thread
- ~Window() : sf::Window
- ~WindowListener() : sf::WindowListener
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without and express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose, including commercial applications,
// subject to the following restrictions:
// 1. The origin of this software must not be misrepresented;
// you must not claim that you wrote the original software.
// If you use this software in a product, an acknowledgment
// in the product documentation would be appreciated but is not required.
// 2. Altered source versions must be plainly marked as such,
// and must not be misrepresented as being the original software.
// 3. This notice may not be removed or altered.

#ifndef SFML_WINDOWSTYLE_HPP
#define SFML_WINDOWSTYLE_HPP

namespace sf
{
namespace Style
{
enum { /* enum */
}
None = 0,
Titlebar = 1 << 0,
Resize = 1 << 1,
Close = 1 << 2,
Fullscreen = 1 << 3
};

} // namespace sf

#endif // SFML_WINDOWSTYLE_HPP

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
   Documentation generated by doxygen 1.5.2 ::
<table>
<thead>
<tr>
<th>Code</th>
<th>sf::Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode</td>
<td>sf::Blend</td>
</tr>
</tbody>
</table>

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
## SFML Simple and Fast Multimedia Library

### Namespaces

<table>
<thead>
<tr>
<th>Namespace List</th>
<th>Namespace Members</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>Enumerations</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
<th>l</th>
<th>m</th>
<th>n</th>
<th>p</th>
<th>q</th>
<th>r</th>
<th>s</th>
<th>t</th>
<th>u</th>
</tr>
</thead>
</table>

### Main Page

- a -
  - Add: `sf::Blend`, `sf::Key`
  - Alpha: `sf::Blend`

- b -
  - ButtonCount: `sf::Joy`

- c -
  - Close: `sf::Style`
  - Comma: `sf::Key`
  - Count: `sf::Joy`

- d -
  - Dash: `sf::Key`
  - Divide: `sf::Key`
- e -
  • Equal : sf::Key

- f -
  • Fullscreen : sf::Style

- l -
  • LBracket : sf::Key
  • Left : sf::Key
  • LSystem : sf::Key

- m -
  • Multiply : sf::Blend , sf::Key

- n -
  • None : sf::Blend , sf::Style

- p -
  • Period : sf::Key

- q -
  • Quote : sf::Key

- r -
  • RBracket : sf::Key
  • Resize : sf::Style
- s -
- t -
- u -
<table>
<thead>
<tr>
<th>Main Page</th>
<th>Namespaces</th>
<th>Classes</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>File List</td>
</tr>
<tr>
<td>02958</td>
<td>2, -102, 64, 15, 24, 25, 20, 24, 24, 25, 24, 27,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>02959</td>
<td>101, 0, 12, 2, -97, 64, 11, 3, 26, 25, 25, 3, 1,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>02960</td>
<td>0, 46, 2, -97, -78, 37, 11, 28, -68, 2, -102, 0,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>02961</td>
<td>2, -102, 0, 21, 1, 0, 0, 15, 2, -102, 64, 9, 32,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>02962</td>
<td>-19, -12, -19, 16, -10, -19, -12, -19, 0, 63, -19,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>02963</td>
<td>1, 17, 18, 57, 57, 17, 18, 57, 57, -121, 46, 43,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>02964</td>
<td>30, 43, 31, 0, 51, 32, 49, 31, 1, 45, 38, 43, 31,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>02965</td>
<td>4, 21, 31, 1, 17, 10, 15, 31, 0, 19, 8, 21, 31,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>02966</td>
<td>39, 46, 31, 0, 48, 35, 46, 31, 0, 14, 1, 12, 31,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>02967</td>
<td>7, 18, 31, 0, 0, 43, 43, 43, 43, 43, 43, 43, 43,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>02968</td>
<td>19, 46, 31, 0, 51, 32, 49, 31, 1, 13, 2, 15, 31,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>02969</td>
<td>30, 43, 31, 0, 51, 32, 49, 31, 1, 13, 2, 15, 31,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>02970</td>
<td>51, 92, 54, 2, 70, 51, 90, 31, 2, 77, 22, 66, 27,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>02971</td>
<td>45, 45, 30, 45, 46, 10, 10, 0, 27, 22, 22, 29, 27,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>02972</td>
<td>38, 41, 16, 1, 52, 41, 30, 58, 3, 11, 27, -122, 2,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>02973</td>
<td>19, 2, -36, 29, -113, 24, 1, 24, -72, 2, 90, 1,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>02974</td>
<td>94, -96, 13, 1, 13, -96, 49, 94, 32, 7, 1, 7, 10,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>02975</td>
<td>16, -12, 93, -28, -12, -19, 93, 16, -12, 93, 93,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>02976</td>
<td>125, 16, -60, 7, 14, 60, -121, 14, 16, -60, 7, 1,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>02977</td>
<td>93, 93, 93, 93, 93, 93, 93, 93, 93, 93, 93, 93, 93,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>02978</td>
<td>6, 6, 35, 34, 39, 38, 53, 52, 54, 55, 38, 38, 53, 2,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>02979</td>
<td>6, 7, 22, 23, 7, 38, 1, 54, 54, 53, 52, 38, 35, 52,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>02980</td>
<td>22, 51, 50, 54, 3, -51, 89, -46, 122, -31, -124, 2,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>02981</td>
<td>25, -69, 48, 82, 101, -128, 121, 109, -2, 30, 12,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>02982</td>
<td>-126, 81, -83, -83, 99, 99, -104, 124, -103, -124, 1,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>02983</td>
<td>40, -64, 124, -122, 91, -113, 70, 3, -123, 69, 1,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>02984</td>
<td>73, 89, -64, 101, 0, 1, 0, 90, 3, -77, 1, 39, 5, 1,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>02985</td>
<td>-18, 2, 0, 3, -127, 32, 1, -112, 1, 2, 1, 106, 6, 1,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Row</td>
<td>Values</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>--------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>03032</td>
<td>57, -105, 3, 1, 2, 3, -97, 3, -81, 3, 2, 3, 118,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>03033</td>
<td>3, -24, 0, 2, -24, 1, -72, 1, -87, -75, 0, 0, 4,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>03034</td>
<td>63, 60, 63, 60, -121, 5, 46, 43, 93, 125, 16, -6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>03035</td>
<td>-2, 88, 25, 5, -20, -6, 20, 0, 0, 2, 0, 85, -1,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>03036</td>
<td>2, 2, 67, 84, 88, 64, 10, 26, 30, 4, 5, 20, 30, 10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>03037</td>
<td>9, -72, -1, -24, 64, 25, 13, 13, 2, 85, 9, 17, 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>03038</td>
<td>12, 13, 13, 2, 85, 0, 47, 43, 43, 43, -51, 47, 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>03039</td>
<td>2, 67, 84, 88, 64, 10, 26, 30, 4, 5, 20, 30, 13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>03040</td>
<td>-72, -1, -26, -76, 13, 13, 6, 85, 9, -72, -1, -3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>03041</td>
<td>85, 0, 16, 12, 12, 6, 85, 0, 16, 11, 11, 6, 85,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>03042</td>
<td>63, -19, 63, -19, 49, 48, 27, -76, 6, 32, 25, 16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>03043</td>
<td>-1, -32, 0, 18, -1, -32, 0, 15, -1, -32, 64, 98,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>03044</td>
<td>11, 24, 2, 69, 19, 76, 21, 74, 25, 67, 27, 84, 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>03045</td>
<td>108, 21, 107, 25, 96, 27, 121, 2, 119, 6, 118, 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>03046</td>
<td>6, -42, 11, -37, 15, 26, 26, 30, 4, 5, 20, 30, 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>03047</td>
<td>9, 16, 9, 2, 9, -112, 31, 17, 115, 0, -72, -1, 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>03048</td>
<td>30, -57, -117, 24, 43, 16, -10, 93, 43, -19, 16, 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>03049</td>
<td>113, 0, 93, 0, 56, 56, 56, 56, 56, 1, 56, 56, 56,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>03050</td>
<td>16, 2, 6, 35, 34, 39, 38, 19, 16, 22, 51, 50, 54</td>
<td></td>
<td></td>
</tr>
<tr>
<td>03051</td>
<td>-78, 116, 66, 106, -45, -95, -44, 121, -111, -75,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>03052</td>
<td>-84, 95, -77, -2, -1, -38, -2, -2, -2, -61, -83,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>03053</td>
<td>-122, 0, 0, 1, 0, -33, 0, 0, 2, -5, 5, -64, 0, 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>03054</td>
<td>127, 2, -113, 2, -103, 8, 4, -84, 4, 1, 9, 0, 6, 5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>03055</td>
<td>-72, -1, -64, 64, 10, 33, 35, 52, 48, 0, 1, 32, 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>03056</td>
<td>-1, -22, 64, 17, 15, 15, 2, 85, 0, 28, 12, 12, 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>03057</td>
<td>25, 15, 15, 6, 85, 0, 16, 12, 12, 6, 85, 0, 16, 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>03058</td>
<td>5, -72, -1, -64, 64, 14, 33, 35, 52, 48, 5, 1, 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>03059</td>
<td>0, 24, 43, 78, 16, -28, 93, 113, 43, 43, 16, -103</td>
<td></td>
<td></td>
</tr>
<tr>
<td>03060</td>
<td>60, 0, 63, 63, 23, 57, 1, 17, 57, 49, 48, 1, 93,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>03061</td>
<td>55, 51, 2, -5, -76, 65, -45, 84, -105, -30, 47, 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>03062</td>
<td>60, 0, 0, 4, 7, 5, -64, 0, 30, 1, -57, -79, 6, 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>03063</td>
<td>6, 85, 13, -72, -1, -12, -76, 17, 17, 6, 85, 13, 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>03064</td>
<td>30, -72, -1, -24, 64, 23, 19, 19, 6, 85, 30, 30, 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>03065</td>
<td>13, 6, 85, 30, -72, 2, -69, 64, 12, 2, 10, 23, 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>03066</td>
<td>-44, -51, 17, 18, 57, 47, -51, 0, 47, -19, 43, 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>03067</td>
<td>2, 2, 67, 84, 88, 64, 9, 17, 16, 13, 12, 18, 18, 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>03068</td>
<td>13, 30, 20, 5, 30, -72, -1, -32, 64, 11, 18, 19, 1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
03439 48, 0, 75, -80, 11, 83, 75, -80, 30, 81, 90, 88, 0
03440 56, 56, 56, 89, 1, 75, -80, 12, 83, 75, -80, 40, 0
03441 89, 1, 67, 92, 88, -71, 0, 13, -1, -44, -74, 33, 0
03442 50, 55, 57, 13, -72, -1, -44, -75, 45, 57, 14, 0
03443 113, 93, 1, 93, 89, 89, 51, 17, 33, 1, 22, 23, 0
03444 36, 1, 91, 48, 22, 25, 53, 1, 95, 1, 5, -69, -2, 0
03445 3, -4, -6, 70, 4, -53, -5, 53, 4, -32, -5, 32, 0
03446 125, -79, 18, 11, -72, -1, -44, -74, 33, 57, 14, 0
03447 22, 2, 85, 8, 2, 3, 3, 32, 7, 8, 20, 7, 7, 8, 2, 0
03448 8, 4, 3, 32, 6, 6, 5, -72, -1, -20, -76, 15, 15, 0
03449 5, 18, 12, 12, 2, 85, 5, -72, -1, -9, 64, 26, 1
03450 96, 11, 112, 11, -128, 11, 5, 11, 8, 9, 32, 1, 0
03451 32, 16, 16, 2, 85, 0, -72, -1, -10, -76, 15, 15, 0
03452 -1, -6, 64, 11, 12, 12, 2, 85, 0, 4, 11, 11, 6, 0
03453 -1, -8, 64, 10, 13, 13, 6, 85, 0, 93, 10, 59, 89, 0
03454 43, 60, -3, 60, 16, 93, 113, -12, 43, 43, 43, 43, 0
03455 17, 57, 57, -121, 4, 46, 43, -121, 125, -60, -79, 0
03456 8, 32, 12, 17, 52, 0, 43, 43, 89, 49, 48, 43, 0
03457 -1, -64, -74, 70, 57, 8, 64, 50, 57, 3, -72, -1, 0
03458 25, 57, 7, 34, 50, 57, 2, -72, -1, -34, -74, 50, 0
03459 7, 14, 20, 57, 7, 14, 19, 57, 2, -72, -1, -12, -74, 0
03460 57, 7, 14, 21, 57, 2, -72, -1, -8, -79, 21, 57, 0
03461 43, 0, 43, 43, 43, 43, 43, 89, 51, 17, 51, 1, 17, 0
03462 5, -70, -5, -127, 4, 127, -6, 70, 4, -128, -5, -79, 0
03463 0, 27, 0, -54, 64, 80, 26, 15, 1, 20, 16, 20, 26, 0
03464 27, 4, -87, 23, -74, 14, -58, 14, 3, 23, 23, 24, 0
03465 5, 88, 9, 87, 16, 85, 17, 95, 19, 90, 23, 95, 2
03466 3, 18, 30, 11, 9, 21, 38, 32, 7, 1, 7, -72, -1, 0
03467 13, 2, 85, 7, -72, -1, -16, -76, 12, 12, 2, 85, 0
03468 -76, 13, 13, 6, 85, 7, -72, -1, -6, 64, 33, 12, 0
03469 1, 0, 6, 11, 11, 6, 85, 0, 6, 12, 12, 6, 85, 0, 0
03470 93, -19, 78, 16, 93, -10, 43, 43, 43, 43, 43, 43, 0
03471 113, 0, 93, 93, 93, 113, 19, 16, 0, 33, 50, 4, 0
03472 50, 0, 17, 52, 2, 38, 35, 34, 0, 99, 1, -120, 1, 0
03473 -88, -56, 1, 29, -41, -37, 1, 27, 121, -23, -113, 0
03474 -2, -96, -75, -56, 1, 90, -66, -2, -9, -2, -49, -76, 0
03475 0, 4, -3, 5, -70, 0, 13, 0, 24, 0, -78, 64, 2
| 03661 | 66, 94, 1, 28, -30, -3, -73, 2, 109, 3, 77, -2, |
| 03662 | 1, 0, 41, 0, 0, 4, -80, 5, -70, 0, 12, 1, 12, -7, |
| 03663 | 1, 71, 8, 72, 9, 3, 10, 8, 11, 9, 2, -79, 6, 2, |
| 03664 | 30, 12, 8, 8, 5, 30, 6, 2, 0, 63, -3, 60, 63, -3, |
| 03665 | 1, 3, 2, 1, 1, 4, 9, 10, 4, 8, 10, 10, 38, 29, 3, |
| 03666 | 4, 20, 1, 1, 4, 10, 40, 11, 28, 52, 1, 40, 11, 2, |
| 03667 | -1, -40, 64, 19, 11, 28, 52, 1, 10, 4, 8, 5, 30, |
| 03668 | -75, 0, 1, 0, 4, 1, -75, 64, 27, 0, 7, 48, 8, 6, |
| 03669 | 1, 0, 5, 6, 81, 0, 25, 13, -74, -103, 24, 43, 78, |
| 03670 | 77, -12, 113, 60, 16, -28, 16, -4, 0, 63, 60, -3, |
| 03671 | -121, 5, 46, 43, 93, 113, 43, -121, 125, -60, 1, |
| 03672 | 93, 43, 67, 92, 88, 64, 9, 2, 34, 33, 57, 1, 24, |
| 03673 | 57, 43, 43, 43, 43, 89, 51, 53, 1, 54, 55, 33, 5, |
| 03674 | -50, 4, 26, -4, -55, 89, 3, -88, -76, 3, -85, 10, |
| 03675 | 2, 24, 5, -70, 0, 7, 0, 70, 64, 43, 4, 3, 43, 1, |
| 03676 | 7, 6, -82, 4, 5, 37, 1, 0, 6, 12, 12, 2, 85, 0, |
| 03677 | -99, 104, 24, 43, 16, -10, 93, 43, 43, 60, -3, 6, |
| 03678 | 60, 49, 48, 19, 17, 33, 21, 35, 17, 51, 21, -117, |
| 03679 | 0, 1, 0, 0, -1, -25, 2, 57, 5, -45, 0, 3, 0, 76, |
| 03680 | -104, 0, 1, 1, 0, -112, 0, -96, 0, 2, 0, 118, 3, |
| 03681 | -24, 0, -72, 1, -87, -73, 2, -24, 1, 1, 4, -77, |
| 03682 | 60, -121, 5, 46, 43, 93, 125, 16, -60, 49, 48, |
| 03683 | -88, 25, 5, -20, -6, 20, 0, 1, 0, 39, -2, 105, |
| 03684 | 7, 6, 16, 3, 2, 43, 0, 1, 18, 6, 5, 5, 1, 2, |
| 03685 | 12, 12, 2, 85, 0, -84, 9, -101, 90, 24, 43, 16, |
| 03686 | -3, 60, 63, 60, -3, 60, 49, 48, 1, 33, 53, 51, |
| 03687 | 105, -107, 6, 39, -107, 0, 0, 1, 0, 54, 2, -78, |
| 03688 | 64, 21, 20, 57, 0, 64, 20, 57, 38, 2, 41, 3, 2, |
| 03689 | 101, 64, 23, 2, 5, 60, 4, 0, 60, 1, 6, 6, 3, 2, |
| 03690 | 8, -68, 1, 50, 0, 33, 1, -65, 1, -127, 0, 24, 43, |
| 03691 | 16, -19, 16, -19, 0, 47, -19, -19, 16, 60, 49, 4, |
| 03692 | -71, 1, 97, -111, 1, 99, -75, -9, 2, -78, 3, 33, |
| 03693 | -2, -21, 0, 3, 0, 26, 64, 12, 1, 63, 0, 2, 26, 5, |
| 03694 | -26, 0, 47, 77, -19, 49, 48, 3, 53, 33, 21, 31, |
| 03695 | 1, -47, 5, -62, 0, 3, 0, 96, 64, 11, 3, 56, 23, |
| 03696 | 23, 25, 52, 3, -72, -1, -64, 64, 26, 22, 25, 52, |
| 03697 | 0, 1, 16, 1, 2, 1, -121, 2, 0, 0, -72, 2, 83, -7, |
04549 113, 1, -113, -4, 56, 3, -56, 1, 4, 1, -109, -89
04550 77, 0, 106, 4, 24, 5, 60, 0, 6, 0, 10, 0, 118, 6
04551 0, 6, 5, 3, 3, 12, 2, 8, 7, 37, 9, 10, -67, 2
04552 -78, 2, 64, 6, -70, 1, 80, 0, 2, 1, 80, 64, 26
04553 4, 60, 1, 48, 0, -96, 0, 2, 0, 25, 11, 113, -116
04554 -19, 0, 25, 47, 26, -19, -19, -19, -19, 24, 26
04555 23, 57, 49, 48, 0, 93, 19, 53, 1, 21, 1, 1, 21
04556 -4, 56, 3, -56, 2, -6, -88, 1, -102, -76, -2, -59
04557 5, 60, 0, 6, 0, 10, 0, -118, 64, 24, -128, 2, -113
04558 11, 5, 10, 9, 7, 8, 37, 64, 9, -72, 2, -84, -73
04559 64, 0, 1, 1, 80, -78, 3, 32, 4, -69, 2, 90, 0, 6
04560 3, 60, 6, 5, -72, 1, 34, 64, 11, 31, 0, 48, 0, 2
04561 77, -19, 60, -19, -12, 60, 0, 25, 47, 26, -3, 2
04562 -19, 24, 16, -10, 26, -3, 60, 16, 60, 1, 17, 18
04563 1, 53, 1, 3, 33, 53, 33, 4, 24, -4, 53, 3, 1
04564 97, -77, 1, 63, 1, 59, -76, -2, 102, -4, -56, -8
04565 0, -23, 64, 55, 36, 8, 36, 11, 43, 15, 43, 18
04566 2, 18, 17, 17, 21, 8, 9, 9, 5, 12, 11, 10, 10
04567 -24, 22, 22, 4, 21, 5, 1, -24, 4, -72, 2, -81
04568 32, 17, 17, 16, 16, 10, 10, 9, 0, 0, 10, 24, 23
04569 56, 9, 16, 60, 32, 17, 1, 17, -72, 1, 0, 64, 11
04570 1, 0, 64, 15, 5, 5, 0, 16, 15, 15, 6, 85, 0, 16
04571 122, 24, 43, 16, -10, 43, 43, 60, 16, -12, 93
04572 60, 16, -12, 60, 16, 60, 16, 60, 0, 63, 63, 60
04573 60, 16, 60, 16, -19, 17, 18, 57, 1, 17, 57, -12
04574 -60, -121, 16, -60, 49, 48, 0, 93, 1, 93, 33, 17
04575 55, 1, 51, 1, 33, 21, 33, 21, 33, 21, 33, 17, 1
04576 1, 34, 49, 27, 23, 59, 1, 18, -42, -2, 107, 1, 8
04577 -57, -3, -4, 88, 66, 53, 110, 1, -5, -3, 57, -10
04578 38, 0, 25, 1, 86, 64, 61, 40, 4, 40, 5, 40, 22
04579 89, 4, 91, 9, 106, 4, 106, 9, 123, 4, 123, 10
04580 22, 25, 15, 6, 2, 10, 20, 28, 7, 11, 13, 14, 2
04581 85, 27, -72, -1, -10, -76, 13, 13, 2, 85, 0, -72
04582 13, 2, 85, 0, -72, -1, -2, -76, 12, 12, 2, 85, 0
04583 64, 11, 16, 16, 6, 85, 0, 27, 14, 15, 6, 85, 0
04584 -76, 12, 12, 6, 85, 0, -72, -1, -21, 64, 28, 11
04585 27, 2, -48, 27, -32, 27, 2, 27, 15, 12, 37, 13
05030  43,  44,  34,  60,  62,  33,  45,  54,  6,  9,  -7,  -9,  4,
05031  60,  -101,  -104,  63,  47,  46,  63,  -4,  114,  -57,  -7,
05032  46,  62,  -108,  -57,  -75,  -74,  -62,  -60,  -58,  -71,
05033  0,  0,  5,  89,  7,  44,  2,  38,  0,  36,  0,  0,  1,  7,  0,
05034  111,  17,  -97,  17,  2,  0,  17,  20,  1,  2,  65,  2,  1,
05035  43,  114,  53,  0,  -1,  -1,  0,  -94,  0,  0,  4,  -24,  7,
05036  107,  1,  106,  0,  42,  64,  18,  1,  12,  64,  30,  32,  8,
05037  -72,  -3,  -1,  -76,  72,  43,  1,  1,  18,  -71,  2,  33,
05038  -3,  0,  0,  5,  89,  7,  44,  2,  38,  0,  36,  0,  0,  1,  7,
05039  18,  -72,  -1,  -64,  64,  11,  18,  25,  52,  18,  12,  0,
05040  0,  43,  1,  43,  43,  53,  0,  -1,  -1,  0,  -94,  0,  0,  4,
05041  0,  -114,  1,  108,  1,  30,  0,  71,  -78,  2,  1,  14,  -7,
05042  52,  14,  -72,  -1,  -64,  64,  20,  29,  32,  52,  14,  6,
05043  14,  4,  -72,  1,  14,  -75,  72,  43,  1,  2,  19,  -71,
05044  43,  43,  53,  53,  0,  -1,  -1,  0,  -94,  0,  0,  4,  -24,
05045  1,  -127,  1,  106,  0,  40,  64,  16,  1,  -97,  13,  -81,
05046  -3,  -5,  -76,  72,  43,  1,  1,  13,  -71,  2,  33,  0,  4,
05047  0,  0,  1,  -2,  7,  44,  2,  38,  0,  44,  0,  0,  1,  7,  0,
05048  -72,  -1,  -64,  -77,  23,  25,  52,  7,  -72,  -1,  -64,
05049  1,  7,  -71,  2,  33,  0,  41,  0,  43,  1,  43,  93,  43,
05050  44,  2,  38,  0,  44,  0,  0,  1,  7,  0,  -42,  -1,  -57,
05051  0,  41,  0,  43,  1,  -79,  6,  2,  67,  84,  88,  -75,  0,
05052  52,  52,  4,  64,  29,  31,  52,  4,  1,  97,  72,  43,  43,
05053  6,  -31,  2,  38,  0,  44,  0,  0,  1,  7,  0,  -114,  -1,
05054  72,  43,  1,  2,  2,  11,  -71,  2,  33,  0,  41,  0,  43,
05055  7,  44,  2,  38,  0,  44,  0,  0,  1,  7,  0,  67,  -1,  -35,
05056  33,  0,  41,  0,  43,  1,  -79,  6,  2,  67,  84,  88,  -75,
05057  23,  25,  52,  5,  64,  34,  37,  52,  32,  5,  1,  5,  2,
05058  0,  -1,  -1,  0,  99,  -1,  -25,  5,  -35,  7,  44,  2,  38,
05059  0,  36,  -79,  2,  31,  -72,  -1,  -64,  64,  16,  22,  25,
05060  31,  -71,  2,  33,  0,  41,  0,  43,  1,  43,  113,  43,  53,
05061  0,  50,  0,  0,  1,  7,  0,  -42,  1,  -58,  1,  106,  0,  22,
05062  34,  -71,  2,  33,  0,  41,  0,  43,  1,  43,  53,  -1,  -1,
05063  0,  0,  1,  7,  0,  67,  1,  -61,  1,  106,  0,  36,  -79,  -25,
05064  -17,  29,  2,  29,  3,  0,  72,  43,  2,  1,  29,  -71,  2,
05065  0,  -95,  -1,  -25,  5,  34,  7,  44,  2,  38,  0,  56,  0,
05066  27,  1,  24,  64,  12,  14,  52,  79,  24,  1,  31,  24,  4
<table>
<thead>
<tr>
<th>Row</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>06251</td>
<td>22, 51, 50, 54, 53, 17, 52, 38, 35, 34, 6, 21,</td>
</tr>
<tr>
<td>06252</td>
<td>-106, 73, 68, 34, 53, 32, 65, 44, 104, 119, 117,</td>
</tr>
<tr>
<td>06253</td>
<td>19, -103, 14, 88, -125, 2, -68, -108, -120, -106,</td>
</tr>
<tr>
<td>06254</td>
<td>-25, 5, -35, 6, -45, 2, 38, 0, 50, 0, 0, 1, 7, 6,</td>
</tr>
<tr>
<td>06255</td>
<td>112, 28, 2, -72, -1, -20, -73, 28, 31, 0, 7, 65,</td>
</tr>
<tr>
<td>06256</td>
<td>53, -1, -1, 0, 68, -1, -24, 4, 39, 5, 95, 2, 38,</td>
</tr>
<tr>
<td>06257</td>
<td>0, 37, -77, 2, 2, 1, 26, -71, 2, -61, 0, 41, 0,</td>
</tr>
<tr>
<td>06258</td>
<td>0, 7, 65, 43, 27, -76, 26, 2, 10, 72, 43, 43, 89,</td>
</tr>
<tr>
<td>06259</td>
<td>2, 38, 0, 50, 0, 0, 1, 7, 0, -39, 1, -37, 1, 106,</td>
</tr>
<tr>
<td>06260</td>
<td>-128, 35, -112, 35, 5, 35, 2, 0, 72, 43, 2, 1, 3,</td>
</tr>
<tr>
<td>06261</td>
<td>-1, -1, 0, 68, -1, -24, 4, 39, 5, -72, 2, 38, 0,</td>
</tr>
<tr>
<td>06262</td>
<td>22, 64, 10, 2, 0, 33, 27, 0, 7, 65, 2, 1, 30, -7,</td>
</tr>
<tr>
<td>06263</td>
<td>0, -95, -2, 91, 5, -83, 5, -70, 2, 38, 0, 53, 0,</td>
</tr>
<tr>
<td>06264</td>
<td>11, 2, 32, 46, 1, 0, 46, 40, 1, 6, 65, 37, 0, 47,</td>
</tr>
<tr>
<td>06265</td>
<td>2, -58, 4, 62, 2, 38, 0, 85, 0, 0, 1, 6, 1, -18,</td>
</tr>
<tr>
<td>06266</td>
<td>92, -1, -25, 4, -21, 7, 38, 2, 38, 0, 54, 0, 0,</td>
</tr>
<tr>
<td>06267</td>
<td>1, 0, 51, 54, 22, 22, 65, 1, 1, 50, -71, 2, 33,</td>
</tr>
<tr>
<td>06268</td>
<td>-24, 3, -79, 5, -62, 2, 38, 0, 86, 0, 0, 1, 7, 6,</td>
</tr>
<tr>
<td>06269</td>
<td>51, 54, 21, 21, 65, 1, 1, 50, -71, 2, 34, 0, 41,</td>
</tr>
<tr>
<td>06270</td>
<td>-70, 5, -70, 0, 15, 0, -76, 64, 38, 0, 17, 16,</td>
</tr>
<tr>
<td>06271</td>
<td>5, 30, 7, 6, 2, 15, 8, 11, 12, 57, 7, 1, 2, 57,</td>
</tr>
<tr>
<td>06272</td>
<td>-73, 9, 32, 4, 5, 47, 6, 1, 6, -72, 1, 1, -78, 4,</td>
</tr>
<tr>
<td>06273</td>
<td>15, 8, 15, 15, 2, 85, 15, -72, -1, -14, -76, 12,</td>
</tr>
<tr>
<td>06274</td>
<td>-72, -1, -4, -76, 12, 12, 6, 85, 15, -72, -1, -2,</td>
</tr>
<tr>
<td>06275</td>
<td>6, 85, 16, 15, 32, 15, 2, 15, -72, 2, 115, -77,</td>
</tr>
<tr>
<td>06276</td>
<td>43, 43, 60, 16, -12, 93, 60, 16, -3, -12, 93, 60,</td>
</tr>
<tr>
<td>06277</td>
<td>60, 17, 18, 57, 47, 60, -3, 60, 49, 48, 1, 93,</td>
</tr>
<tr>
<td>06278</td>
<td>33, 17, 35, 2, 19, -2, -74, 1, 74, -2, 29, 4,</td>
</tr>
<tr>
<td>06279</td>
<td>-83, -83, -3, -20, -124, -3, -117, 0, 0, 1, 0,</td>
</tr>
<tr>
<td>06280</td>
<td>-1, -64, -77, 35, 38, 52, 6, -72, -1, -64, 64, 9,</td>
</tr>
<tr>
<td>06281</td>
<td>2, 26, 12, 5, 43, 11, 6, 6, 22, -55, 26, 3, 24,</td>
</tr>
<tr>
<td>06282</td>
<td>11, 12, 6, 85, 8, 9, 6, 17, 14, 13, 10, 4, 9, 18,</td>
</tr>
<tr>
<td>06283</td>
<td>2, -80, 2, -64, 2, -48, 2, 4, 2, 2, 6, 11, 12, 3,</td>
</tr>
<tr>
<td>06284</td>
<td>5, 69, 30, -72, -1, -6, -76, 16, 16, 2, 85, 30,</td>
</tr>
<tr>
<td>06285</td>
<td>2, 85, 30, 8, 11, 11, 2, 85, 30, 6, 16, 16, 6, 8,</td>
</tr>
<tr>
<td>06286</td>
<td>-1, -4, 64, 11, 11, 11, 6, 85, 30, 18, 12, 12, 6,</td>
</tr>
<tr>
<td>06287</td>
<td>30, -65, 30, 2, 0, 30, -48, 30, 2, 30, 78, 31, 2,</td>
</tr>
</tbody>
</table>
07065  -127, -32, -59, -2, 126, 1, 123, -112, -97, 90,  
07066  44, 2, 38, 2, 68, 0, 0, 1, 7, 0, -115, 1, 47, 1,  
07067  41, 1, 100, -123, 0, 43, -1, -1, 0, 10, -1, -20,  
07068  -39, 1, 100, 1, 95, 0, 22, 64, 10, 1, 0, 24, 18,  
07069  43, 1, 43, 53, 0, 1, 0, -96, -2, 105, 5, 33, 5,  
07070  32, 13, -128, 13, -32, 13, 4, 9, 6, 2, 2, 7, 30,  
07071  15, 15, 2, 85, 11, 16, 12, 12, 2, 85, 11, -72, -  
07072  85, 11, 10, 13, 13, 6, 85, 11, 25, 15, 15, 6, 85,  
07073  11, -96, 11, -64, 11, 6, 32, 11, 96, 11, -64, 11,  
07074  36, 16, 16, 2, 85, 4, -72, -1, -25, -76, 15, 15,  
07075  -1, -4, 64, 25, 12, 12, 2, 85, 4, 16, 11, 11, 2,  
07076  95, 4, 1, 4, -72, 2, 20, 64, 15, 1, 6, 13, 13, 2,  
07077  -1, -14, -76, 13, 13, 2, 85, 2, -72, -1, -16, -76,  
07078  2, -72, -1, -6, -76, 12, 12, 6, 85, 2, -72, -1,  
07079  15, 6, 85, 0, 2, 80, 2, -96, 2, -80, 2, -16, 2,  
07080  43, 43, 43, 43, 43, 43, 43, -3, 43, -3, 93, 113,  
07081  43, 43, 43, 43, 43, 43, -19, 0, 63, 60, 16, -19, 47,  
07082  51, 17, 33, 17, 51, 17, 3, 55, -83, -2, 22, -62,  
07083  -6, 70, -1, -1, -1, -3, 0, 0, 5, 89, 5, -70, 2,  
07084  -8, 5, -70, 0, 14, 0, 24, 0, -28, 64, 21, 40, 8,  
07085  8, 2, 2, 0, 19, 38, 9, -72, -1, -15, -76, 11, 11,  
07086  9, 4, 16, 16, 6, 85, 9, -72, -1, -64, 64, 19, 36,  
07087  9, 49, -33, 26, 1, 26, -72, -1, -64, 64, 17, 30,  
07088  2, 85, 0, -72, -1, -10, -76, 15, 15, 2, 85, 0,  
07089  12, 12, 2, 85, 0, -72, -1, -10, -76, 12, 12, 6,  
07090  -10, -74, 15, 16, 6, 85, 0, 93, 25, 16, -10, 43,  
07091  93, 93, 43, 43, 43, 43, 43, -19, 18, 57, 47, 0, 63,  
07092  6, 22, 11, 38, 7, 37, 21, 38, 17, 12, 19, 54, 1,  
07093  23, 54, 1, 43, 43, 1, 43, 43, 43, 43, 43, -127,  
07094  35, 33, 55, 33, 50, 54, 53, 52, 38, 38, 35, 33,  
07095  -3, -62, -62, 1, -124, -99, -99, 90, -96, -63,  
07096  -128, 91, 122, 40, 0, -1, -1, 0, -106, 0, 0, 4,  
07097  0, 0, 4, 85, 5, -70, 0, 5, 0, 123, 64, 23, 2, 3,  
07098  32, 5, 5, 0, 36, 16, 16, 2, 85, 0, -72, -1, -14,  
07099  2, 85, 0, -72, -1, -2, -76, 12, 12, 2, 85, 0, -76,  
07100  15, 15, 6, 85, 0, -72, -1, -23, -76, 13, 13, 6,  
07101  6, 59, -114, 24, 43, 78, 16, -12, 43, 43, 43, 43,
08249 0, 50, 0, 0, 4, 41, 4, 49, 2, 38, 2, -95, 0, 0, 0
08250 0, 29, 16, 29, 32, 29, 96, 29, 4, 0, 29, 22, 15, 2
08251 1, 43, 93, 53, -1, -1, 0, 25, 0, 0, 2, -24, 4, 2
08252 70, 0, 40, 64, 26, 1, 64, 36, -128, 36, 2, 32, 3
08253 17, 17, 65, 1, 1, 26, -71, 2, -32, 0, 41, 0, 43, 2
08254 -28, 4, 37, 2, 38, 2, -93, 0, 0, 1, 6, 2, -104, 4
08255 70, 0, 40, 64, 26, 1, 64, 36, -128, 36, 2, 32, 3
08256 17, 17, 65, 1, 1, 26, -71, 2, -32, 0, 41, 0, 43, 2
08257 64, 17, 2, 64, 29, 112, 29, 4, 0, 29, 22, 15, 8
08258 43, 1, 43, 93, 53, 0, 2, 0, 0, 0, 0, 1, -80, 4, 2
08259 -45, -73, 4, 2, 10, 3, 6, 3, 32, 0, -72, -1, -18, 3
08260 0, 64, 67, 68, 52, 0, 64, 61, 53, -97, 0, 1, 79, 3
08261 2, -45, 64, 9, 47, 15, 1, 15, 64, 16, 17, 52, 15
08262 -3, 0, 63, 63, 47, -19, 49, 48, 1, 17, 35, 17, 3
08263 54, 54, 1, -80, -62, -102, 22, 37, 25, 22, 38, 2
08264 38, 26, 24, 38, 22, 43, 24, 41, 23, 0, 0, 2, 1
08265 0, -114, 64, 10, 15, 36, 31, 36, 80, 1, 98, 1, 2
08266 0, 39, 17, -7, 18, 6, 16, 16, 0, 17, 8, 32, 9, 2
08267 -6, 64, 43, 11, 13, 6, 85, 14, 22, 15, 15, 6, 85
08268 14, 1, 15, 14, -49, 14, -33, 14, 3, 14, 19, 64, 3
08269 36, 18, -59, 24, -71, 2, -45, 0, 32, 47, -3, -26, 3
08270 47, 43, -19, 17, 18, 57, 47, 0, 63, -3, -28, 63, 3
08271 22, 21, 21, 35, 53, 52, 39, 38, 53, 52, 55, 33, 3
08272 38, 53, 52, 54, 3, 59, -103, -83, 9, 26, -64, 26
08273 24, 24, 37, 23, 48, 3, -98, -81, -109, 29, 84, 3
08274 38, 26, 24, 38, 22, 22, 38, 24, 35, 49, 0, -1, -1
08275 0, 0, 1, 7, 2, -104, 1, 20, 0, 0, 0, 22, 64, 10, 2
08276 2, -33, 0, 41, 0, 43, 1, 43, 53, 0, 2, 0, 0, 2, 2
08277 -71, 0, 10, 2, -45, 64, 12, 4, 4, 0, 2, 1, 2, 2
08278 16, 6, 85, 0, 10, 13, 15, 6, 85, 0, 40, 11, 12, 2
08279 0, 1, 79, 0, -1, 0, 2, 0, -69, 2, -56, 0, 18, 0
08280 17, 52, 0, 47, 43, 113, -19, 16, -12, 113, 114, 3
08281 49, 48, 1, 17, 35, 17, 7, 50, 22, 21, 20, 6, 35, 3
08282 49, 35, 31, 53, 22, 38, 4, 37, -3, -37, 2, 37, 3
08283 0, 40, -2, 104, 3, -126, 4, 49, 2, 38, 2, -86, 0, 3
08284 0, 25, 18, 14, 8, 65, 1, 1, 15, -71, 2, -31, 0, 0
08285 3, 86, 4, 55, 2, 38, 2, -85, 0, 0, 1, 6, 2, -104, 3
| 12874 | 3, 15, 12, 1, 12, 31, 22, 63, 22, 79, 22, 3, 22, 12875 | 26, 24, 10, 5, 40, 40, 38, 26, 5, 16, 10, 32, 12876 | 43, 35, 29, 21, 14, 1, -72, 1, 42, -123, 0, 47, 12877 | 16, -42, -51, 1, 25, 47, 50, 50, 50, 50, 42, -42, 12878 | -35, 93, -42, -35, -64, 17, 57, 17, 18, 57, 47, 12879 | 17, 57, 47, -35, 93, -42, -35, -64, 17, 57, 17, 12880 | 17, 19, 1, 1, 23, 17, 7, 33, 39, 17, 55, 7, 17, 12881 | 33, 17, 39, 55, 23, 39, 7, 23, 17, 51, 17, 4, -12882 | -2, 26, -46, -96, 3, 105, -96, 104, -106, -2, 52, -2, 12883 | -2, -5, 88, -38, -90, -90, -91, 74, -74, 1, 5, 2, 12884 | -2, -5, 88, -38, -90, -90, -91, 74, -74, 1, 5, 2, 12885 | -52, -116, -4, -14, 3, 14, -116, 97, -3, 24, 2, 12886 | 88, -2, -82, 0, -88, -1, -2, 1, 23, 6, 90, 0, 0, 12887 | 14, 12, 16, 52, 31, 1, 1, 16, 1, -122, 1, -65, -12888 | 4, 38, 0, -88, 5, 118, 0, 3, 7, 0, 120, 6, -12889 | -72, -1, -64, 64, 17, 34, 39, 52, 15, 0, 31, 0, 12890 | 64, 14, 22, 27, 52, 2, 0, 4, 2, 6, 4, 7, 5, 3, 12891 | 3, 95, 3, 111, 3, 3, 3, 7, 31, 1, 63, 1, 95, 1, 12892 | 93, 43, 0, 26, 24, 16, -51, 17, 23, 57, 1, 47, 12893 | 24, 16, -51, 17, 23, 57, 49, 48, 19, 7, 39, 55, 12894 | -50, -88, -88, -88, -88, 104, 104, 104, 0, -1, 12895 | -6, 107, 0, 20, 64, 10, 0, 9, 64, 67, 68, 52, -1, -1, 12896 | -1, 0, 62, -1, 108, 6, -110, 5, -56, 2, 54, 3, 12897 | 5, 47, 3, -24, 5, 60, 0, 63, 64, 36, 4, 3, 2, 0, 12898 | 65, 4, 3, 2, 16, 84, 47, 84, 96, 84, -128, 84, 12899 | 9, 10, 52, 72, 0, 47, 43, 43, 53, 47, 93, 53, 53, 12900 | 0, 62, -1, 108, 6, -110, 5, -56, 2, 54, 3, -79, 0, 12901 | 3, -24, 5, 60, 0, 63, 64, 36, 4, 3, 2, 0, 85, 79, 12902 | 3, 2, 16, 84, 47, 84, 96, 84, -128, 84, 4, 84, 12903 | 52, 72, 0, 47, 43, 43, 53, 47, 93, 53, 53, 53, 12904 | -95, 4, 63, 5, -56, 2, 54, 3, -77, 0, 0, 0, 55, 12905 | 5, 60, 0, 63, 64, 36, 4, 3, 2, 0, 74, 68, 26, 0, 12906 | 16, 73, 47, 73, 96, 73, -128, 73, 4, 73, 1, 61, 12907 | 0, 47, 43, 43, 53, 47, 93, 53, 53, 53, 1, 43, 53, 12908 | 63, 5, -56, 2, 54, 3, -77, 0, 0, 0, 55, 5, 45, 2, 12909 | 0, 63, 64, 36, 4, 3, 2, 0, 74, 68, 26, 0, 65, 1, 12910 | 47, 73, 96, 73, -128, 73, 4, 73, 1, 61, 64, 11, 12911 | -95, 4, 63, 5, -56, 2, 54, 3, -77, 0, 0, 0, 55, 12912 | 5, 60, 0, 63, 64, 36, 4, 3, 2, 0, 74, 68, 26, 0, 12913 | 16, 73, 47, 73, 96, 73, -128, 73, 4, 73, 1, 61, 12914 | 0, 47, 43, 43, 53, 47, 93, 53, 53, 53, 1, 43, 53, 12915 | 63, 5, -56, 2, 54, 3, -77, 0, 0, 0, 55, 5, 45, 2, 12916 | 0, 63, 64, 36, 4, 3, 2, 0, 74, 68, 26, 0, 65, 1, 12917 | 47, 73, 96, 73, -128, 73, 4, 73, 1, 61, 64, 11, 12918 | -95, 4, 63, 5, -56, 2, 54, 3, -77, 0, 0, 0, 55, 12919 | 5, 60, 0, 63, 64, 36, 4, 3, 2, 0, 74, 68, 26, 0, 12920 | 16, 73, 47, 73, 96, 73, -128, 73, 4, 73, 1, 61, 12921 | 0, 47, 43, 43, 53, 47, 93, 53, 53, 53, 1, 43, 53, 12922 | 63, 5, -56, 2, 54, 3, -77, 0, 0, 0, 55, 5, 45, 2, 12923 | 0, 63, 64, 36, 4, 3, 2, 0, 74, 68, 26, 0, 65, 1, 12924 | 47, 73, 96, 73, -128, 73, 4, 73, 1, 61, 64, 11,
<table>
<thead>
<tr>
<th>Number</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>12948</td>
<td>12, 12, 9, 5, -72, 2, -3, -76, 16, 18, 1, 18, 12, 12949</td>
</tr>
<tr>
<td>12950</td>
<td>63, -19, -3, -34, 113, -19, 16, -50, 50, 16, -34,</td>
</tr>
<tr>
<td>12951</td>
<td>-50, 43, 43, 1, 26, 24, 16, -35, -50, 95, 94, 95,</td>
</tr>
<tr>
<td>12952</td>
<td>17, 57, 47, 95, 94, 93, 49, 48, 93, 93, 93, 1, 3,</td>
</tr>
<tr>
<td>12953</td>
<td>39, 38, 39, 38, 53, 52, 54, 55, 54, 55, 38, 39,</td>
</tr>
<tr>
<td>12954</td>
<td>22, 23, 22, 23, 1, 17, 35, 17, 33, 17, 35, 17, 2,</td>
</tr>
<tr>
<td>12955</td>
<td>23, 36, 107, 111, 88, -79, 63, 15, 51, 52, 33, 3,</td>
</tr>
<tr>
<td>12956</td>
<td>-50, 43, 43, 1, 26, 24, 16, -35, -50, 95, 94, 95,</td>
</tr>
<tr>
<td>12957</td>
<td>-119, 33, 26, 18, 64, 12, 40, 20, 16, 39, 29, 75,</td>
</tr>
<tr>
<td>12958</td>
<td>-2, 103, 1, -103, 0, -1, -1, 0, 54, 1, 10, 2, 24,</td>
</tr>
<tr>
<td>12959</td>
<td>-2, -15, 1, -12, 3, -90, 0, 12, 0, 27, 0, 96, 64,</td>
</tr>
<tr>
<td>12960</td>
<td>106, 6, 2, 22, 23, 20, 13, 25, 25, 1, 0, 8, 8, 64,</td>
</tr>
<tr>
<td>12961</td>
<td>1, 7, -70, 2, -17, 0, 8, 3, 4, -77, 23, 16, 22,</td>
</tr>
<tr>
<td>12962</td>
<td>63, -19, -34, -52, 51, 63, -19, 1, 47, 16, -48,</td>
</tr>
<tr>
<td>12963</td>
<td>-50, 50, 49, 48, 93, 93, 93, 93, 1, 33, 53, 33,</td>
</tr>
<tr>
<td>12964</td>
<td>34, 39, 38, 53, 52, 55, 23, 6, 21, 20, 22, 1, 3,</td>
</tr>
<tr>
<td>12965</td>
<td>54, 38, 56, 33, 27, -116, 22, 94, -116, 1, 37,</td>
</tr>
<tr>
<td>12966</td>
<td>55, 47, 60, -111, 99, 35, 86, 56, 28, 45, 0, 0,</td>
</tr>
<tr>
<td>12967</td>
<td>0, -85, 64, 16, 11, 27, 1, 21, 13, 36, 29, 36,</td>
</tr>
<tr>
<td>12968</td>
<td>9, 17, 52, 41, 38, 0, 31, 16, 31, 2, 9, 3, 31,</td>
</tr>
<tr>
<td>12969</td>
<td>64, 16, 25, 5, 23, 23, 16, 0, 47, 16, 25, 30, 7,</td>
</tr>
<tr>
<td>12970</td>
<td>52, 0, 40, 16, 40, -32, 40, -16, 40, 4, 40, 34,</td>
</tr>
<tr>
<td>12971</td>
<td>0, 16, 0, 30, 2, -17, 0, 0, 0, 16, 1, 44, -123,</td>
</tr>
<tr>
<td>12972</td>
<td>-52, 93, 43, 0, 17, 51, 26, 24, 16, -50, 17, 57,</td>
</tr>
<tr>
<td>12973</td>
<td>-42, -51, 17, 57, 16, -44, -52, 95, 94, 93, -51,</td>
</tr>
<tr>
<td>12974</td>
<td>34, 7, 6, 7, 6, 35, 34, 39, 38, 39, 38, 55, 6, 3,</td>
</tr>
<tr>
<td>12975</td>
<td>55, 54, 51, 51, 3, 20, 6, 35, 34, 39, 38, 53, 55,</td>
</tr>
<tr>
<td>12976</td>
<td>19, 17, 10, 39, 31, 27, 5, 4, 8, 80, -101, 90, 9,</td>
</tr>
<tr>
<td>12977</td>
<td>38, 56, 33, 27, -116, 22, 94, -116, 1, 37, 36,</td>
</tr>
<tr>
<td>12978</td>
<td>-104, 113, 60, 77, 65, 95, -4, -31, 38, 50, 55,</td>
</tr>
<tr>
<td>12979</td>
<td>-1, 116, 1, -58, 0, -100, 0, 3, 0, 7, 0, 52, 64,</td>
</tr>
<tr>
<td>12980</td>
<td>4, 112, 5, 1, 5, -97, 7, 1, 7, 7, 2, 0, 3, 1, 0,</td>
</tr>
<tr>
<td>12981</td>
<td>57, 1, 47, 51, 47, -51, 57, 57, 16, -51, 57, 57,</td>
</tr>
<tr>
<td>12982</td>
<td>-112, 56, 67, 54, -112, 56, 56, 105, 44, 105, -1,</td>
</tr>
<tr>
<td>12983</td>
<td>0, 3, 0, 7, 0, 11, 0, -33, 64, 52, 1, 3, 0, 2, 2,</td>
</tr>
<tr>
<td>12984</td>
<td>0, 64, 0, 64, 72, 84, 52, 0, 64, 61, 69, 52, 0,</td>
</tr>
</tbody>
</table>
13096 113, 53, 53, 53, 1, 43, 93, 53, 53, 53, 0, -1, 
13097 0, 0, 1, 23, 6, 110, 2, 88, 7, 73, 0, 102, -75, 
13098 67, 48, 51, 65, 4, 3, 2, 95, 66, 111, 66, -32, 63, 
13099 -1, -64, -77, 101, -1, 52, 66, -72, -1, -64, -75, 
13100 -64, -77, 60, 61, 52, 66, -72, -1, -64, 64, 9, 73, 
13101 43, 43, 93, 113, 53, 53, 53, 1, 43, 93, 53, 53, 
13102 54, 3, -5, 0, 0, 1, 23, 6, 111, 1, -12, -1, -46, 
13103 15, 17, 52, 48, 79, 64, 79, 2, 126, 79, 79, 11, 74, 
13104 74, -2, -69, 4, -50, 5, -34, 2, 54, 3, -4, 0, 0, 1, 
13105 3, 2, -72, -1, -46, 64, 14, 63, 63, 24, 24, 65, 
13106 53, 53, 1, 43, 53, 53, 53, 0, 1, 0, 113, 1, 37, 
13107 25, 52, 32, -72, -1, -32, -77, 22, 33, 52, 18, 
13108 16, 52, 31, 9, 1, 3, 9, 9, 15, 23, 15, 29, 31, 2, 
13109 -32, -74, 9, 31, 52, 1, 0, 3, 3, -72, -1, -64, 64, 
13110 3, 15, 31, 33, 33, 7, -72, 2, -5, -78, 11, 64, 1, 
13111 3, 11, 22, 17, -72, -1, -64, -77, 22, 64, 52, 17, 
13112 17, 52, 17, 29, 23, -72, 2, -5, 64, 13, 64, 0, 0, 
13113 -32, -74, 9, 31, 52, 1, 0, 38, 29, 47, 16, -58, 74, 
13114 57, 43, 43, 43, 1, 24, 16, -58, 95, 94, 93, 43, 
13115 95, 94, 93, -48, -51, 43, 0, 25, 16, -60, 50, 43, 
13116 95, 93, 49, 48, 1, 43, 43, 43, 43, 1, 7, 38, 35, 
13117 34, 7, 18, 17, 20, 7, 7, 35, 2, 39, 38, 39, 38, 
13118 64, -100, 29, 1, 7, 7, 14, 12, 6, 11, 23, 37, 91, 
13119 -97, -56, 126, 40, 25, -107, 88, 5, 38, 16, 47, -97, 
13120 71, 77, 1, 62, 90, -79, -127, 115, -100, -73, -7, 
13121 0, 107, 64, 11, 13, 16, 14, 20, 52, 14, 16, 17, 
13122 1, 22, 4, 4, 64, 9, 12, 52, 4, 9, 25, -72, 2, -72, 1, 
13123 23, -72, -1, -64, -75, 12, 60, 52, 23, 9, 15, -5, 
13124 47, 16, -42, -19, 16, -44, -19, 18, 57, 43, 1, 9, 
13125 49, 48, 1, 95, 94, 93, 43, 43, 43, 43, 1, 20, 6, 21, 
13126 51, 20, 22, 21, 20, 7, 6, 7, 54, 51, 50, 23, 22, 18, 
13127 51, 26, 75, 30, 4, 48, 51, 82, 90, -120, -118, 24, 
13128 -86, 90, 1, 53, 30, 118, 30, -88, -52, -33, -115, -8, 
13129 2, 54, 3, -5, 0, 0, 1, 22, 5, 52, 0, 0, 0, 23, -76, 3, 
13130 92, 42, 42, 65, 1, 43, 53, 53, 0, 47, 53, 53, 0, 
13131 3, -4, 0, 0, 1, 22, 5, 52, 0, 0, 0, 23, -76, 3, 
13132 16, 65, 1, 43, 53, 53, 0, 47, 53, 53, 0, 0, 0, 0.
13725 4, -86, 4, -85, 4, -84, 4, -83, 4, -82, 4, -81, 4, -80, 4, -79, 4, -78, 4, -77, 4, -76, 4, -75, 4,
13726 -74, 4, -73, 4, -72, 4, -71, 4, -70, 4, -69, 4, -68, 4, -67, 4, -66, 4, -65, 4, -64, 4, -63, 4, -62,
13730 -25, 4, -24, 4, -23, 4, -22, 4, -21, 4, -20, 4, -19, 4, -18, 4, -17, 4, -16, 4, -15, 4, -14, 4,
13731 -13, 4, -12, 4, -11, 4, -10, 4, -9, 4, -8, 4, -7, 4, -6, 4, -5, 4, -4, 4, -3, 4, -2, 4,
13732 1, 5, 2, 5, 3, 5, 4, 5, 5, 5, 5, 6, 5, 7, 5, 8, 5,
13733 5, 14, 5, 15, 5, 16, 5, 17, 5, 18, 5, 19, 5, 20, 5,
13734 26, 5, 27, 5, 28, 5, 29, 5, 30, 5, 31, 5, 32, 5,
13735 5, 39, 5, 40, 5, 41, 5, 42, 5, 43, 5, 44, 5, 45, 5,
13736 51, 5, 52, 5, 53, 5, 54, 5, 55, 5, 56, 5, 57, 5,
13737 5, 64, 5, 65, 5, 66, 5, 67, 5, 68, 5, 69, 5, 70, 5,
13738 76, 5, 77, 5, 78, 5, 79, 5, 80, 5, 81, 5, 82, 5,
13739 5, 89, 5, 90, 5, 91, 5, 92, 5, 93, 5, 94, 5, 95, 5,
13740 101, 5, 102, 5, 103, 5, 104, 5, 105, 5, 106, 5,
13741 5, 114, 5, 115, 5, 116, 5, 117, 5, 118, 5, 119, 5,
13742 126, 5, 127, 5, -128, 5, -127, 5, -126, 5, -125, 5,
13743 5, -117, 5, -116, 5, -115, 5, -114, 5, -113, 5,
13744 -105, 5, -104, 5, -103, 5, -102, 5, -101, 5, -100, 5,
13745 5, -92, 5, -91, 5, -90, 5, -89, 5, -88, 5, -87, 5,
13746 -80, 5, -79, 5, -78, 5, -77, 5, -76, 5, -75, 5,
13747 5, -67, 5, -66, 5, -65, 5, -64, 5, -63, 5, -62, 5,
13748 -55, 5, -54, 5, -53, 5, -52, 5, -51, 5, -50, 5,
13749 5, -42, 5, -41, 5, -40, 5, -39, 5, -38, 5, -37, 5,
13750 -30, 5, -29, 5, -28, 5, -27, 5, -26, 5, -25, 5,
13751 5, -17, 5, -16, 5, -15, 5, -14, 5, -13, 5, -12, 5,
13752 -5, 5, -4, 5, -3, 5, -2, 5, -1, 6, 0, 6, 1, 6, 2,
13753 6, 8, 6, 9, 6, 10, 6, 11, 6, 12, 6, 13, 6, 14, 6,
13754 20, 6, 21, 6, 22, 6, 23, 6, 24, 6, 25, 6, 26, 6,
13755 6, 33, 6, 34, 6, 35, 6, 36, 6, 37, 6, 38, 6, 39, 6,
13756 45, 6, 46, 6, 47, 6, 48, 6, 49, 6, 50, 6, 51, 6,
13757 6, 58, 6, 59, 6, 60, 6, 61, 6, 62, 6, 63, 6, 64, 6,
13758 70, 6, 71, 6, 72, 6, 73, 6, 74, 6, 75, 6, 76, 6,
13759 6, 83, 6, 84, 6, 85, 6, 86, 6, 87, 6, 88, 6, 89, 6,
13760 95, 6, 96, 6, 97, 6, 98, 6, 99, 6, 100, 6, 101, 6,
13761 6, 108, 6, 109, 6, 110, 6, 111, 6, 112, 6, 113, 6,
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.

// 3. This notice may not be removed or altered.

#ifndef SFML_AUDIO_HPP
#define SFML_AUDIO_HPP

// Headers

#include <SFML/System.hpp>
#include <SFML/Audio/Listener.hpp>
#include <SFML/Audio/Music.hpp>
#include <SFML/Audio/Sound.hpp>
#include <SFML/Audio/SoundBuffer.hpp>
#include <SFML/Audio/SoundBufferRecorder.hpp>
#endif
#include <SFML/Audio/SoundRecorder.hpp>
#include <SFML/Audio/SoundStream.hpp>

#endif // SFML_AUDIO_HPP

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
  Documentation generated by doxygen 1.5.2 ::
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
including commercial applications, and to alter it and redistribute it freely,
subject to the following restrictions:

1. The origin of this software must not be misrepresented;
you must not claim that you wrote the original software.
If you use this software in a product, an acknowledgment
in the product documentation would be appreciated but is not required.

2. Altered source versions must be plainly marked as such,
and must not be misrepresented as being the original software.

3. This notice may not be removed or altered.

// Headers
#include <SFML/Audio/AudioDevice.hpp>
#include <SFML/Audio/AudioResource.hpp>
#include <SFML/Audio/Listener.hpp>
#include <algorithm>
#include <iostream>

namespace sf
{
namespace priv
AudioDevice* AudioDevice::ourInstance;

AudioDevice::AudioDevice() :
    myRefCount(0)
{
    // Create the device
    myDevice = alcOpenDevice(NULL);
    if (myDevice)
    {
        // Create the context
        myContext = alcCreateContext(myDevice, NULL);
        if (myContext)
        {
            // Set the context as the current one (we'll only need one)
            alcMakeContextCurrent(myContext);
            // Initialize the listener, located at the origin and looking along the Z axis
            Listener::SetPosition(0.f, 0.f, 0.f);
            Listener::SetTarget(0.f, 0.f, -1.f);
        }
        else
        {
            std::cerr << "Failed to create the audio context";
        }
    }
    else
    {
        std::cerr << "Failed to open the audio device";
    }
}
AudioDevice::~AudioDevice()
{
    // Destroy the context
    alcMakeContextCurrent(NULL);
    if (myContext)
    {
        alcDestroyContext(myContext);
    }
    // Destroy the device
    if (myDevice)
    {
        alcCloseDevice(myDevice);
    }
}

AudioDevice& AudioDevice::GetInstance()
{
    // Create the audio device if it doesn't exist
    if (!ourInstance)
    {
        ourInstance = new AudioDevice;
        return *ourInstance;
    }
}

void AudioDevice::AddReference()
{
    // Create the audio device if it doesn't exist
    if (!ourInstance)
    {
        ourInstance = new AudioDevice;
    }
    // Increase the references count
    ourInstance->myRefCount++;
}

void AudioDevice::RemoveReference()
{
    // Decrease the references count
ourInstance->myRefCount--;  
// Destroy the audio device if the references count reaches 0
if (ourInstance->myRefCount == 0) {
    delete ourInstance;
    ourInstance = NULL;
}

ALCdevice* AudioDevice::GetDevice() const {
    return myDevice;
}

ALenum AudioDevice::GetFormatFromChannelsCount(unsigned) {
    // Find the good format according to the number of channels
    switch (ChannelsCount) {
        case 1: return AL_FORMAT_MONO16;
        case 2: return AL_FORMAT_STEREO16;
        case 4: return alGetEnumValue("AL_FORMAT_"
        case 6: return alGetEnumValue("AL_FORMAT_"
        case 7: return alGetEnumValue("AL_FORMAT_"
        case 8: return alGetEnumValue("AL_FORMAT_"
    }
    return 0;
}

} // namespace priv
} // namespace sf
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)
// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.
// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:
// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an
//    acknowledgment in the product documentation would be appreciated but
//    is not required.
// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.
// 3. This notice may not be removed or altered.

#ifndef SFML_AUDIODEVICE_HPP
#define SFML_AUDIODEVICE_HPP

// Headers
#include <SFML/Audio/OpenAL.hpp>
#include <set>
#include <string>

namespace sf {
  // Headers
  #include <SFML/Audio/OpenAL.hpp>
  #include <set>
  #include <string>

  namespace sf {
    

}
class AudioResource;

namespace priv {

class AudioDevice {

public:

static AudioDevice& GetInstance();
static void AddReference();
static void RemoveReference();

ALCdevice* GetDevice() const;

ALenum GetFormatFromChannelsCount(unsigned int);

private:

AudioDevice();
~AudioDevice();

// Static member data
static AudioDevice* ourInstance;

// Member data
ALCdevice* myDevice;
ALCcontext* myContext;
unsigned int myRefCount;

};

} // namespace priv

} // namespace sf
#endif // SFML_AUDIODEVICE_HPP

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
   Documentation generated by doxygen 1.5.2 ::
Config.hpp

00001
00002 //
00003 // SFML - Simple and Fast Multimedia Library
00004 // Copyright (C) 2007-2009 Laurent Gomila (laurent.gomila@gmail.com)
00005 //
00006 // This software is provided 'as-is', without any
00007 // In no event will the authors be held liable for
00008 //
00009 // Permission is granted to anyone to use this software for
00010 // including commercial applications, and to alter it and
00011 // subject to the following restrictions:
00012 //
00013 // 1. The origin of this software must not be misrepresented;
00014 //    you must not claim that you wrote the original
00015 //    If you use this software in a product, and
00016 //    in the product documentation would be appreciated.
00017 //
00018 // 2. Altered source versions must be plainly marked as such,
00019 //    and must not be misrepresented as being the original
00020 //
00021 // 3. This notice may not be removed or altered
00022 //
00024
00025 #ifndef SFML_CONFIG_HPP
00026 #define SFML_CONFIG_HPP
00027
00029 // Identify the operating system
00031 #if defined(_WIN32) || defined(__WIN32__)
00032
00033    // Windows
00034    #define SFML_SYSTEM_WINDOWS
00035    #ifndef WIN32_LEAN_AND_MEAN
00036        #define WIN32_LEAN_AND_MEAN
00037    #endif

#ifndef NOMINMAX
    #define NOMINMAX
#endif

#if defined(linux) || defined(__linux)
    // Linux
    #define SFML_SYSTEM_LINUX
#elif defined(__APPLE__) || defined(MACOSX) || defined(macintosh) || defined(Macintosh)
    // MacOS
    #define SFML_SYSTEM_MACOS
#elif defined(__FreeBSD__) || defined(__FreeBSD_kernel__)
    // FreeBSD
    #define SFML_SYSTEM_FREEBSD
#else
    // Unsupported system
    #error This operating system is not supported by SFML library
#endif

#ifndef NDEBUG
    #define SFML_DEBUG
#endif

#if defined(SFML_SYSTEM_WINDOWS)
    // Define portable debug macro
    #if !defined(NDEBUG)
        #define SFML_DEBUG
    #endif
#endif

// Define portable import / export macros
#if defined(SFML_SYSTEM_WINDOWS)
    // Define a portable debug macro
    #if !defined(NDEBUG)
        #define SFML_DEBUG
    #endif
#endif
#ifdef SFML_DYNAMIC

// Windows platforms
#ifdef SFML_EXPORTS

  // From DLL side, we must export
  #define SFML_API __declspec(dllexport)
#else

  // From client application side, we must import
  #define SFML_API __declspec(dllimport)
#endif

#endif

// For Visual C++ compilers, we also need
// You can read lots of different things
// just work fine, and so the simplest way to
// disable this warning is to
define _MSC_VER

#pragma warning(disable : 4251)
#endif
#else

// No specific directive needed for static build
#define SFML_API
#endif
#else

// Other platforms don't need to define anything
#define SFML_API
#endif
// Define portable fixed-size types
#include <climits>
namespace sf {
    // 8 bits integer types
    #if UCHAR_MAX == 0xFF
        typedef signed char Int8;
        typedef unsigned char Uint8;
    #else
        #error No 8 bits integer type for this platform
    #endif

    // 16 bits integer types
    #if USHRT_MAX == 0xFFFF
        typedef signed short Int16;
        typedef unsigned short Uint16;
    #elif UINT_MAX == 0xFFFFFFFF
        typedef signed int Int16;
        typedef unsigned int Uint16;
    #elif ULONG_MAX == 0xFFFFFFFF
        typedef signed long Int16;
        typedef unsigned long Uint16;
    #else
        #error No 16 bits integer type for this platform
    #endif

    // 32 bits integer types
    #if USHRT_MAX == 0xFFFFFFFF
        typedef signed short Int32;
        typedef unsigned short Uint32;
    #elif UINT_MAX == 0xFFFFFFFF
        typedef signed int Int32;
        typedef unsigned int Uint32;
    #elif ULONG_MAX == 0xFFFFFFFF
        typedef signed long int Int32;
        typedef unsigned long int Uint32;
    #else
        #error No 32 bits integer type for this platform
    #endif
}
#elif ULONG_MAX == 0xFFFFFFFF

typedef signed long Int32;
typedef unsigned long Uint32;
#else
#error No 32 bits integer type for this platform
#endif

#endif // SFML_CONFIG_HPP

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
<table>
<thead>
<tr>
<th>Main Page</th>
<th>Namespaces</th>
<th>Classes</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>File List</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Main Page
Namespaces
Classes
Files

File List
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an
//    acknowledgment in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.

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

// Headers
#include <SFML/Graphics/FontLoader.hpp>
#include <SFML/Graphics/Color.hpp>
#include <SFML/Graphics/Font.hpp>
#include <SFML/Graphics/Image.hpp>
#include <SFML/Graphics/GraphicsContext.hpp>
#include <math.h>
#include <iostream>
#include <map>
#include <vector>
namespace
{
    // Functor to sort glyphs by size
    struct SizeCompare
    {
        bool operator()(FT_BitmapGlyph Glyph1,
                        FT_BitmapGlyph Glyph2)
        {
            return Glyph2->bitmap.rows < Glyph1.
        }
    };
}

namespace sf
{
    namespace priv
    {
        FontLoader& FontLoader::GetInstance()
        {
            static FontLoader Instance;
            return Instance;
        }
        FontLoader::FontLoader()
        {
            // Initialize FreeType library
            FT_Error Error = FT_Init_FreeType(&myLibrary);
            if (Error)
            {
                std::cerr << "Failed to initialize FreeType library";
                return;
            }
        }
    }
}
FontLoader::~FontLoader()
{
    // Shutdown FreeType library
    if (myLibrary)
    {
        FT_Done_FreeType(myLibrary);
    }
}

bool FontLoader::LoadFontFromFile(const std::string& Filename)
{
    // Check if FreeType is correctly initialized
    if (!myLibrary)
    {
        std::cerr << "Failed to load font " << Filename << "\n"
        return false;
    }

    // Create a new font face from the specified file
    FT_Face FontFace;
    FT_Error Error = FT_New_Face(myLibrary, Filename.c_str(), 0, &FontFace);
    if (Error)
    {
        std::cerr << "Failed to load font " << Filename << "\n"
        return false;
    }

    // Create the bitmap font
    Error = CreateBitmapFont(FontFace, CharSize, Charset, LoadedFont);
    if (Error)
    {
        std::cerr << "Failed to load font " << Filename << "\n"
        return false;
    }

    // Delete the font
    FT_Done_Face(FontFace);
    return Error == 0;
}
bool FontLoader::LoadFontFromMemory(const char* Data, std::size_t SizeInBytes) {
    // Check if Freetype is correctly initialized
    if (!myLibrary)
        {
            std::cerr << "Failed to load font from memory,
            return false;
        }
    // Create a new font face from the specified memory data
    FT_Face FontFace;
    FT_Error Error = FT_New_Memory_Face(myLibrary, reinterpret_cast<const FT_Byte*>(Data), static_cast<FT_Long>(SizeInBytes), 0, &FontFace);
    if (Error)
        {
            std::cerr << "Failed to load font from memory,
            return false;
            }
    // Create the bitmap font
    Error = CreateBitmapFont(FontFace, CharSize, Charset, LoadedFont);
    if (Error)
        std::cerr << "Failed to load font from memory,
        return false;
    // Delete the font
    FT_Done_Face(FontFace);
    return Error == 0;
}

FT_Error FontLoader::CreateBitmapFont(FT_Face FontFace, {
    // Make sure we have a valid context
    priv::GraphicsContext Ctx;
    //
Let's find how many characters to put in each row to make them fit into a squared texture

GLint MaxSize;
GLCheck(glGetIntegerv(GL_MAX_TEXTURE_SIZE, &MaxSize));

int NbChars = static_cast<int>(sqrt(static_cast<double>(Charset.length())) * 0.75);

// Clamp the character size to make sure we won't create a texture too big
if (NbChars * CharSize >= static_cast<unsigned int>(MaxSize))
    CharSize = MaxSize / NbChars;

// Initialize the dimensions
unsigned int Left = 0;
unsigned int Top = 0;
unsigned int TexWidth = Image::GetValidTextureSize;
unsigned int TexHeight = CharSize * NbChars;
std::vector<unsigned int> Tops(TexWidth, 0);

// Create a pixel buffer for rendering every glyph
std::vector<Uint8> GlyphsBuffer(TexWidth * TexHeight * 4);

// Setup the font size
FT_Error Error = FT_Set_Pixel_Sizes(FontFace, CharSize, CharSize);
if (Error)
    return Error;

// Select the unicode character map
Error = FT_Select_Charmap(FontFace, FT_ENCODING_UNICODE);
if (Error)
    return Error;

// Render all glyphs and sort them by size
typedef std::multimap<FT_BitmapGlyph, Uint32, SizeCompare> GlyphTable;
GlyphTable Glyphs;
for (std::size_t i = 0; i < Charset.length(); ++i)
{
    // Load the glyph corresponding to the character
    Error = FT_Load_Char(FontFace, Charset[i], FT_LOAD_TARGET_NORMAL);
    if (Error)
return Error;

// Convert the glyph to a bitmap (ie. rasterize it)
FT_Glyph Glyph;
Error = FT_Get_Glyph(FontFace->glyph, &Glyph);
if (Error)
    return Error;
FT_Glyph_To_Bitmap(&Glyph, FT_RENDER_MODE_NORMAL, 0, 1);
FT_BitmapGlyph BitmapGlyph = (FT_BitmapGlyph)Glyph;

/* Add it to the sorted table of glyphs */
Glyphs.insert(std::make_pair(BitmapGlyph, Charset[i]);
}

// Copy the rendered glyphs into the texture
unsigned int MaxHeight = 0;
std::map<Uint32, IntRect> Coords;
for (GlyphTable::const_iterator i = Glyphs.begin(); i != Glyphs.end(); ++i)
{
    // Get the bitmap of the current glyph
    Glyph& CurGlyph = LoadedFont.myGlyphs[i->second];
    FT_BitmapGlyph BitmapGlyph = i->first;
    FT_Bitmap& Bitmap = BitmapGlyph;

    // Make sure we don't go over the texture
    if (Left + Bitmap.width + 1 >= TexWidth)
        Left = 0;

    // Compute the top coordinate
    Top = Tops[Left];
    for (int x = 0; x < Bitmap.width + 1; ++x)
        Top = std::max(Top, Tops[Left + x]);
    Top++;

    // Make sure we don't go over the texture
    if (Top + Bitmap.rows + 1 >= TexHeight)
        {
            // Compute the right coordinate
            Right = TexWidth - (Top + Bitmap.rows);
            // Copy the bitmap
            std::copy(Bitmap.buffer, Bitmap.buffer + Bitmap.width * Bitmap.rows, TexBuffer + TexWidth * Top);
            // Update the coordinate
            Coords[BitmapGlyph] = IntRect(Top, Left, TexWidth - (Top + Bitmap.rows), Bitmap.width);
        }
TexHeight *= 2;
GlyphsBuffer.resize(TexWidth * TexHeight)
}

// Store the character's position and size
CurGlyph.Rectangle.Left = BitmapGlyph->left;
CurGlyph.Rectangle.Top = -BitmapGlyph->top;
CurGlyph.Advance = BitmapGlyph->root.advance.x >> 16;

// Texture size may change, so let the texture coordinates be calculated later
Coords[i->second] = IntRect(Left + 1, Top + 1, Left + Bitmap.width + 1, Top + Bitmap.rows + 1);

// Draw the glyph into our bitmap font
const Uint8* Pixels = Bitmap.buffer;
for (int y = 0; y < Bitmap.rows; ++y)
{
    for (int x = 0; x < Bitmap.width; ++x)
    {
        std::size_t Index = x + Left + 1 + (y + Top + 1) * TexWidth;
        GlyphsBuffer[Index * 4 + 0] = 255;
        GlyphsBuffer[Index * 4 + 1] = 255;
        GlyphsBuffer[Index * 4 + 2] = 255;
        GlyphsBuffer[Index * 4 + 3] = Pixels[x];
    }
    Pixels += Bitmap.pitch;
}

// Update the rendering coordinates
for (int x = 0; x < Bitmap.width + 1; ++x)
    Tops[Left + x] = Top + Bitmap.rows;
Left += Bitmap.width + 1;
if (Top + Bitmap.rows > MaxHeight)
    MaxHeight = Top + Bitmap.rows;

// Delete the glyph
FT_Done_Glyph((FT_Glyph)BitmapGlyph);

// Create the font's texture
TexHeight = MaxHeight + 1;
GlyphsBuffer.resize(TexWidth * TexHeight * 4);
LoadedFont.myTexture.LoadFromPixels(TexWidth, TexHeight, &GlyphsBuffer[0]);

// Now that the texture is created, we can precompute texture coordinates
for (std::size_t i = 0; i <Charset.size(); ++i)
{
    Uint32 CurChar =Charset[i];
    LoadedFont.myGlyphs[CurChar].TexCoords =
}

// Update the character size (it may have been updated)
LoadedFont.myCharSize = CharSize;
return 0;

std::string FontLoader::GetErrorDesc(FT_Error Error)
{
    switch (Error)
    {
        // Generic errors
        case FT_Err_Cannot_Open_Resource :
        case FT_Err_Unknown_File_Format :
        case FT_Err_Invalid_File_Format :
        case FT_Err_Invalid_Version :
        case FT_Err_Lower_Module_Version :
        case FT_Err_Invalid_Argument :
        case FT_Err_Unimplemented_Feature :
        case FT_Err_Invalid_Table :
        case FT_Err_Invalid_Offset :
case FT_Err_Invalid_Glyph_Index :  
  case FT_Err_Invalid_Character_Code :  
  case FT_Err_Invalid_Glyph_Format :  
  case FT_Err_Cannot_Render_Glyph :  
  case FT_Err_Invalid_Outline :  
  case FT_Err_Invalid_Composite :  
  case FT_Err_Too_Many_Hints :  
  case FT_Err_Invalid_Pixel_Size :  

  // Handle errors
  case FT_Err_Invalid_Handle :  
  case FT_Err_Invalid_Library_Handle :  
  case FT_Err_Invalid_Driver_Handle :  
  case FT_Err_Invalid_Face_Handle :  
  case FT_Err_Invalid_Size_Handle :  
  case FT_Err_Invalid_Slot_Handle :  
  case FT_Err_Invalid_CharMap_Handle :  
  case FT_Err_Invalid_Cache_Handle :  
  case FT_Err_Invalid_Stream_Handle :  

  // Driver errors
  case FT_Err_Too_Many_Drivers :  
  case FT_Err_Too_Many_Extensions :  

  // Memory errors
  case FT_Err_Out_Of_Memory :  
  case FT_Err_Unlisted_Object :  

  // Stream errors
  case FT_Err_Cannot_Open_Stream :  
  case FT_Err_Invalid_Stream_Seed :  
  case FT_Err_Invalid_Stream_Skip :  
  case FT_Err_Invalid_Stream_Read :  
  case FT_Err_Invalid_Stream_Operation :  
  case FT_Err_Invalid_Frame_Operation :  
  case FT_Err_Nested_Frame_Access :  

case FT_Err_Invalid_Frame_Read :
// Raster errors
case FT_Err_Raster_Uninitialized :
case FT_Err_Raster_Corrupted :
case FT_Err_Raster_Overflow :
case FT_Err_Raster_Negative_Height :
// Cache errors
case FT_Err_Too_Many_Caches :
// TrueType and SFNT errors
case FT_Err_Invalid_Opcode :
case FT_Err_Too_Few_Arguments :
case FT_Err_Stack_Overflow :
case FT_Err_Code_Overflow :
case FT_Err_Bad_Argument :
case FT_Err_Divide_By_Zero :
case FT_Err_Invalid_Reference :
case FT_Err_Debug_OpCode :
case FT_Err_ENDF_In_Exec_Stream :
case FT_Err_Nested_DEFS :
case FT_Err_Invalid_CodeRange :
case FT_Err_Execution_Too_Long :
case FT_Err_Too_Many_Function_Defs :
case FT_Err_Too_Many_Instruction_Defs :
case FT_Err_Table_Missing :
case FT_Err_Horiz_Header_Missing :
case FT_Err_Locations_Missing :
case FT_Err_Name_Table_Missing :
case FT_Err_CMap_Table_Missing :
case FT_Err_Hmtx_Table_Missing :
case FT_Err_Post_Table_Missing :
case FT_Err_Invalid_Horiz_Metrics :
case FT_Err_Invalid_CharMap_Format :
case FT_Err_Invalid_PPem :
case FT_Err_Invalid_Vert_Metrics :
case FT_Err_Could_Not_Find_Context :

case FT_Err_Invalid_Post_Table_Format :

case FT_Err_Invalid_Post_Table :

// CCF, CID and Type 1 errors

case FT_Err_Syntax_Error :

case FT_Err_Stack_Underflow :

case FT_Err_Ignore :

// BDF errors

case FT_Err_Missing_Startfont_Field :

case FT_Err_Missing_Font_Field :

case FT_Err_Missing_Size_Field :

case FT_Err_Missing_Chars_Field :

case FT_Err_Missing_Startchar_Field :

case FT_Err_Missing_Encoding_Field :

case FT_Err_Missing_Bbx_Field :

} } // namespace priv

} // namespace sf

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::

Documentation generated by doxygen 1.5.2 ::
<table>
<thead>
<tr>
<th>Main Page</th>
<th>Namespaces</th>
<th>Classes</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>File List</strong></td>
</tr>
</tbody>
</table>
//
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)
//
// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.
// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:
// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is not required.
// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.
// 3. This notice may not be removed or altered from any source distribution.

#ifndef SFML_FONTLOADER_HPP
#define SFML_FONTLOADER_HPP

// Headers
#include <SFML/System/NonCopyable.hpp>
#include <SFML/System/Unicode.hpp>
#include <ft2build.h>
#include FT_FREETYPE_H
#include <string>

#endif // SFML_FONTLOADER_HPP
namespace sf {
    class Font;

namespace priv {
    class FontLoader : NonCopyable {
        public:
            static FontLoader& GetInstance();

            bool LoadFontFromFile(const std::string& Filename);
            bool LoadFontFromMemory(const char* Data, std::size_t SizeInBytes);

        private:
            FontLoader();
            ~FontLoader();

            FT_Error CreateBitmapFont(FT_Face FontFace, static std::string GetErrorDesc(FT_Error Error);

        } // namespace priv
    } // namespace sf

#endif // SFML_FONTLOADER_HPP
Graphics.hpp

00001 //
00002 // SFML - Simple and Fast Multimedia Library
00003 // Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)
00004 //
00005 // This software is provided 'as-is', without any express or implied
00006 // warranty. In no event will the authors be held liable for any
00007 // damages arising from the use of this software.
00008 //
00009 // Permission is granted to anyone to use this software for any
00010 // purpose, including commercial applications, and to alter it and
00011 // redistribute it freely, subject to the following restrictions:
00012 //
00013 // 1. The origin of this software must not be misrepresented;
00014 //    you must not claim that you wrote the original software.
00015 //    If you use this software in a product, an
00016 //    acknowledgment in the product documentation would be
00017 //    appreciated but is not required.
00018 //
00019 // 2. Altered source versions must be plainly marked as such,
00020 //    and must not be misrepresented as being the original
00021 //    software.
00022 //
00023 // 3. This notice may not be removed or altered.
00024 //
00025 #ifndef SFML_GRAPHICS_HPP
00026 #define SFML_GRAPHICS_HPP
00027
00029 // Headers
00031
00032 #include <SFML/Window.hpp>
00033 #include <SFML/Graphics/Color.hpp>
00034 #include <SFML/Graphics/Font.hpp>
00035 #include <SFML/Graphics/Glyph.hpp>
00036 #include <SFML/Graphics/Image.hpp>
00037 #include <SFML/Graphics/PostFX.hpp>
```cpp
#include <SFML/Graphics/RenderWindow.hpp>
#include <SFML/Graphics/Shape.hpp>
#include <SFML/Graphics/Sprite.hpp>
#include <SFML/Graphics/String.hpp>
#include <SFML/Graphics/View.hpp>

#endif // SFML_GRAPHICS_HPP
```
SFML: Simple and Fast Multimedia Library

Main Page  Namespaces  Classes  Files

File List
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any
// implied warranty. In no event will the authors be held liable for
// any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any
// purpose, including commercial applications, and to alter it and
// redistribute it freely, subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
// you must not claim that you wrote the original software.
// If you use this software in a product, an
// acknowledgment in the product documentation would be appreciated.

// 2. Altered source versions must be plainly marked as such,
// and must not be misrepresented as being the original software.

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

// Headers
#include <SFML/Graphics/GraphicsContext.hpp>
#include <SFML/Window/Context.hpp>

namespace
{
    // One time initialization of 3rd party libraries.
    // We use a global function with a static boolean
    // than directly a global boolean, to avoid
    // of global variables initializations across
}
void InitGraphicsLibs()
{
    static bool InitDone = false;
    if (!InitDone)
    {
        // Initialize GLEW
        glewInit();
        InitDone = true;
    }
}

namespace sf
{
    namespace priv
    {
        GraphicsContext::GraphicsContext()
        {
            // Activate the global context
            if (!Context::IsContextActive())
            {
                Context::GetGlobal().SetActive(true);
                myActivated = true;
            }
            else
            {
                myActivated = false;
            }

            // Make sure third party libraries are initialized
            InitGraphicsLibs();
        }
    }
}
GraphicsContext::~GraphicsContext()
{
    // Deactivate the global context
    if (myActivated)
        Context::GetGlobal().SetActive(false);
}

// namespace priv

// namespace sf
GraphicsContext.hpp

#ifndef SFML_GRAPHICSCONTEXT_HPP
#define SFML_GRAPHICSCONTEXT_HPP

// Headers
#include <SFML/Config.hpp>
#include <SFML/System/NonCopyable.hpp>
#include <SFML/Graphics/GLEW/glew.h>
#include <iostream>
#include <string>

#endif // SFML_GRAPHICSCONTEXT_HPP
namespace sf {
    class Context;

namespace priv {
    class GraphicsContext : NonCopyable {
        public:
            GraphicsContext();
            ~GraphicsContext();

        private:
            // Member data
            bool myActivated;
    }
}

} // namespace priv

#ifdef SFML_DEBUG
    // In debug mode, perform a test on every OpenGL call
    #define GLCheck(Func) ((Func), GLCheckError(__FILE__, __LINE__))
#else
    // Else, we don't add any overhead
    #define GLCheck(Func) (Func)
#endif

inline void GLCheckError(const std::string& File,
// Get the last error
GLenum ErrorCode = glGetError();

if (ErrorCode != GL_NO_ERROR)
{
    std::string Error = "unknown error";
    std::string Desc = "no description";

    // Decode the error code
    switch (ErrorCode)
    {
    case GL_INVALID_ENUM :
        {
            Error = "GL_INVALID_ENUM";
            Desc = "an unacceptable value has been specified for an enumerated argument";
            break;
        }
    case GL_INVALID_VALUE :
        {
            Error = "GL_INVALID_VALUE";
            Desc = "a numeric argument is out of range";
            break;
        }
    case GL_INVALID_OPERATION :
        {
            Error = "GL_INVALID_OPERATION";
            Desc = "the specified operation is not allowed in the current state";
            break;
        }
    case GL_STACK_OVERFLOW :
        {
            Error = "GL_STACK_OVERFLOW";
            Desc = "this command would cause a stack overflow";
            break;
        }
    case GL_STACK_UNDERFLOW :
        {
            Error = "GL_STACK_UNDERFLOW";
            Desc = "this command would cause a stack underflow";
            break;
        }
    case GL_TABLE_TOO_LARGE :
        {
            Error = "GL_TABLE_TOO_LARGE";
            Desc = "table storage exceeds maximum limit";
            break;
        }
    case GL_OUT_OF_MEMORY :
        {
            Error = "GL_OUT_OF_MEMORY";
            Desc = "insufficient resources are available to complete the command";
            break;
        }
    case GL_INVALID_FRAMEBUFFER_OPERATION :
        {
            Error = "GL_INVALID_FRAMEBUFFER_OPERATION";
            Desc = "render target arrays are associated with a non-writeable renderbuffer";
            break;
        }
    case GL_CONTEXT_LOST :
        {
            Error = "GL_CONTEXT_LOST";
            Desc = "context lost";
            break;
        }
    case GL_INVALID_INDEX :
        {
            Error = "GL_INVALID_INDEX";
            Desc = "the specified index is not a valid index into the array specified in the corresponding array parameter";
            break;
        }
    case GL_INVALIDAPIView :
        {
            Error = "GL_INVALIDAPIView";
            Desc = "invalid view point";
            break;
        }
    case GL_INVALID_NAME :
        {
            Error = "GL_INVALID_NAME";
            Desc = "the specified name is invalid";
            break;
        }
    case GL_INVALID_OPERATION_IN_SUBPASS剥：
        {
            Error = "GL_INVALID_OPERATION_IN_SUBPASS剥：";
            Desc = "the specified operation is not allowed in the current subpass";
            break;
        }
    case GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT :
        {
            Error = "GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT";
            Desc = "invalid attachment to framebuffer";
            break;
        }
    case GL_FRAMEBUFFER_INCOMPLETE_MISSING_ATTACHMENT :
        {
            Error = "GL_FRAMEBUFFER_INCOMPLETE_MISSING_ATTACHMENT";
            Desc = "missing attachment to framebuffer";
            break;
        }
    case GL_FRAMEBUFFER_INCOMPLETE_DRAW_BUFFER :
        {
            Error = "GL_FRAMEBUFFER_INCOMPLETE_DRAW_BUFFER";
            Desc = "invalid draw buffer";
            break;
        }
    case GL_FRAMEBUFFER_INCOMPLETE_READ_BUFFER :
        {
            Error = "GL_FRAMEBUFFER_INCOMPLETE_READ_BUFFER";
            Desc = "invalid read buffer";
            break;
        }
    case GL_FRAMEBUFFER_INCOMPLETE_SUBPASSእכוונוהן":
        {
            Error = "GL_FRAMEBUFFER_INCOMPLETE_SUBPASSእ microtime");
            Desc = "invalid subpass in framebuffer with a renderbuffer target";
            break;
        }
    case GL_FRAMEBUFFER_INCOMPLETE_MULTISAMPLE :
        {
            Error = "GL_FRAMEBUFFER_INCOMPLETE_MULTISAMPLE";
            Desc = "invalid multisample attachment to framebuffer";
            break;
        }
    case GL_FRAMEBUFFER_INCOMPLETE_LAYER_TARGETS :
        {
            Error = "GL_FRAMEBUFFER_INCOMPLETE_LAYER_TARGETS";
            Desc = "specified layers of framebuffer have different targets";
            break;
        }
    case GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_VIEWPORT :
        {
            Error = "GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_VIEWPORT";
            Desc = "viewport dimensions are not valid for the specified attachment";
            break;
        }
    case GL_FRAMEBUFFER_INCOMPLETE_LAYOUT :
        {
            Error = "GL_FRAMEBUFFER_INCOMPLETE_LAYOUT";
            Desc = "specified layout of framebuffer is not supported";
            break;
        }
    case GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_LAYER_COUNT :
        {
            Error = "GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_LAYER_COUNT";
            Desc = "invalid number of layers for specified attachment";
            break;
        }
    case GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_LAYER_NAME :
        {
            Error = "GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_LAYER_NAME";
            Desc = "invalid name for specified attachment layer";
            break;
        }
    case GL_FRAMEBUFFER_INCOMPLETE_LAYER_TARGETS :
        {
            Error = "GL_FRAMEBUFFER_INCOMPLETE_LAYER_TARGETS";
            Desc = "invalid targets for specified layers of framebuffer";
            break;
        }
    case GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_HEIGHT :
        {
            Error = "GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_HEIGHT";
            Desc = "specified attachment height is not valid";
            break;
        }
    case GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_WIDTH :
        {
            Error = "GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_WIDTH";
            Desc = "specified attachment width is not valid";
            break;
        }
    case GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_FORMAT :
        {
            Error = "GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_FORMAT";
            Desc = "specified attachment format is not valid";
            break;
        }
    case GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_TYPE :
        {
            Error = "GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_TYPE";
            Desc = "specified attachment type is not valid";
            break;
        }
    case GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_LEVEL :
        {
            Error = "GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_LEVEL";
            Desc = "specified attachment level is not valid";
            break;
        }
    case GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_SAMPLES :
        {
            Error = "GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_SAMPLES";
            Desc = "specified attachment samples is not valid";
            break;
        }
    case GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_COLOR.index is: 5"
        {
            Error = "GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_COLOR.entry is: 5";
            Desc = "invalid color attachment index specified in attachment parameter";
            break;
        }
    case GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_DEPTH.index is: 5"
        {
            Error = "GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_DEPTH.entry is: 5";
            Desc = "invalid depth attachment index specified in attachment parameter";
            break;
        }
    case GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_STENCIL.index is: 5"
        {
            Error = "GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_STENCIL.entry is: 5";
            Desc = "invalid stencil attachment index specified in attachment parameter";
            break;
        }
    case GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_STENCILindex is: 5"
        {
            Error = "GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_STENCIL.entry is: 5";
            Desc = "invalid stencil attachment type specified in attachment parameter";
            break;
        }
    case GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_COLOR.index is: 5"
        {
            Error = "GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_COLOR.entry is: 5";
            Desc = "invalid color attachment type specified in attachment parameter";
            break;
        }
    case GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_DEPTH.index is: 5"
        {
            Error = "GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_DEPTH.entry is: 5";
            Desc = "invalid depth attachment type specified in attachment parameter";
            break;
        }
    case GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_STENCIL.index is: 5"
        {
            Error = "GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_STENCIL.entry is: 5";
            Desc = "invalid stencil attachment type specified in attachment parameter";
            break;
        }
    case GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_COLOR.index is: 5"
        {
            Error = "GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_COLOR.entry is: 5";
            Desc = "invalid color attachment type specified in attachment parameter";
            break;
        }
    case GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_DEPTH.index is: 5"
        {
            Error = "GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_DEPTH.entry is: 5";
            Desc = "invalid depth attachment type specified in attachment parameter";
            break;
        }
    case GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_STENCIL.index is: 5"
        {
            Error = "GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_STENCIL.entry is: 5";
            Desc = "invalid stencil attachment type specified in attachment parameter";
            break;
        }
    case GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_COLOR.index is: 5"
        {
            Error = "GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_COLOR.entry is: 5";
            Desc = "invalid color attachment type specified in attachment parameter";
            break;
        }
    case GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_DEPTH.index is: 5"
        {
            Error = "GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_DEPTH.entry is: 5";
            Desc = "invalid depth attachment type specified in attachment parameter";
            break;
        }
    case GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_STENCIL.index is: 5"
        {
            Error = "GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_STENCIL.entry is: 5";
            Desc = "invalid stencil attachment type specified in attachment parameter";
            break;
        }
    case GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_COLOR.index is: 5"
        {
            Error = "GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_COLOR.entry is: 5";
            Desc = "invalid color attachment type specified in attachment parameter";
            break;
        }
    case GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_DEPTH.index is: 5"
        {
            Error = "GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_DEPTH.entry is: 5";
            Desc = "invalid depth attachment type specified in attachment parameter";
            break;
        }
    case GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_STENCIL.index is: 5"
        {
            Error = "GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_STENCIL.entry is: 5";
            Desc = "invalid stencil attachment type specified in attachment parameter";
            break;
        }
    case GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_COLOR.index is: 5"
        {
            Error = "GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_COLOR.entry is: 5";
            Desc = "invalid color attachment type specified in attachment parameter";
            break;
        }
    case GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_DEPTH.index is: 5"
        {
            Error = "GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_DEPTH.entry is: 5";
            Desc = "invalid depth attachment type specified in attachment parameter";
            break;
        }
    case GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_STENCIL.index is: 5"
        {
            Error = "GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_STENCIL.entry is: 5";
            Desc = "invalid stencil attachment type specified in attachment parameter";
            break;
        }
    case GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_COLOR.index is: 5"
        {
            Error = "GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_COLOR.entry is: 5";
            Desc = "invalid color attachment type specified in attachment parameter";
            break;
        }
    case GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_DEPTH.index is: 5"
        {
            Error = "GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_DEPTH.entry is: 5";
            Desc = "invalid depth attachment type specified in attachment parameter";
            break;
        }
    case GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_STENCIL.index is: 5"
        {
            Error = "GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_STENCIL.entry is: 5";
            Desc = "invalid stencil attachment type specified in attachment parameter";
break;
}

case GL_STACK_UNDERFLOW :
{
   Error = "GL_STACK_UNDERFLOW";
   Desc = "this command would cause a stack underflow"
   break;
}

case GL_OUT_OF_MEMORY :
{
   Error = "GL_OUT_OF_MEMORY";
   Desc = "there is not enough memory left to execute the command"
   break;
}

case GL_INVALID_FRAMEBUFFER_OPERATION_EXT :
{
   Error = "GL_INVALID_FRAMEBUFFER_OPERATION_EXT"
   Desc = "the object bound to FRAMEBUFFER_BINDING_EXT is not "framebuffer complete"
   break;
}

// Log the error
std::cerr << "An internal OpenGL call failed in "
   << File.substr(File.find_last_of( ", ", "
   << Error << ", " << Desc
   << std::endl;
}

// namespace sf

// SFML_GRAPHICSCONTEXT_HPP
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any
// warranty of any kind, express or implied.
// In no event will the authors be held liable for
// any damages arising from the use of this software.

// Permission is granted to anyone to use this software
// for any purpose, including commercial applications,
// and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misinterpreted;
// you must not claim that you wrote the original code.

// 2. Altered source versions must be plainly marked
// and must not be misrepresented as being the
// original code.

// 3. This notice may not be removed or altered.

// Headers
#include <SFML/Graphics/ImageLoader.hpp>
extern "C"
{
#include <SFML/Graphics/libjpeg/jpeglib.h>
#include <SFML/Graphics/libjpeg/jerror.h>
}
#include <SFML/Graphics/libpng/png.h>
#include <SFML/Graphics/SOIL/SOIL.h>
#include <iostream>
namespace {
    void PngErrorHandler(png_structp Png, png_const_charp Message) {
        std::cerr << "Failed to write PNG image.
        longjmp(Png->jmpbuf, 1);
    }
}

namespace sf {
    namespace priv {
        ImageLoader& ImageLoader::GetInstance() {
            static ImageLoader Instance;
            return Instance;
        }

        ImageLoader::ImageLoader() {
            // Nothing to do
        }

        ImageLoader::~ImageLoader() {
            // Nothing to do
        }

        bool ImageLoader::LoadImageFromFile(const std::string& Filename, std::vector<Color>& Pixels) {
            // Implementation
        }
    }
}

// Clear the array (just in case)
Pixels.clear();

// Load the image and get a pointer to the pixels
int ImgWidth, ImgHeight, ImgChannels;
unsigned char* PixelsPtr = SOIL_load_image(Filename.c_str(), &ImgWidth, &ImgHeight, &ImgChannels);

if (PixelsPtr)
{
    // Assign the image properties
    Width = ImgWidth;
    Height = ImgHeight;

    // Copy the loaded pixels to the pixel buffer
    Pixels.resize(Width * Height);
    memcpy(&Pixels[0], PixelsPtr, Width * Height * 4);

    // Free the loaded pixels (they are now in our own pixel buffer)
    SOIL_free_image_data(PixelsPtr);

    return true;
}
else
{
    // Error, failed to load the image
    std::cerr << "Failed to load image " << Filename << std::endl;
    return false;
}

bool ImageLoader::LoadImageFromMemory(const char* Filename)
{
    // Clear the array (just in case)
    Pixels.clear();

// Load the image and get a pointer to the pixels in memory
const unsigned char* Buffer = reinterpret_cast<const unsigned char*>(SizeInBytes);
int Size = static_cast<int>(SizeInBytes);
int ImgWidth, ImgHeight, ImgChannels;
unsigned char* PixelsPtr = SOIL_load_image_from_memory(Buffer, Size, &ImgWidth, &ImgHeight, &ImgChannels, SOIL_LOAD_RGBA);

if (PixelsPtr)
{
    // Assign the image properties
    Width = ImgWidth;
    Height = ImgHeight;

    // Copy the loaded pixels to the pixel buffer
    Pixels.resize(Width * Height);
    memcpy(&Pixels[0], PixelsPtr, Width * Height * 4);

    // Free the loaded pixels (they are now in our own pixel buffer)
    SOIL_free_image_data(PixelsPtr);

    return true;
}
else
{
    // Error, failed to load the image
    std::cerr << "Failed to load image from memory."
    return false;
}

bool ImageLoader::SaveImageToFile(const std::string& Filename)
{
    // Deduce the image type from its extension
    int Type = -1;
    if (Filename.size() > 3)
    {

```cpp
std::string Extension = Filename.substr(Filename.size() - 3);

if (Extension == "bmp" || Extension == "tga" ||
    Extension == "dds"
    // Special handling for PNG and JPG -- not handled by SOIL
    else if (Extension == "png" || Extension == "jpg"
        else if (Type == -1)
```

```cpp
if (Type == -1)
    { // Error, incompatible type
        std::cerr << "Failed to save image """ << Filename << 
        return false;
    }
```

```cpp
// Finally save the image
const unsigned char* PixelsPtr = reinterpret_cast<const unsigned char*>(Pixels);
if (!SOIL_save_image(Filename.c_str(), Type,
    // Error, failed to save the image
    std::cerr << "Failed to save image "" << Filename << 
    return false;
}
```

```cpp
return true;
}
```

```cpp
bool ImageLoader::WriteJpg(const std::string& Filename, ...
```
    return false;
}

// Initialize the error handler
jpeg_compress_struct CompressInfo;
jpeg_error_mgr ErrorManager;
CompressInfo.err = jpeg_std_error(&ErrorManager);

// Initialize all the writing and compression infos
jpeg_create_compress(&CompressInfo);
CompressInfo.image_width = Width;
CompressInfo.image_height = Height;
CompressInfo.input_components = 3;
CompressInfo.in_color_space = JCS_RGB;
jpeg_stdio_dest(&CompressInfo, File);
jpeg_set_defaults(&CompressInfo);
jpeg_set_quality(&CompressInfo, 90, TRUE);

// Get rid of the aplha channel
std::vector<Uint8> PixelsBuffer(Width * Height);
for (std::size_t i = 0; i < Pixels.size(); ++i)
{
    PixelsBuffer[i * 3 + 0] = Pixels[i].r;
    PixelsBuffer[i * 3 + 1] = Pixels[i].g;
    PixelsBuffer[i * 3 + 2] = Pixels[i].b;
}

Uint8* PixelsPtr = &PixelsBuffer[0];

// Start compression
jpeg_start_compress(&CompressInfo, TRUE);

// Write each row of the image
while (CompressInfo.next_scanline < CompressInfo.image_height)
{
    JSAMPROW RowPointer = PixelsPtr + (CompressInfo.next_sample + (CompressInfo.next_scanline * CompressInfo.image_width));
    jpeg_write_scanlines(&CompressInfo, &RowPointer, 1);
}
// Finish compression
jpeg_finish_compress(&CompressInfo);
jpeg_destroy_compress(&CompressInfo);

// Close the file
fclose(File);

return true;

bool ImageLoader::WritePng(const std::string& Filename) {
    // Open the file to write in
    FILE* File = fopen(Filename.c_str(), "wb");
    if (!File) {
        std::cerr << "Failed to save image file" << std::endl;
        return false;
    }
    // Create the main PNG structure
    png_structp Png = png_create_write_struct(PNG_LIBPNG_VER_STRING, NULL, &PngErrorHandler, NULL);
    if (!Png) {
        fclose(File);
        std::cerr << "Failed to save image file" << std::endl;
        return false;
    }
    // Initialize the image informations
    png_infop PngInfo = png_create_info_struct(Png);
    if (!PngInfo) {
        fclose(File);
        png_destroy_write_struct(&Png, NULL);
    } else {
        // ...
std::cerr << "Failed to save image file"
return false;
}

// For proper error handling...
if (setjmp(Png->jmpbuf))
{
png_destroy_write_struct(&Png, &PngInfo);
return false;
}

// Link the file to the PNG structure
png_init_io(Png, File);

// Set the image informations
png_set_IHDR(Png, PngInfo, Width, Height, 8,

// Write the header
png_write_info(Png, PngInfo);

// Get the pointers to the pixels rows into
png_byte* PixelsPtr = (png_byte*)&Pixels[0];
std::vector<png_byte*> RowPointers(Height);
for (unsigned int i = 0; i < Height; ++i)
{
    RowPointers[i] = PixelsPtr;
    PixelsPtr += Width * 4;
}

// Write pixels row by row
png_set_rows(Png, PngInfo, &RowPointers[0]);
png_write_png(Png, PngInfo, PNG_TRANSFORM_IDENTITY,

// Finish writing the file
png_write_end(Png, PngInfo);

// Cleanup resources
00324    png_destroy_write_struct(&Png, &PngInfo);
00325    fclose(File);
00326
00327    return true;
00328 }
00329
00330 } // namespace priv
00331
00332 } // namespace sf
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gomila@gmail.com)

// This software is provided 'as-is', without any expressed or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
//    and not be misrepresented as being the original software.

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

#ifndef SFML_IMAGELOADER_HPP
#define SFML_IMAGELOADER_HPP

// Headers
#include <SFML/Graphics/Color.hpp>
#include <SFML/System/NonCopyable.hpp>
#include <string>
#include <vector>

namespace sf {
namespace priv
{
    class ImageLoader : NonCopyable
    {
        public:
        static ImageLoader& GetInstance();
        bool LoadImageFromFile(const std::string& Filename, std::vector<Color>& Pixels);
        bool LoadImageFromMemory(const char* Data, std::size_t SizeInBytes, std::vector<Color>& Pixels);
        bool SaveImageToFile(const std::string& Filename);
    }

    private:
    ImageLoader();
    ~ImageLoader();
    bool WriteJpg(const std::string& Filename, const std::vector<Color>& Pixels);
    bool WritePng(const std::string& Filename, const std::vector<Color>& Pixels);
};

} // namespace priv

} // namespace sf

#endif // SFML_IMAGELOADER_HPP

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
Main Page  Namespaces  Classes  Files

File List
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.

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

// Headers
#include <SFML/Window/Joystick.hpp>

namespace sf {
    namespace priv {
        void Joystick::Initialize(unsigned int Index) {
            // Reset the joystick state
        }
    }
}

// Initialize the Index-th available joystick

JoystickState Joystick::UpdateState()
{
    // Fill a JoystickState instance with the current joystick state
    JoystickState s;
    return s;
}

bool Joystick::HasAxis(Joy::Axis Axis) const
{
    return false;
}

unsigned int Joystick::GetButtonsCount() const
{
    // Return number of supported buttons
    return 0;
}

// namespace priv

// namespace sf
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.

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

// Headers
#include <SFML/xxx/Joystick.hpp>

namespace sf {

namespace priv {

void Joystick::Initialize(unsigned int Index) {
    // Reset the joystick state
}
Initialize the Index-th available joystick

JoystickState Joystick::UpdateState()
{
    // Fill a JoystickState instance with the current joystick state
}

unsigned int Joystick::GetAxesCount() const
{
    // Return number of supported axes
}

unsigned int Joystick::GetButtonsCount() const
{
    // Return number of supported buttons
}

} // namespace priv

} // namespace sf
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.

// 3. This notice may not be removed or altered.

// Headers
#define _WIN32_WINDOWS 0x0501
#define _WIN32_WINNT 0x0501
#include <SFML/Window/Joystick.hpp>
#include <windows.h>
#include <mmsystem.h>

namespace sf {

namespace priv {
void Joystick::Initialize(unsigned int Index) {
  // Reset state
  myIndex = JOYSTICKID1;
  myNbButtons = 0;
  myIsConnected = false;
  myHasContinuousPOV = false;
  for (int i = 0; i < Joy::AxisCount; ++i)
    myAxes[i] = false;

  // Get the Index-th connected joystick
  MMRESULT Error;
  JOYINFOEX JoyInfo;
  JoyInfo.dwSize = sizeof(JoyInfo);
  JoyInfo.dwFlags = JOY_RETURNALL;
  for (unsigned int NbFound = 0; (Error = joyGetPosEx(myIndex, &JoyInfo)) != JOYERR_PARMS; myIndex++) {
    // Check if the current joystick is connected
    if (Error == JOYERR_NOERROR) {
      // Check if it's the required index
      if (NbFound == Index) {
        // Ok : store its parameters and return
        myIsConnected = true;
        JOYCAPS Caps;
        joyGetDevCaps(myIndex, &Caps, sizeof(Caps));
        myNbButtons = Caps.wNumButtons;
        if (myNbButtons > Joy::ButtonCount)
          myNbButtons = Joy::ButtonCount;
        myAxes[Joy::AxisX] = true;
        myAxes[Joy::AxisY] = true;
        myAxes[Joy::AxisZ] = (Caps.wCaps & JOYCAPS_HASZ) != 0;
        myAxes[Joy::AxisR] = (Caps.wCaps & JOYCAPS_HASR) != 0;
        myAxes[Joy::AxisU] = (Caps.wCaps & JOYCAPS_HASU) != 0;
      }
    }
  }
}
myAxes[Joy::AxisV] = (Caps.wCaps & JOYCAPS_HASV) != 0;
myAxes[Joy::AxisPOV] = (Caps.wCaps & JOYCAPS_HASPOV) != 0;
myHasContinuousPOV = (Caps.wCaps & JOYCAPS_POVCTS) != 0;

return;

// Go to the next valid joystick
++NbFound;

JoystickState Joystick::UpdateState()
{
    JoystickState State;
    if (myIsConnected)
    {
        // Get the joystick caps (for range conversion)
        JOYCAPS Caps;
        if (joyGetDevCaps(myIndex, &Caps, sizeof(Caps)) != JOYERR_NOERROR)
        {
            // Get the current joystick state
            JOYINFOEX Pos;
            Pos.dwFlags = JOY_RETURNX | JOY_RETURNY | JOY_RETURNZ | JOY_RETURNR | JOY_RETURNU | JOY_RETURNV |
            myHasContinuousPOV ? JOY_RETURNPOVCTS : JOY_RETURNPOV;
            Pos.dwSize = sizeof(Pos);
            if (joyGetPosEx(myIndex, &Pos) == JOYERR_NOERROR)
            {
                // Axes
                State.Axis[Joy::AxisX] = (Pos.dwXpos - (Caps.wXmax + Caps.wXmin) / 2.f) * 200.f / (Caps.wXmax - Caps.wXmin);
                State.Axis[Joy::AxisY] = (Pos.dwYpos - (Caps.wYmax + Caps.wYmin) / 2.f) * 200.f / (Caps.wYmax - Caps.wYmin);
                State.Axis[Joy::AxisZ] = (Pos.dwZpos - (Caps.wZmax + Caps.wZmin) / 2.f) * 200.f / (Caps.wZmax - Caps.wZmin);
                State.Axis[Joy::AxisR] = (Pos.dwRpos - (Caps.wRmax + Caps.wRmin) / 2.f) * 200.f / (Caps.wRmax - Caps.wRmin);
                State.Axis[Joy::AxisU] = (Pos.dwUpos - (Caps.wUmax + Caps.wUmin) / 2.f) * 200.f / (Caps.wUmax - Caps.wUmin);
            }
        }
    }
    return State;
}
State.Axis[Joy::AxisV] = (Pos.dwVpos - (Caps.wVmax + Caps.wVmin) / 2.f) * 200.f / (Caps.wVmax - Caps.wVmin);

// POV
if (Pos.dwPOV != 0xFFFF)
    State.Axis[Joy::AxisPOV] = Pos.dwPOV / 100.f;
else
    State.Axis[Joy::AxisPOV] = -1.f;

// Buttons
for (unsigned int i = 0; i < GetButtonsCount(); ++i)
    State.Buttons[i] = (Pos.dwButtons & (1 << i)) != 0;

return State;

bool Joystick::HasAxis(Joy::Axis Axis) const
{
    return myAxes[Axis];
}

unsigned int Joystick::GetButtonsCount() const
{
    return myNbButtons;
}

} // namespace priv
} // namespace sf
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any
// In no event will the authors be held liable for
// Permission is granted to anyone to use this software
// including commercial applications, and to alter it
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
// you must not claim that you wrote the original software.
// If you use this software in a product, an
// in the product documentation would be appreciated.

// 2. Altered source versions must be plainly
// and must not be misrepresented as being
// 3. This notice may not be removed or altered.

#ifndef SFML_JOYSTICKCARBON_HPP
#define SFML_JOYSTICKCARBON_HPP

namespace sf
{
namespace priv
{

class Joystick
{  
public:
  void Initialize(unsigned int Index);
  JoystickState UpdateState();
  bool HasAxis(Joy::Axis Axis) const;
  unsigned int GetButtonsCount() const;
};

// namespace priv

// namespace sf

#endif // SFML_JOYSTICKCARBON_HPP
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.

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

#ifndef SFML_JOYSTICK_HPP
#define SFML_JOYSTICK_HPP

// Headers
#include <SFML/Config.hpp>
#include <SFML/Window/Event.hpp>

namespace sf {
namespace priv {
struct JoystickState
{
    JoystickState()
    {
        // Default value for axes
        for (int i = 0; i < Joystick::AxisCount; ++i)
            Axis[i] = 0.f;
        Axis[Joystick::AxisPOV] = -1.f;

        // Default value for buttons
        for (int i = 0; i < Joystick::ButtonCount; ++i)
            Buttons[i] = false;
    }

    float Axis[Joystick::AxisCount];
    bool Buttons[Joystick::ButtonCount];
};

// namespace priv

} // namespace sf

#if defined(SFML_SYSTEM_WINDOWS)
    #include <SFML/Window/Win32/Joystick.hpp>
#elif defined(SFML_SYSTEM_LINUX) || defined(SFML_SYSTEM_FREEBSD)
    #include <SFML/Window/Linux/Joystick.hpp>
#elif defined(SFML_SYSTEM_MACOS)
    #include <SFML/Window/Cocoa/Joystick.hpp>
#endif
#endif // SFML_JOYSTICK_HPP

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
Joystick.hpp

#ifndef SFML_JOYSTICKXXX_HPP
#define SFML_JOYSTICKXXX_HPP

// Headers
namespace sf {
namespace priv {

class Joystick

#endif // SFML_JOYSTICKXXX_HPP
#define SFML_JOYSTICKXXX_HPP
#define SFML_JOYSTICKXXX_HPP

// Headers
namespace sf {
namespace priv {

class Joystick

#endif // SFML_JOYSTICKXXX_HPP
#define SFML_JOYSTICKXXX_HPP
#define SFML_JOYSTICKXXX_HPP


public:

void Initialize(unsigned int Index);
JoystickState UpdateState();
unsigned int GetAxesCount() const;
unsigned int GetButtonsCount() const;

};

} // namespace priv

} // namespace sf

#endif // SFML_JOYSTICKXXX_HPP

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documention generated by doxygen 1.5.2 ::
<table>
<thead>
<tr>
<th>Main Page</th>
<th>Namespaces</th>
<th>Classes</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>File List</strong></td>
</tr>
</tbody>
</table>
namespace sf
{
namespace priv
{
class Joystick
}
public:
void Initialize(unsigned int Index);
JoystickState UpdateState();
bool HasAxis(Joy::Axis Axis) const;
unsigned int GetButtonsCount() const;
private:
// Member data
bool myIsConnected;
unsigned int myIndex;
unsigned int myNbButtons;
bool myAxes[Joy::AxisCount];
bool myHasContinuousPOV;
};

} // namespace priv
} // namespace sf

#endif // SFML_JOYSTICKWIN32_HPP

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
// SFGE - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gomila@gmail.com)
// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.
// Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:
// 1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.
// 2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.
// 3. This notice may not be removed or altered from any source distribution.

inline Matrix3::Matrix3() {
    myData[0] = 1.f; myData[4] = 0.f; myData[8] = 0.f; myData[12] = 0.f;
    myData[1] = 0.f; myData[5] = 1.f; myData[9] = 0.f; myData[13] = 0.f;
    myData[2] = 0.f; myData[6] = 0.f; myData[10] = 1.f; myData[14] = 0.f;
}

inline Matrix3::Matrix3(float a00, float a01, float a02, float a10, float a11, float a12, float a20, float a21, float a22) {
    myData[0] = a00; myData[1] = a01; myData[2] = a02; myData[3] = 0.f;
    myData[12] = 0.f; myData[13] = 0.f; myData[14] = 0.f; myData[15] = 1.f;
}
float a10, float a11, float a20, float a21, float a22;
{
    myData[0] = a00; myData[4] = a01; myData[8] = a02;
    myData[2] = 0.f; myData[6] = 0.f; myData[10] = 0.f;
}

inline void Matrix3::SetFromTransformations(const
{
    float Angle = Rotation * 3.141592654f / 180.f;
    float Cos = static_cast<float>(cos(Angle));
    float Sin = static_cast<float>(sin(Angle));
    float SxCos = Scale.x * Cos;
    float SyCos = Scale.y * Cos;
    float SxSin = Scale.x * Sin;
    float SySin = Scale.y * Sin;
    float Tx = -Center.x * SxCos - Center.y * SySin + Translation.x;
    float Ty = Center.x * SxSin - Center.y * SyCos + Translation.y;
    myData[0] = SxCos; myData[4] = SySin; myData[8] = Tx;
    myData[2] = 0.f; myData[6] = 0.f; myData[10] = 1.f;
    myData[3] = 0.f; myData[7] = 0.f; myData[11] = 0.f;
}

inline Vector2f Matrix3::Transform(const Vector2f &Point) {
    return Vector2f(myData[0] * Point.x + myData[4] * Point.y + myData[12],
}

inline Matrix3 Matrix3::GetInverse() const
```
00088 { // Compute the determinant
00093 // Compute the inverse if determinant is not zero
00094     if ((Det < -1E-7f) || (Det > 1E-7f))
00095         {
00101                     -(myData[13] * myData[0] - myData[1] * myData[12]) / Det,
00104                     (myData[5] * myData[0] - myData[1] * myData[4]) / Det);
00106     }
00107   else
00108       {
00109           return Identity;
00110       }
00111 }
00112
00113 inline const float* Matrix3::Get4x4Elements() const
00114 {
00115     return myData;
00116 }
00118 inline float Matrix3::operator()(unsigned int Row, unsigned int Col)
00119 {
00120     switch (Row + Col * 3)
00121     {
00122         case 0 : return myData[0];
00123         case 1 : return myData[1];
00124         case 2 : return myData[2];
00125         case 3 : return myData[3];
00126         case 4 : return myData[4];
00127         case 5 : return myData[5];
00128         case 6 : return myData[6];
00129         case 7 : return myData[7];
00130         case 8 : return myData[8];
00131     }
case 1 : return myData[1];
case 2 : return myData[3];
case 3 : return myData[4];
case 4 : return myData[5];
case 5 : return myData[7];
case 6 : return myData[12];
case 7 : return myData[13];
case 8 : return myData[15];

default : return myData[0];
}

inline float& Matrix3::operator[](unsigned int Row)
{
    switch (Row + Col * 3)
    {
        case 0 : return myData[0];
        case 1 : return myData[1];
        case 2 : return myData[3];
        case 3 : return myData[4];
        case 4 : return myData[5];
        case 5 : return myData[7];
        case 6 : return myData[12];
        case 7 : return myData[13];
        case 8 : return myData[15];
        default : return myData[0];
    }
}

inline Matrix3 Matrix3::operator*(const Matrix3& Mat)
{
}
inline Matrix3& Matrix3::operator *=(const Matrix3& Mat) {
    return *this = *this * Mat;
}
Mutex.hpp

00001 //
00002 // SFML - Simple and Fast Multimedia Library
00003 // Copyright (C) 2007-2009 Laurent Gomila (laurent.gomila@gmail.com)
00004 //
00005 // This software is provided 'as-is', without any
00006 // In no event will the authors be held liable for
00007 // Permission is granted to anyone to use this software
00008 // Permission is granted to anyone to use this software
00009 // including commercial applications, and to alter
00010 // subject to the following restrictions:
00011 // 1. The origin of this software must not be
00012 // 1. The origin of this software must not be
00013 // you must not claim that you wrote the original
00014 // If you use this software in a product, and
00015 // in the product documentation would be appreciated.
00016 // 2. Altered source versions must be plainly
00017 // and must not be misrepresented as being the
00018 // 2. Altered source versions must be plainly
00019 // 3. This notice may not be removed or altered
00020 // 3. This notice may not be removed or altered
00021 //
00022 #ifndef SFML_MUTEX_HPP
00023 #define SFML_MUTEX_HPP
00024 #ifdef SFML_SYSTEM_WINDOWS
00025 #ifdef SFML_SYSTEM_WINDOWS
00026 #include <SFML/System/Win32/Mutex.hpp>
00027 #include <SFML/System/Win32/Mutex.hpp>
00028 // Headers
00029 #include <SFML/Config.hpp>
00030 #include <SFML/Config.hpp>
00031 #ifndef SFML_MUTEX_HPP
00032 #define SFML_MUTEX_HPP
00033 #ifndef SFML_MUTEX_HPP
00034 #ifdef SFML_SYSTEM_WINDOWS
00035 #ifdef SFML_SYSTEM_WINDOWS
00036 #include <SFML/System/Win32/Mutex.hpp>
00037 #include <SFML/System/Win32/Mutex.hpp>
#else
#include <SFML/System/Unix/Mutex.hpp>
#endif
#endif // SFML_MUTEX_HPP

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any
// express or implied warranty. In no event will the
// authors be held liable for any damages arising
// from the use of this software.

// Permission is granted to anyone to use this software
// for any purpose, including commercial applications,
// and to alter it and redistribute it freely, subject to
// the following restrictions:

// 1. The origin of this software must not be misrepresen-
// you must not claim that you wrote the origin.
// If you use this software in a product, an
// in the product documentation would be appreciated.

// 2. Altered source versions must be plainly marked
// and must not be misrepresented as being the
// 3. This notice may not be removed or altered.

#ifndef SFML_NETWORK_HPP
#define SFML_NETWORK_HPP

// Headers

#include <SFML/System.hpp>
#include <SFML/Network/Ftp.hpp>
#include <SFML/Network/Http.hpp>
#include <SFML/Network/IPAddress.hpp>
#include <SFML/Network/Packet.hpp>
#include <SFML/Network/Selector.hpp>
#endif
```cpp
#include <SFML/Network/SocketTCP.hpp>
#include <SFML/Network/SocketUDP.hpp>

#endif // SFML_NETWORK_HPP
```
OpenAL.hpp

#ifndef SFML_OPENAL_HPP
#define SFML_OPENAL_HPP

// Headers
#include <SFML/Config.hpp>

#if defined(SFML_SYSTEM_MACOS)
#include <OpenAL/al.h>
#include <OpenAL/alc.h>
#else
#include <AL/al.h>
#endif

// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
// you must not claim that you wrote the original software.
// If you use this software in a product, an acknowledgment
// in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
// and must not be misrepresented as being the original software.

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

#endif
```cpp
#include <AL/alc.h>
#endif

#include <iostream>
#include <string>

namespace sf
{
    namespace priv
    {
#ifdef SFML_DEBUG

    // If in debug mode, perform a test on every call
    #define ALCheck(Func) ((Func), priv::ALCheckError(__FILE__, __LINE__))

#else

    // Else, we don't add any overhead
    #define ALCheck(Func) (Func)

#endif

    #endif

    inline void ALCheckError(const std::string& File)
    {
        // Get the last error
        ALenum ErrorCode = alGetError();

        if (ErrorCode != AL_NO_ERROR)
        {
            std::string Error, Desc;

            // Decode the error code
            switch (ErrorCode)
            {
            case AL_INVALID_NAME:  
```
case AL_INVALID_NAME :
{
    Error = "AL_INVALID_NAME";
    Desc = "an unacceptable name has been specified";
    break;
}

case AL_INVALID_ENUM :
{
    Error = "AL_INVALID_ENUM";
    Desc = "an unacceptable value has been specified for an enumerated argument";
    break;
}

case AL_INVALID_VALUE :
{
    Error = "AL_INVALID_VALUE";
    Desc = "a numeric argument is out of range";
    break;
}

case AL_INVALID_OPERATION :
{
    Error = "AL_INVALID_OPERATION";
    Desc = "the specified operation is not allowed in the current state";
    break;
}

case AL_OUT_OF_MEMORY :
{
    Error = "AL_OUT_OF_MEMORY";
    Desc = "there is not enough memory left to execute the command";
    break;
}

// Log the error
std::cerr << "An internal OpenAL call failed in" 

<< File.substr(File.find_last_of(
<< Error << ", " << Desc
<< std::endl;
}
}
} // namespace priv
}
} // namespace sf

#endif // SFML_OPENAL_HPP

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.

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

#ifndef SFML_OPENGL_HPP
#define SFML_OPENGL_HPP

#include <SFML/Config.hpp>

#if defined(SFML_SYSTEM_WINDOWS)
#include <windows.h>
#include <GL/gl.h>
#endif
#include <GL/glu.h>

#elif defined(SFML_SYSTEM_LINUX) || defined(SFML_SYSTEM_FREEBSD)
#include <GL/gl.h>
#include <GL/glu.h>

#elif defined(SFML_SYSTEM_MACOS)
#include <OpenGL/gl.h>
#include <OpenGL/glu.h>

#endif

#endif // SFML_OPENGL_HPP
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any
// express or implied warranty. In no event will the authors be held liable for
// any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an
//    acknowledgment in the product documentation would be appreciated but
//    is not required.

// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.

// 3. This notice may not be removed or altered.

// Headers
#include <SFML/System/Win32/Platform.hpp>
#include <windows.h>

namespace sf {
    namespace priv {
        double Platform::GetSystemTime() {
            // Platform-specific implementation
        }
    }
}
static LARGE_INTEGER Frequency;
static BOOL UseHighPerformanceTimer = QueryPerformanceFrequency(&Frequency);

if (UseHighPerformanceTimer) {
    // High performance counter available :
    LARGE_INTEGER CurrentTime;
    QueryPerformanceCounter(&CurrentTime);
    return static_cast<double>(CurrentTime.QuadPart) / Frequency.QuadPart;
} else {
    // High performance counter not available :
    return GetTickCount() * 0.001;
}

void Platform::Sleep(float Time) {
    ::Sleep(static_cast<DWORD>(Time * 1000));
}

} // namespace priv

} // namespace sf

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gomila@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
// you must not claim that you wrote the original software.
// If you use this software in a product, an acknowledgment
// in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
// and must not be misrepresented as being the original software.

// 3. This notice may not be removed or altered.

#ifndef SFML_PLATFORM_HPP
#define SFML_PLATFORM_HPP

// Headers
#include <SFML/Config.hpp>

#if defined(SFML_SYSTEM_WINDOWS)

#include <SFML/System/Win32/Platform.hpp>

#endif

#endif
#elif defined(SFML_SYSTEM_LINUX) || defined(SFML_SYSTEM_MACOS) || defined(SFML_SYSTEM_FREEBSD)

#include <SFML/System/Unix/Platform.hpp>

#endif

// SFML_PLATFORM_HPP
```cpp
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
// you must not claim that you wrote the original software.
// If you use this software in a product, an acknowledgment
// in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
// and must not be misrepresented as being the original software.

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

#ifndef SFML_PLATFORMWIN32_HPP
#define SFML_PLATFORMWIN32_HPP

// Headers
#include <SFML/Config.hpp>
#include <vector>

namespace sf {

namespace priv {
```
class Platform {
    public:
    static double GetSystemTime();
    static void Sleep(float Time);
};

// namespace priv

// namespace sf

#endif // SFML_PLATFORMWIN32_HPP
Rect.inl

template <typename T>
Rect<T>::Rect():
Left(0), Top(0), Right(0), Bottom(0) {
}
template <typename T>
Rect<T>::Rect(T LeftCoord, T TopCoord, T RightCoord, T BottomCoord) :
Left (LeftCoord),
Top (TopCoord),
Right (RightCoord),
Bottom(BottomCoord)
{
}

template <typename T>
T Rect<T>::GetWidth() const
{
return Right - Left;
}

template <typename T>
T Rect<T>::GetHeight() const
{
return Bottom - Top;
}

void Rect<T>::Offset(T OffsetX, T OffsetY)
{
Left += OffsetX;
Right += OffsetX;
Top += OffsetY;
Bottom += OffsetY;
}
bool Rect<T>::Contains(T X, T Y) const
{
    return (X >= Left) && (X <= Right) && (Y >= Top) && (Y <= Bottom);
}

template<typename T>
bool Rect<T>::Intersects(const Rect<T>& Rectangle, Rect<T>* OverlappingRect)
{
    // Compute overlapping rect
    Rect Overlapping(std::max(Left, Rectangle.Left),
                     std::max(Top, Rectangle.Top),
                     std::min(Right, Rectangle.Right),
                     std::min(Bottom, Rectangle.Bottom));

    // If overlapping rect is valid, then there
    if ((Overlapping.Left < Overlapping.Right) &&
        (Overlapping.Top < Overlapping.Bottom))
        {
        if (OverlappingRect)
            *OverlappingRect = Overlapping;
        return true;
        }
    else
    {
        if (OverlappingRect)
            *OverlappingRect = Rect(0, 0, 0, 0);
        return false;
    }
}
template <typename T> 
Resource<T>::Resource() 
{ 
    // Nothing to do 
} 

template <typename T> 
Resource<T>::Resource(const Resource<T>&) 
{
template <typename T>
Resource<T>::~Resource()
{
    // Notify all observers
    for (typename std::set<ResourcePtr<T>*>::iterator i = myObservers.begin(); i != myObservers.end(); ++i)
    {
        (*i)->OnResourceDestroyed();
    }
}

template <typename T>
Resource<T>& Resource<T>::operator=(const Resource<T>&)
{
    // Nothing to do, we don't want to copy observers
    return *this;
}

template <typename T>
void Resource<T>::Connect(ResourcePtr<T>& Observer)
{
    myObservers.insert(&Observer);
}

template <typename T>
void Resource<T>::Disconnect(ResourcePtr<T>& Observer)
{
    myObservers.erase(&Observer);
}
### File List

- **Main Page**
- **Namespaces**
- **Classes**
- **Files**
ResourcePtr.inl

template <typename T>
ResourcePtr<T>::ResourcePtr(): myResource(NULL) {
}

template <typename T>
ResourcePtr<T>::ResourcePtr(const T* Resource) {
```cpp
myResource(Resource) {
    if (myResource)
        myResource->Connect(*this);
}

template <typename T>
ResourcePtr<T>::ResourcePtr(const ResourcePtr<T>& Copy) :
    myResource(Copy.myResource)
{
    if (myResource)
        myResource->Connect(*this);
}

template <typename T>
ResourcePtr<T>::~ResourcePtr() :
{
    if (myResource)
        myResource->Disconnect(*this);
}

template <typename T>
ResourcePtr<T>& ResourcePtr<T>::operator =(const
{
    if (myResource)
        myResource->Disconnect(*this);
    myResource = Other.myResource;
    if (myResource)
        myResource->Connect(*this);
    return *this;
}
template <typename T>
ResourcePtr<T>& ResourcePtr<T>::operator =(const T& Resource) {
    if (myResource)
        myResource->Disconnect(*this);
    myResource = Resource;
    if (myResource)
        myResource->Connect(*this);
    return *this;
}

template <typename T>
ResourcePtr<T>::operator const T*() const {
    return myResource;
}

template <typename T>
const T& ResourcePtr<T>::operator *() const {
    return *myResource;
}

template <typename T>
const T* ResourcePtr<T>::operator ->() const {
    return myResource;
}
template <typename T>
void ResourcePtr<T>::OnResourceDestroyed()
{
    myResource = NULL;
}
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)
//
// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.
//
// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:
//
// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is not required.
//
// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.
//
// 3. This notice may not be removed or altered from any source distribution.

template<typename Type>
void Selector<Type>::Add(Type Socket)
{
    if (Socket.IsValid())
    {
        SelectorBase::Add(Socket.mySocket);
        mySockets[Socket.mySocket] = Socket;
    }
}
template <typename Type>
void Selector<Type>::Remove(Type Socket) {
    typename SocketTable::iterator It = mySockets.find(Socket.mySocket);
    if (It != mySockets.end()) {
        SelectorBase::Remove(Socket.mySocket);
        mySockets.erase(It);
    }
}

template <typename Type>
void Selector<Type>::Clear() {
    SelectorBase::Clear();
    mySockets.clear();
}

template <typename Type>
unsigned int Selector<Type>::Wait(float Timeout) {
    if (mySockets.empty()) return 0;
    return SelectorBase::Wait(Timeout);
}

template <typename Type>
Type Selector<Type>::GetSocketReady(unsigned int) {
    SocketHelper::SocketType Socket = SelectorBase::GetSocketReady();
    return Socket;
}
typename SocketTable::const_iterator It = mySockets.find(Socket);
if (It != mySockets.end())
    return It->second;
else
    return Type(Socket);
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any
// express or implied warranty. In no event will the authors be held liable for
// any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
// you must not claim that you wrote the original software.
// If you use this software in a product, an
// acknowledgment in the product documentation would be appreciated but
// is not required.

// 2. Altered source versions must be plainly marked as such,
// and must not be misrepresented as being the
// original software.

// 3. This notice may not be removed or altered.

// Windows specific : defines the WinMain entry
// so that developers can use the standard main
// even in a Win32 Application project, and keep

#ifndef _WIN32

#include <windows.h>

extern int main(int argc, char* argv[]);
#endif
00038 int WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, INT)
00039 {
00040     return main(__argc, __argv);
00041 }
00042 #endif // _WIN32
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)
// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.
// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:
// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is not required.
// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.
// 3. This notice may not be removed or altered from any source distribution.

// Headers
#include <SFML/System/Sleep.hpp>
#include <SFML/System/Platform.hpp>

namespace sf
{
    void Sleep(float Duration)
    {
        if (Duration >= 0)
            priv::Platform::Sleep(Duration);
    }
}
} // namespace sf

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gomila@gmail.com)
// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.
// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:
// 1. The origin of this software must not be misrepresented;
// you must not claim that you wrote the original software.
// If you use this software in a product, an acknowledgment
// in the product documentation would be appreciated but is not required.
// 2. Altered source versions must be plainly marked as such,
// and must not be misrepresented as being the original software.
// 3. This notice may not be removed or altered.

#ifndef SFML_SLEEP_HPP
#define SFML_SLEEP_HPP

// Headers
#include <SFML/Config.hpp>

namespace sf {

void SFML_API Sleep(float Duration);

} // namespace sf
#endif // SFML_SLEEP_HPP
} // namespace sf

#endif // SFML_SLEEP_HPP

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
<table>
<thead>
<tr>
<th>Main Page</th>
<th>Namespaces</th>
<th>Classes</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>File List</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gomila@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
// you must not claim that you wrote the original software.
// If you use this software in a product, an acknowledgment
// in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
// and must not be misrepresented as being the original software.

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

#ifndef SFML_SOCKETHELPER_HPP
#define SFML_SOCKETHELPER_HPP

// Headers
#include <SFML/Config.hpp>

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

// Headers
#include <SFML/Config.hpp>

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {

namespace sf {
namespace Socket {
```
enum Status {
    Done,
    NotReady,
    Disconnected,
    Error
};
```

```cpp
#include <SFML/Network/Win32/SocketHelper.hpp>
#endif
#include <SFML/Network/Unix/SocketHelper.hpp>
#endif // SFML_SOCKETHELPER_HPP
```

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.

// 3. This notice may not be removed or altered.

#ifndef SFML_SOCKETS_HPP
#define SFML_SOCKETS_HPP

#include <SFML/Config.hpp>

#ifdef SFML_SYSTEM_WINDOWS
#include <SFML/Network/Win32/Sockets.hpp>
#endif

#endif // SFML_SOCKETS_HPP
#else
#include <SFML/Network/Unix/Sockets.hpp>
#endif
#endif // SFML_SOCKETS_HPP
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.

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

// Headers
#include <SFML/Audio/SoundFile.hpp>
#include <SFML/Audio/SoundFileDefault.hpp>
#include <SFML/Audio/SoundFileOgg.hpp>
#include <iostream>

namespace sf {

namespace priv {

} // namespace priv

} // namespace sf
SoundFile* SoundFile::CreateRead(const std::string& Filename)
{
    // Create the file according to its type
    SoundFile* File = NULL;
    if (SoundFileOgg::IsFileSupported(Filename, File))
    else if (SoundFileDefault::IsFileSupported(Filename, File))

    // Open it for reading
    if (File)
    {
        std::size_t SamplesCount;
        unsigned int ChannelsCount;
        unsigned int SampleRate;
        if (File->OpenRead(Filename, SamplesCount, ChannelsCount, SampleRate))
            File->myFilename = Filename;
            File->myData = NULL;
            File->mySize = 0;
            File->myNbSamples = SamplesCount;
            File->myChannelsCount = ChannelsCount;
            File->mySampleRate = SampleRate;
        }
        else
        {
            delete File;
            File = NULL;
        }
    }
    return File;
}

SoundFile* SoundFile::CreateRead(const char* Data, std::size_t SizeInMemory)
{
    // Create the file according to its type
SoundFile* File = NULL;
if (SoundFileOgg::IsFileSupported(Data, SizeInMemory))
else if (SoundFileDefault::IsFileSupported(Data, SizeInMemory))
    // Open it for reading
    if (File)
        {
            std::size_t SamplesCount;
            unsigned int ChannelsCount;
            unsigned int SampleRate;
            if (File->OpenRead(Data, SizeInMemory, &SamplesCount, &ChannelsCount, &SampleRate))
                {
                    File->myFilename = "";
                    File->myData = Data;
                    File->mySize = SizeInMemory;
                    File->myNbSamples = SamplesCount;
                    File->myChannelsCount = ChannelsCount;
                    File->mySampleRate = SampleRate;
                }
            else
                {
                    delete File;
                    File = NULL;
                }
        }
    return File;
}

SoundFile* SoundFile::CreateWrite(const std::string& Filename, {
    // Create the file according to its type
    SoundFile* File = NULL;
    if (SoundFileOgg::IsFileSupported(Filename, SizeInMemory))
        else if (SoundFileDefault::IsFileSupported(Filename, SizeInMemory))
            // Local parameters
// Open it for writing

if (File)
{
    if (File->OpenWrite(Filename, ChannelsCount, SampleRate))
    {
        File->myFilename = "";
        File->myData = NULL;
        File->mySize = 0;
        File->myNbSamples = 0;
        File->myChannelsCount = ChannelsCount;
        File->mySampleRate = SampleRate;
    }
    else
    {
        delete File;
        File = NULL;
    }
}
return File;

SoundFile::SoundFile()
{
    myNbSamples = 0,
    myChannelsCount = 0,
    mySampleRate = 0
}

SoundFile::~SoundFile()
{
    // Nothing to do
}
std::size_t SoundFile::GetSamplesCount() const
{
    return myNbSamples;
}

unsigned int SoundFile::GetChannelsCount() const
{
    return myChannelsCount;
}

unsigned int SoundFile::GetSampleRate() const
{
    return mySampleRate;
}

bool SoundFile::Restart()
{
    if (myData)
    {
        // Reopen from memory
        return OpenRead(myData, mySize, myNbSamples,
    }
    else if (myFilename != "")
    {
        // Reopen from file
        return OpenRead(myFilename, myNbSamples,
    }
    else
    {
        // Trying to reopen a file opened in write mode...
        std::cerr << "Warning : trying to restart a file opened in write mode...
        return false;
    }
bool SoundFile::OpenRead(const std::string& Filename, std::size_t&)
{
    std::cerr << "Failed to open sound file ";
    return false;
}

bool SoundFile::OpenRead(const char*, std::size_t, std::size_t&)
{
    std::cerr << "Failed to open sound file from memory, format is not supported by SFML";
    return false;
}

bool SoundFile::OpenWrite(const std::string& Filename)
{
    std::cerr << "Failed to open sound file ";
    return false;
}

std::size_t SoundFile::Read(Int16*, std::size_t)
{
    std::cerr << "Failed to read from sound file";
    return 0;
}

void SoundFile::Write(const Int16*, std::size_t)
{ }
    std::cerr << "Failed to write to sound file"
} // namespace priv

} // namespace sf
Simple and Fast Multimedia Library

Main Page  Namespaces  Classes  Files

File List
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)
//
// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.
//
// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:
//
// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an
//    acknowledgment in the product documentation would be appreciated but is not required.
// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.
// 3. This notice may not be removed or altered from any source distribution.

#ifndef SFML_SOUNDFILE_HPP
#define SFML_SOUNDFILE_HPP

namespace sf
{
    namespace priv
    {
...

#endif // SFML_SOUNDFILE_HPP

#include <SFML/System/NonCopyable.hpp>
#include <string>

namespace sf
{
    namespace priv
    {
...
```cpp
class SoundFile : NonCopyable {
public:
static SoundFile* CreateRead(const std::string& Filename);
static SoundFile* CreateRead(const char* Data, std::size_t SizeInBytes);
static SoundFile* CreateWrite(const std::string& Filename);

virtual ~SoundFile();
std::size_t GetSamplesCount() const;
unsigned int GetChannelsCount() const;
unsigned int GetSampleRate() const;
bool Restart();

virtual std::size_t Read(Int16* Data, std::size_t NbSamples);
virtual void Write(const Int16* Data, std::size_t NbSamples);

protected:
SoundFile();

private:
virtual bool OpenRead(const std::string& FileName, std::size_t& NbSamples);
virtu
// Member data
std::size_t myNbSamples;
unsigned int myChannelsCount;
unsigned int mySampleRate;
std::string myFilename;
const char* myData;
std::size_t mySize;
};

} // namespace priv

} // namespace sf

 ifndef // SFML_SOUNDFILE_HPP
endif // SFML_SOUNDFILE_HPP

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gomila@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.

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

// Headers
#include <SFML/Audio/SoundFileDefault.hpp>
#include <iostream>
#include <cstring>

namespace sf {
    namespace priv {
        SoundFileDefault::SoundFileDefault() :
        
    }
}

namespace sf {
    namespace priv {
        
    }

    
}

namespace sf {
    
}


myFile(NULL) {
    
    SoundFileDefault::~SoundFileDefault() {
        if (myFile)
            sf_close(myFile);
    }

    bool SoundFileDefault::IsFileSupported(const std::string& Filename) {
        if (Read)
            // Open the sound file
            SF_INFO FileInfos;
            SNDFILE* File = sf_open(Filename.c_str(), SFM_READ, &FileInfos);
            if (File)
                sf_close(File);
                return true;
            else
                return false;
        else
            // Check the extension
            return GetFormatFromFilename(Filename) != -1;
    }
}
bool SoundFileDefault::IsFileSupported(const char *filename) {
    // Define the I/O custom functions for reading from memory
    SF_VIRTUAL_IO VirtualIO;
    VirtualIO.get_filelen = &SoundFileDefault::MemoryGetLength;
    VirtualIO.read = &SoundFileDefault::MemoryRead;
    VirtualIO.seek = &SoundFileDefault::MemorySeek;
    VirtualIO.tell = &SoundFileDefault::MemoryTell;
    VirtualIO.write = &SoundFileDefault::MemoryWrite;

    // Initialize the memory data
    MemoryInfos Memory;
    Memory.DataStart = Data;
    Memory.DataPtr = Data;
    Memory.TotalSize = SizeInBytes;

    // Open the sound file
    SF_INFO FileInfos;
    SNDFILE* File = sf_open_virtual(&VirtualIO,

    if (File) {
      sf_close(File);
      return true;
    } else {
      return false;
    }

    // If the file is already opened, first close it
    bool SoundFileDefault::OpenRead(const std::string &filename) {
        // If the file is already opened, first close it
        // Define the I/O custom functions for reading from memory
        // Initialize the memory data
        // Open the sound file
        // Check if the file is opened
        // Close the file and return true
        // Return false if not opened
    }
}
if (myFile)
  sf_close(myFile);

// Open the sound file
SF_INFO FileInfos;
myFile = sf_open(Filename.c_str(), SFM_READ,
if (!myFile)
{
  std::cerr << "Failed to read sound file"
            << std::endl;
  return false;
}

// Set the sound parameters
ChannelsCount = FileInfos.channels;
SampleRate = FileInfos.samplerate;
NbSamples = static_cast<std::size_t>(FileInfos.frames) * ChannelsCount;

return true;

bool SoundFileDefault::OpenRead(const char* Data, std::size_t SizeInBytes, std::size_t& NbSamples)
{
// If the file is already opened, first close it
if (myFile)
  sf_close(myFile);

// Define the I/O custom functions for reading
SF_VIRTUAL_IO VirtualIO;
VirtualIO.get_filelen = &SoundFileDefault::MemoryGetLength;
VirtualIO.read = &SoundFileDefault::MemoryRead;
VirtualIO.seek = &SoundFileDefault::MemorySeek;
VirtualIO.tell = &SoundFileDefault::MemoryTell;
VirtualIO.write = &SoundFileDefault::MemoryWrite;

// Initialize the memory data
myMemory.DataStart = Data;
myMemory.DataPtr = Data;
myMemory.TotalSize = SizeInBytes;

// Open the sound file
SF_INFO FileInfos;
myFile = sf_open_virtual(&VirtualIO, SFM_READ, &FileInfos, &myMemory);
if (!myFile)
{
    std::cerr << "Failed to read sound file"
    return false;
}

// Set the sound parameters
ChannelsCount = FileInfos.channels;
SampleRate = FileInfos.samplerate;
NbSamples = static_cast<std::size_t>(FileInfos.frames) * ChannelsCount;
return true;

bool SoundFileDefault::OpenWrite(const std::string& Filename) {
    // If the file is already opened, first close it
    if (myFile)
        sf_close(myFile);

    // Find the right format according to the file extension
    int Format = GetFormatFromFilename(Filename);
    if (Format == -1)
    {
        // Error : unrecognized extension
        std::cerr << "Failed to create sound file"
        return false;
    }

    // Fill the sound infos with parameters
SF_INFO FileInfos;
FileInfos.channels = ChannelsCount;
FileInfos.samplerate = SampleRate;
FileInfos.format = Format | SF_FORMAT_PCM_16;

// Open the sound file for writing
myFile = sf_open(Filename.c_str(), SFM_WRITE, &FileInfos);
if (!myFile)
{
    std::cerr << "Failed to create sound file"
    return false;
}
return true;

std::size_t SoundFileDefault::Read(Int16* Data, std::size_t NbSamples)
{
    if (myFile && Data && NbSamples)
        return static_cast<std::size_t>(sf_read_short(myFile, Data, NbSamples));
    else
        return 0;
}

void SoundFileDefault::Write(const Int16* Data, std::size_t NbSamples)
{
    if (myFile && Data && NbSamples)
        sf_write_short(myFile, Data, NbSamples);
}

int SoundFileDefault::GetFormatFromFilename(const std::string& Filename)
{
    // Extract the extension
    std::string Ext = "wav";
std::string::size_type Pos = Filename.find_last_of(
  if (Pos != std::string::npos)
    Ext = Filename.substr(Pos + 1);

  if (Ext == "wav" || Ext == "WAV") return $;
  if (Ext == "aif" || Ext == "AIF") return $;
  if (Ext == "aiff" || Ext == "AIFF") return $;
  if (Ext == "au" || Ext == "AU") return $;
  if (Ext == "raw" || Ext == "RAW") return $;
  if (Ext == "paf" || Ext == "PAF") return $;
  if (Ext == "svx" || Ext == "SVX") return $;
  if (Ext == "voc" || Ext == "VOC") return $;
  if (Ext == "sf" || Ext == "SF") return $;
  if (Ext == "w64" || Ext == "W64") return $;
  if (Ext == "mat4" || Ext == "MAT4") return $;
  if (Ext == "mat5" || Ext == "MAT5") return $;
  if (Ext == "pvf" || Ext == "PVF") return $;
  if (Ext == "htk" || Ext == "HTK") return $;
  if (Ext == "caf" || Ext == "CAF") return $;
  if (Ext == "nist" || Ext == "NIST") return $;
  if (Ext == "sds" || Ext == "SDS") return $;
  if (Ext == "avr" || Ext == "AVR") return $;
  if (Ext == "sd2" || Ext == "SD2") return $;
  if (Ext == "flac" || Ext == "FLAC") return $;
  return -1;
}

sf_count_t SoundFileDefault::MemoryGetLength(void) {
  MemoryInfos* Memory = static_cast<
    return Memory->TotalSize;
}

sf_count_t SoundFileDefault::MemoryRead(void* Ptr,
MemoryInfos* Memory = static_cast<MemoryInfos*>(Memory);

sf_count_t Position = Memory->DataPtr - Memory->DataStart;

if (Position + Count >= Memory->TotalSize)
    Count = Memory->TotalSize - Position;

memcpy(Ptr, Memory->DataPtr, static_cast<std::size_t>(Count));

Memory->DataPtr += Count;

return Count;

}

sf_count_t SoundFileDefault::MemorySeek(sf_count_t Offset, {
    MemoryInfos* Memory = static_cast<MemoryInfos*>(Memory);
    sf_count_t Position = 0;
    switch (Whence)
    {
    case SEEK_SET :
        Position = Offset;
        break;
    case SEEK_CUR :
        Position = Memory->DataPtr - Memory->DataStart;
        break;
    case SEEK_END :
        Position = Memory->TotalSize - Offset;
        break;
    default :
        Position = 0;
        break;
    }

    if (Position >= Memory->TotalSize)
        Position = Memory->TotalSize - 1;
    else if (Position < 0)
Position = 0;
Memory->DataPtr = Memory->DataStart + Position;
return Position;
}
sf_count_t SoundFileDefault::MemoryTell(void* UserData)
{
    MemoryInfos* Memory = static_cast<MemoryInfos*>(UserData);
    return Memory->DataPtr - Memory->DataStart;
}
sf_count_t SoundFileDefault::MemoryWrite(const void *Data, sf_count_t DataSize)
{
    return 0;
}

} // namespace priv
{// namespace sf
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gomila@gmail.com)
//
// This software is provided 'as-is', without any implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.
//
// Permission is granted to anyone to use this software for any purpose, including commercial applications,
// subject to the following restrictions:
// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is not required.
// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.
// 3. This notice may not be removed or altered.

#ifndef SFML_SOUNDFILEDEFAULT_HPP
#define SFML_SOUNDFILEDEFAULT_HPP

// Headers
#include <SFML/Audio/SoundFile.hpp>
#include <sndfile.h>

namespace sf {

namespace priv


```cpp
class SoundFileDefault : public SoundFile {
public:
  SoundFileDefault();
  ~SoundFileDefault();

  static bool IsFileSupported(const std::string& Filename);
  static bool IsFileSupported(const char* Data);

  virtual std::size_t Read(Int16* Data, std::size_t NbSamples);
  virtual void Write(const Int16* Data, std::size_t NbSamples);

private:
  virtual bool OpenRead(const std::string& Filename, std::size_t& NbSamples);
  virtual bool OpenRead(const char* Data, std::size_t SizeInBytes, std::size_t& NbSamples);
  virtual bool OpenWrite(const std::string& Filename);

  static int GetFormatFromFilename(const std::string& Filename);
  static sf_count_t MemoryGetLength(void* UserData);
  static sf_count_t MemoryRead(void* Ptr, sf_count_t Count);
  static sf_count_t MemorySeek(sf_count_t Offset);
  static sf_count_t MemoryTell(void* UserData);
  static sf_count_t MemoryWrite(const void* Ptr, sf_count_t Count);

  struct MemoryInfos {
    const char* DataStart;
    const char* DataPtr;
  };
};
```
sf_count_t TotalSize;
};

// Member data
SNDFILE* myFile;
MemoryInfos myMemory;
};

} // namespace priv

} // namespace sf

#endif // SFML_SOUNDFILEDEFAULT_HPP
SFML - Simple and Fast Multimedia Library
Copyright (C) 2007-2009 Laurent Gomila (laurent.gomila@gmail.com)

This software is provided 'as-is', without any express or implied warranty.
In no event will the authors be held liable for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software.
   If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.

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

Headers
#include <SFML/Audio/SoundFileOgg.hpp>
#include <SFML/Audio/stb_vorbis/stb_vorbis.h>
#include <iostream>

namespace sf
{
namespace priv
{
SoundFileOgg::SoundFileOgg();
}
SoundFileOgg::~SoundFileOgg()
{
    if (myStream)
        stb_vorbis_close(myStream);
}

bool SoundFileOgg::IsFileSupported(const std::string& Filename)
{
    if (Read)
    {
        // Open the vorbis stream
        stb_vorbis* Stream = stb_vorbis_open_filename(const_cast<char*>(Filename.c_str()), NULL, NULL);
        if (Stream)
            stb_vorbis_close(Stream);
        return true;
    }
    else
    {
        // No support for writing ogg files yet...
        return false;
    }
}

bool SoundFileOgg::IsFileSupported(const char* Data, std::size_t SizeInBytes) {
    // Open the vorbis stream
    unsigned char* Buffer = reinterpret_cast<unsigned char*>(static_cast<int>(SizeInBytes));
    stb_vorbis* Stream = stb_vorbis_open_memory(Buffer, static_cast<int>(SizeInBytes), NULL, NULL);
    if (Stream)
        stb_vorbis_close(Stream);
    return true;
    else
    { return false; }
}

bool SoundFileOgg::OpenRead(const std::string& Filename, std::size_t& NbSamples) {
    // Close the file if already opened
    if (myStream)
        stb_vorbis_close(myStream);
    myStream = stb_vorbis_open_filename(const_cast<char*>(Filename.c_str()), NULL, NULL);
    if (myStream == NULL)
    { std::cerr << "Failed to read sound file";
      return false;
    }
    // Get the music parameters
    stb_vorbis_info Infos = stb_vorbis_get_info(myStream);
ChannelsCount = myChannelsCount = Infos.channels;
SampleRate = Infos.sample_rate;
NbSamples = static_cast<std::size_t>(stb_vorbis_stream_length_in_samples(myStream) * ChannelsCount);

return true;

bool SoundFileOgg::OpenRead(const char* Data, std::size_t SizeInBytes, std::size_t& NbSamples) {
    // Close the file if already opened
    if (myStream)
        stb_vorbis_close(myStream);
    // Open the vorbis stream
    unsigned char* Buffer = reinterpret_cast<unsigned char*>(SizeInBytes);
    int Length = static_cast<int>(SizeInBytes);
    myStream = stb_vorbis_open_memory(Buffer, Length, NULL, NULL);
    if (myStream == NULL)
        {  
            std::cerr << "Failed to read sound file from memory (cannot open the file)"
            return false;
        }
    // Get the music parameters
    stb_vorbis_info Infos = stb_vorbis_get_info(myStream);
    ChannelsCount = myChannelsCount = Infos.channels;
    SampleRate = Infos.sample_rate;
    NbSamples = static_cast<std::size_t>(stb_vorbis_stream_length_in_samples(myStream) * ChannelsCount);
    return true;
}

std::size_t SoundFileOgg::Read(Int16* Data, std::size_t NbSamples) {
    if (myStream && Data && NbSamples)
```cpp
{ int Read = stb_vorbis_get_samples_short_interleaved(myStream,
myChannelsCount,
Data,
static_cast<int>(NbSamples));
return static_cast<std::size_t>(Read * n);
}
else {
return 0;
}
}
} // namespace priv
} // namespace sf
```
<table>
<thead>
<tr>
<th>Main Page</th>
<th>Namespaces</th>
<th>Classes</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>File List</strong></td>
</tr>
</tbody>
</table>
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.

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

#ifndef SFML_SOUNDFILEOGG_HPP
#define SFML_SOUNDFILEOGG_HPP

namespace sf
{

#include <SFML/Audio/SoundFile.hpp>

struct stb_vorbis;

namespace sf
{

namespace priv {

class SoundFileOgg : public SoundFile {

public:

    SoundFileOgg();

    ~SoundFileOgg();

    static bool IsFileSupported(const std::string& Filename);

    static bool IsFileSupported(const char* Data, std::size_t SizeInBytes);

    virtual std::size_t Read(Int16* Data, std::size_t NbSamples);

private:

    virtual bool OpenRead(const std::string& Filename, std::size_t& NbSamples);

    virtual bool OpenRead(const char* Data, std::size_t SizeInBytes, std::size_t& NbSamples);

    // Member data
    stb_vorbis* myStream;

    unsigned int myChannelsCount;

};

} // namespace priv

} // namespace sf

} // namespace priv

#endif // SFML_SOUNDFILEOGG_HPP

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::

Documentation generated by doxygen 1.5.2 ::
System.hpp

// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any
// express or implied warranty. In no event will
// the authors be held liable for any damages arising
// from the use of this software.

// Permission is granted to anyone to use this software
// for any purpose, including commercial applications,
// and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misreprented;
//    you must not claim that you wrote the original
//    software. If you use this software in a product,
//    an acknowledgment in the product documentation
//    would be appreciated but is not required.

// 2. Altered source versions must be plainly marked
//    as such, and must not be misrepresented as being
//    the original software.

// 3. This notice may not be removed or altered.

#ifndef SFML_SYSTEM_HPP
#define SFML_SYSTEM_HPP

// Headers

#include <SFML/Config.hpp>
#include <SFML/System/Clock.hpp>
#include <SFML/System/Lock.hpp>
#include <SFML/System/Mutex.hpp>
#include <SFML/System/Randomizer.hpp>
#include <SFML/System/Sleep.hpp>
#endif // SFML_SYSTEM_HPP
#include <SFML/System/Thread.hpp>
#include <SFML/System/Unicode.hpp>
#include <SFML/System/Vector2.hpp>
#include <SFML/System/Vector3.hpp>

#endif // SFML_SYSTEM_HPP
SFML
Simple and Fast Multimedia Library

Main Page  Namespaces  Classes  Files

File List
Thread.hpp

// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gomila@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
// you must not claim that you wrote the original software.
// If you use this software in a product, an acknowledgment
// in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
// and must not be misrepresented as being the original software.

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

#ifndef SFML_THREAD_HPP
#define SFML_THREAD_HPP

// Headers
#include <SFML/Config.hpp>

#ifdef SFML_SYSTEM_WINDOWS
#include <SFML/System/Win32/Thread.hpp>

#endif
#endif

#include <SFML/System/Win32/Thread.hpp>
```cpp
#include <SFML/System/Unix/Thread.hpp>
#endif
#endif // SFML_THREAD_HPP
```
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any
// express or implied warranty. In no event will the authors
// be held liable for any damages arising from the use of this
// software.

// Permission is granted to anyone to use this software
// for any purpose, including commercial applications,
// and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be
//    misrepresented; you must not claim that you
//    wrote the original software. If you use this
//    software in a product, an acknowledgment
//    in the product documentation would be
//    appreciated but is not required.

// 2. Altered source versions must be plainly
//    marked as such, and must not be misrepresented
//    as being the original software.

// 3. This notice may not be removed or altered.

template <typename T>
Vector2<T>::Vector2() :
    x(0),
    y(0)
{
}

template <typename T>
Vector2<T>::Vector2(T X, T Y) :
    x(X),
    y(Y)
{
}

template <typename T>
Vector2<T> operator -(const Vector2<T>& V)
{
    return Vector2<T>(-V.x, -V.y);
}

template <typename T>
Vector2<T>& operator +=(Vector2<T>& V1, const Vector2<T>& V2)
{
    V1.x += V2.x;
    V1.y += V2.y;
    return V1;
}

template <typename T>
Vector2<T>& operator -= (Vector2<T>& V1, const Vector2<T>& V2)
{
    V1.x -= V2.x;
    V1.y -= V2.y;
    return V1;
}

template <typename T>
Vector2<T> operator +(const Vector2<T>& V1, const
return Vector2<T>(V1.x + V2.x, V1.y + V2.y);

template <typename T>
Vector2<T> operator -(const Vector2<T>& V1, const Vector2<T>& V2) {
    return Vector2<T>(V1.x - V2.x, V1.y - V2.y);
}

template <typename T>
Vector2<T> operator *(const Vector2<T>& V, T X) {
    return Vector2<T>(V.x * X, V.y * X);
}

template <typename T>
Vector2<T> operator *(T X, const Vector2<T>& V) {
    return Vector2<T>(V.x * X, V.y * X);
}

template <typename T>
Vector2<T>& operator *=(Vector2<T>& V, T X) {
    V.x *= X;
    V.y *= X;
    return V;
}
Vector2<T> operator /(const Vector2<T>& V, T X) {
    return Vector2<T>(V.x / X, V.y / X);
}

template<typename T>
Vector2<T>& operator /= (Vector2<T>& V, T X) {
    V.x /= X;
    V.y /= X;
    return V;
}

template<typename T>
bool operator ==(const Vector2<T>& V1, const Vector2<T>& V2) {
    return (V1.x == V2.x) && (V1.y == V2.y);
}

template<typename T>
bool operator !=(const Vector2<T>& V1, const Vector2<T>& V2) {
    return (V1.x != V2.x) || (V1.y != V2.y);
}
Vector3.inl

template <typename T>
Vector3<T>::Vector3() :
    x(0),
    y(0),
    z(0)
{ }

template <typename T> 
Vector3<T>::Vector3(T X, T Y, T Z) : 
  x(X), 
  y(Y), 
  z(Z) 
{ 
  } 

template <typename T> 
Vector3<T> operator -(const Vector3<T>& V) 
{ 
  return Vector3<T>(-V.x, -V.y, -V.z); 
} 

template <typename T> 
Vector3<T>& operator +=(Vector3<T>& V1, const Vector3<T>& V2) 
{ 
  V1.x += V2.x; 
  V1.y += V2.y; 
  V1.z += V2.z; 
  return V1; 
} 

template <typename T> 
Vector3<T>& operator -=(Vector3<T>& V1, const Vector3<T>& V2) 
{ 
  V1.x -= V2.x; 
  V1.y -= V2.y; 
  V1.z -= V2.z; 
  return V1; 
}
template <typename T>
Vector3<T> operator + (const Vector3<T>& V1, const Vector3<T>& V2)
{
    return Vector3<T>(V1.x + V2.x, V1.y + V2.y, V1.z + V2.z);
}

template <typename T>
Vector3<T> operator - (const Vector3<T>& V1, const Vector3<T>& V2)
{
    return Vector3<T>(V1.x - V2.x, V1.y - V2.y, V1.z - V2.z);
}

template <typename T>
Vector3<T> operator *(const Vector3<T>& V, T X)
{
    return Vector3<T>(V.x * X, V.y * X, V.z * X);
}

template <typename T>
Vector3<T> operator *(T X, const Vector3<T>& V)
{
    return Vector3<T>(V.x * X, V.y * X, V.z * X);
}

template <typename T>
Vector3<T>& operator *=(Vector3<T>& V, T X)
{
    V.x *= X;
    V.y *= X;
    V.z *= X;
}
template <typename T>
Vector3<T> operator / (const Vector3<T>& V, T X) {
    return Vector3<T>(V.x / X, V.y / X, V.z / X);
}

template <typename T>
Vector3<T>& operator /= (Vector3<T>& V, T X) {
    V.x /= X;
    V.y /= X;
    V.z /= X;
    return V;
}

template <typename T>
bool operator ==(const Vector3<T>& V1, const Vector3<T>& V2) {
    return (V1.x == V2.x) && (V1.y == V2.y) && (V1.z == V2.z);
}

template <typename T>
bool operator !=(const Vector3<T>& V1, const Vector3<T>& V2) {
    return (V1.x != V2.x) || (V1.y != V2.y) || (V1.z != V2.z);
}
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)
//
// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from
// the use of this software.
//
// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it
// freely, subject to the following restrictions:
//
// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is
//    not required.
//
// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the
//    original software.
//
// 3. This notice may not be removed or altered.

// Headers
#include <SFML/Window/Cocoa/VideoModeSupport.hpp>
#include <ApplicationServices/ApplicationServices.h>
#include <algorithm>

namespace sf
{
    namespace priv
    {
        void VideoModeSupport::GetSupportedVideoModes(std::vector<VideoMode>& Modes)
        {
            // Implementation
        }
    }
}
00041 // Ceylo -- using same implementation as in OSXCarbon
00042
00043 // First, clear array to fill
00044 Modes.clear();
00045
00046 // Enumerate all available video modes for primary display adapter
00047 CFArrayRef DisplayModes = CGDisplayAvailableModes(kCGDirectMainDisplay);
00048 CFIndex DisplayModeCount = CFArrayGetCount(DisplayModes);
00049 CFDictionaryRef CurrentMode;
00050
00051 for (int Count = 0; Count < DisplayModeCount; ++Count)
00052 {
00053 CurrentMode = (CFDictionaryRef)CFArrayGetValueAtIndex(DisplayModes, Count);
00054 VideoMode Mode;
00055
00056 CFNumberGetValue((CFNumberRef)CFDictionaryGetValue(CurrentMode, kCGDisplayWidth), kCFNumberIntType, &Mode.Width);
00057 CFNumberGetValue((CFNumberRef)CFDictionaryGetValue(CurrentMode, kCGDisplayHeight), kCFNumberIntType, &Mode.Height);
00058 CFNumberGetValue((CFNumberRef)CFDictionaryGetValue(CurrentMode, kCGDisplayBitsPerPixel), kCFNumberIntType, &Mode.BitsPerPixel);
00059
00060 // Add it only if it is not already in the array
00061 if (std::find(Modes.begin(), Modes.end(), Mode) == Modes.end())
00062 {
00063 Modes.push_back(Mode);
00064 }
00065 }
00066
00067 VideoMode VideoModeSupport::GetDesktopVideoMode()
00068 {
00069 // Ceylo -- using same implementation as in OSXCarbon
00070 CFDictionaryRef CurrentVideoMode = CGDisplayCurrentMode(kCGDirectMainDisplay);
00071 VideoMode DesktopMode;
00072
00073 // Get video mode width
00074 CFNumberGetValue((CFNumberRef)CFDictionaryGetValue(CurrentVideoMode, kCGDisplayWidth), kCFNumberIntType, &DesktopMode.Width);
00075 CFNumberGetValue((CFNumberRef)CFDictionaryGetValue(CurrentVideoMode, kCGDisplayHeight), kCFNumberIntType, &DesktopMode.Height);
00076 CFNumberGetValue((CFNumberRef)CFDictionaryGetValue(CurrentVideoMode, kCGDisplayBitsPerPixel), kCFNumberIntType, &DesktopMode.BitsPerPixel);
00077
00078 return DesktopMode;
00079
00080 // Get video mode width
CFNumberGetValue((CFNumberRef)CFDictionaryGetValue(CurrentVideoMode, kCGDisplayWidth),
&((DesktopMode.Width));

// Get video mode height
CFNumberGetValue((CFNumberRef)CFDictionaryGetValue(CurrentVideoMode, kCGDisplayHeight),
&((DesktopMode.Height));

// Get video mode depth
CFNumberGetValue((CFNumberRef)CFDictionaryGetValue(CurrentVideoMode, kCGDisplayBitsPerPixel),
&((DesktopMode.BitsPerPixel));

return DesktopMode;

} // namespace priv

} // namespace sf
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.
// 3. This notice may not be removed or altered from any source distribution.

// Headers
#include <SFML/Window/XXX/VideoModeSupport.hpp>

namespace sf {
    namespace priv {

        void VideoModeSupport::GetSupportedVideoModes(std::vector<VideoMode>& Modes) {
            // Get all the supported fullscreen modes and put them in the Modes array
        }
    }
}

// Just care about width, height and bpp (ignore frequency and other attributes)
// You must remove duplicates
// Order doesn't matter (the array will be sorted later)

VideoMode VideoModeSupport::GetDesktopVideoMode()
{
    // Return the current desktop video mode

    } // namespace priv

} // namespace sf

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gomila@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
including commercial applications, and to alter it and redistribute it freely,
subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software.
If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.

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

// Headers
#include <SFML/Window/Win32/VideoModeSupport.hpp>
#include <windows.h>
#include <algorithm>

namespace sf
{
namespace priv
{
void VideoModeSupport::GetSupportedVideoModes(std::vector<VideoMode>& Modes)
// First, clear array to fill
Modes.clear();

// Enumerate all available video modes for primary display adapter
DEVMODE Win32Mode;
Win32Mode.dmSize = sizeof(DEVMODE);
for (int Count = 0; EnumDisplaySettings(NULL, Count, &Win32Mode); ++Count)
{
    // Convert to sfVideoMode
    VideoMode Mode(Win32Mode.dmPelsWidth, Win32Mode.dmPelsHeight, Win32Mode.dmBitsPerPel);
    // Add it only if it is not already in the array
    if (std::find(Modes.begin(), Modes.end(), Mode) == Modes.end())
        Modes.push_back(Mode);
}

// namespace priv

// namespace sf

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
<table>
<thead>
<tr>
<th>Main Page</th>
<th>Namespaces</th>
<th>Classes</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>File List</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.

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

#ifndef SFML_VIDEOMODESUPPORTCOCOA_HPP
#define SFML_VIDEOMODESUPPORTCOCOA_HPP

namespace sf
{

namespace priv

#include <SFML/Window/VideoMode.hpp>
#include <vector>

namespace sf
{

namespace priv

#endif
class VideoModeSupport {
public:
    static void GetSupportedVideoModes(std::vector<VideoMode>& Modes);
    static VideoMode GetDesktopVideoMode();
};

} // namespace priv

} // namespace sf

#endif // SFML_VIDEOMODESUPPORTCOCOA_HPP
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
//  you must not claim that you wrote the original software.
//  If you use this software in a product, an
//  acknowledgment in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
//  and must not be misrepresented as being the original software.

// 3. This notice may not be removed or altered.

#ifndef SFML_VIDEOMODESUPPORTXXX_HPP
#define SFML_VIDEOMODESUPPORTXXX_HPP

// Headers
#include <SFML/Window/VideoMode.hpp>
#include <vector>

namespace sf
{
  namespace priv
class VideoModeSupport {
public:
    static void GetSupportedVideoModes(std::vector<VideoMode>& Modes);
    static VideoMode GetDesktopVideoMode();
};

} // namespace priv
} // namespace sf

#endif // SFML_VIDEOMODESUPPORTXXX_HPP
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gomila@gmail.com)
// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.
// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.

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

#ifndef SFML_VIDEOMODESUPPORT_HPP
#define SFML_VIDEOMODESUPPORT_HPP

// Headers
#include <SFML/Config.hpp>

#if defined(SFML_SYSTEM_WINDOWS)

#include <SFML/Window/Win32/VideoModeSupport.hpp>

#endif

#endif
#elif defined(SFML_SYSTEM_LINUX) || defined(SFML_SYSTEM_FREEBSD)
#include <SFML/Window/Linux/VideoModeSupport.hpp>

#elif defined(SFML_SYSTEM_MACOS)
#include <SFML/Window/Cocoa/VideoModeSupport.hpp>
#endif
#endif // SFML_VIDEOMODESUPPPORT_HPP

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2008 Laurent Gomila (laurent.gomila@gmail.com)
//
// This software is provided 'as-is', without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software.
//
// Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:
//
// 1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.
//
// 2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.
//
// 3. This notice may not be removed or altered from any source distribution.

#ifndef SFML_VIDEOMODESUPPORTWIN32_HPP
#define SFML_VIDEOMODESUPPORTWIN32_HPP

// Headers
#include <SFML/Window/VideoMode.hpp>
#include <vector>

namespace sf {

namespace priv {
class VideoModeSupport {
    public :
        static void GetSupportedVideoModes(std::vector<VideoMode>& Modes);
        static VideoMode GetDesktopVideoMode();
};

// namespace priv

// namespace sf

#endif // SFML_VIDEOMODESUPPORTWIN32_HPP

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.

// 3. This notice may not be removed or altered.

#ifndef SFML_SFML_WINDOW_HPP
#define SFML_SFML_WINDOW_HPP

// Headers

#include <SFML/System.hpp>
#include <SFML/Window/Context.hpp>
#include <SFML/Window/Event.hpp>
#include <SFML/Window/Input.hpp>
#include <SFML/Window/VideoMode.hpp>
#include <SFML/Window/Window.hpp>
#endif
```cpp
#include <SFML/Window/WindowListener.hpp>
#include <SFML/Window/WindowStyle.hpp>
#include <SFML/Window/OpenGL.hpp>

#endif // SFML_SFML_WINDOW_HPP
```
<table>
<thead>
<tr>
<th>File List</th>
</tr>
</thead>
</table>
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.

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

#ifndef SFML_WINDOWHANDLE_HPP
#define SFML_WINDOWHANDLE_HPP

// Headers
#include <SFML/Config.hpp>

namespace sf
{
    #if defined(SFML_SYSTEM_WINDOWS)

    #ifndef SFML_WINDOWHANDLE_HPP
    #define SFML_WINDOWHANDLE_HPP
    #endif

    // Headers
    #include <SFML/Config.hpp>

    namespace sf
    {
        #if defined(SFML_SYSTEM_WINDOWS)

        #endif
    }

    #endif
}
// Windows defines a void* handle (HWND)
typedef void* WindowHandle;

#elif defined(SFML_SYSTEM_LINUX) || defined(SFML_SYSTEM_FREEBSD)
// Unix - X11 defines an unsigned integer handle (Window)
typedef unsigned long WindowHandle;

#elif defined(SFML_SYSTEM_MACOS)
// Mac OS X defines a void* handle (NSWindow)
typedef void* WindowHandle;

#endif // SFML_WINDOWHANDLE_HPP
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.

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

// Headers
#include <SFML/Window/WindowImpl.hpp>
#include <SFML/Window/Event.hpp>
#include <SFML/Window/WindowListener.hpp>
#include <algorithm>
#include <cmath>

#if defined(SFML_SYSTEM_WINDOWS)

typedef sf::priv::WindowImplWin32 WindowImplType;
#endif
#elif defined(SFML_SYSTEM_LINUX) || defined(SFML_SYSTEM_FREEBSD)
#include <SFML/Window/Linux/WindowImplX11.hpp>
typedef sf::priv::WindowImplX11 WindowImplType;
#elif defined(SFML_SYSTEM_MACOS)
#include <SFML/Window/Cocoa/WindowImplCocoa.hpp>
typedef sf::priv::WindowImplCocoa WindowImplType;
#endif
namespace sf {
namespace priv {

WindowImpl* WindowImpl::New()
{
    return new WindowImplType();
}

WindowImpl* WindowImpl::New(VideoMode Mode, const...
{
    return new WindowImplType(Mode, Title, WindowStyle, Params);
}

WindowImpl* WindowImpl::New(WindowHandle Handle, WindowSettings&
{
    return new WindowImplType(Handle, Params);
}

WindowImpl::WindowImpl() :

WindowImpl::~WindowImpl()
{
    // Nothing to do
}

void WindowImpl::AddListener(WindowListener* Listener)
{
    if (Listener)
    
        myListeners.insert(Listener);
}

void WindowImpl::RemoveListener(WindowListener* Listener)
{
    myListeners.erase(Listener);
}

void WindowImpl::Initialize()
{
    // Initialize the joysticks
    for (unsigned int i = 0; i < Joy::Count; ++i)
    {
        myJoysticks[i].Initialize(i);
        myJoyStates[i] = myJoysticks[i].UpdateState();
    }
}
unsigned int WindowImpl::GetWidth() const
{
    return myWidth;
}

unsigned int WindowImpl::GetHeight() const
{
    return myHeight;
}

void WindowImpl::SetJoystickThreshold(float Threshold)
{
    myJoyThreshold = Threshold;
}

void WindowImpl::DoEvents()
{
    // Read the joysticks state and generate the appropriate events
    ProcessJoystickEvents();

    // Let the derived class process other events
    ProcessEvents();
}

bool WindowImpl::IsContextActive()
{
    return WindowImplType::IsContextActive();
}

void WindowImpl::SendEvent(const Event& EventToSend)
{
    for (std::set<WindowListener*>::iterator i = myListeners.begin(); i != myListeners.end(); ++i)
    {
        (*i)->ProcessEvent(EventToSend);
    }
}
{ (*i)->OnEvent(EventToSend); }

int WindowImpl::EvaluateConfig(const VideoMode& Mode, { return abs(static_cast<int>(Mode.BitsPerPixel - ColorBits)) + abs(static_cast<int>(Settings.DepthBits - DepthBits)) + abs(static_cast<int>(Settings.StencilBits - StencilBits)) + abs(static_cast<int>(Settings.AntialiasingLevel - Antialiasing)); }

void WindowImpl::ProcessJoystickEvents() { for (unsigned int i = 0; i < Joy::Count; ++i) { // Copy the previous state of the joystick JoystickState PreviousState = myJoyStates[i]; myJoyStates[i] = myJoysticks[i].UpdateState(); // Axis for (unsigned int j = 0; j < Joy::AxisCount; ++j) { Joy::Axis Axis = static_cast<Joy::Axis>(j); if (myJoysticks[i].HasAxis(Axis)) { float PrevPos = PreviousState.Axis[j]; float CurrPos = myJoyStates[i].Axis[j]; if (fabs(CurrPos - PrevPos) >= myJoyThreshold) { Event Event; Event.Type = Event::JoyMoved; Event.JoyMove.JoystickId = i; Event.JoyMove.Axis = Axis; } } } } }
SendEvent(Event);

// Buttons
for (unsigned int j = 0; j < myJoysticks.size(); ++j)
{
    bool PrevPressed = PreviousState.Buttons[j];
    bool CurrPressed = myJoyStates[i].Buttons[j];
    {
        Event Event;
        Event.JoyButton.JoystickId = i;
        SendEvent(Event);
    }
}

} // namespace priv
} // namespace sf
//
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)
//
// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.
// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:
// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is not required.
// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.
// 3. This notice may not be removed or altered.

#ifndef SFML_WINDOWIMPL_HPP
#define SFML_WINDOWIMPL_HPP

// Headers
#include <SFML/Config.hpp>
#include <SFML/System/NonCopyable.hpp>
#include <SFML/Window/Joystick.hpp>
#include <SFML/Window/VideoMode.hpp>
#include <SFML/Window/WindowHandle.hpp>
#include <SFML/Window/WindowSettings.hpp>
#include <set>

```cpp
#include <string>

namespace sf {
    class Event;
    class WindowListener;
}

namespace priv {
    class WindowImpl : NonCopyable {
        public :
            static WindowImpl* New();
            static WindowImpl* New(VideoMode Mode, const WindowSettings& Params);
            public :
            virtual ~WindowImpl();
            void AddListener(WindowListener* Listener);
            void RemoveListener(WindowListener* Listener);
            void Initialize();
            unsigned int GetWidth() const;
            unsigned int GetHeight() const;
            virtual void SetActive(bool Active = true);
            void SetJoystickThreshold(float Threshold);
    }
```
void DoEvents();
static bool IsContextActive();
virtual void Display() = 0;
virtual void UseVerticalSync(bool Enabled) = 0;
virtual void ShowMouseCursor(bool Show) = 0;
virtual void SetCursorPosition(unsigned int Left, unsigned int Top) = 0;
virtual void SetPosition(int Left, int Top) = 0;
virtual void SetSize(unsigned int Width, unsigned int Height) = 0;
virtual void Show(bool State) = 0;
virtual void EnableKeyRepeat(bool Enabled) = 0;
virtual void SetIcon(unsigned int Width, unsigned int Height) = 0;

protected:
WindowImpl();

void SendEvent(const Event& EventToSend);

static int EvaluateConfig(const VideoMode& Mode);

// Member data
unsigned int myWidth;
unsigned int myHeight;

private:
void ProcessJoystickEvents();

virtual void ProcessEvents() = 0;

// Member data
std::set<WindowListener*> myListeners;

Joystick myJoysticks[Joy::Count]

JoystickState myJoyStates[Joy::Count]

float myJoyThreshold;

};

} // namespace priv

} // namespace sf

#endif // SFML_WINDOWIMPL_HPP

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::

Documentation generated by doxygen 1.5.2 ::
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Lucas Soltic (ceylow@gmail.com) and Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
// you must not claim that you wrote the original software.
// If you use this software in a product, an acknowledgment
// in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
// and must not be misrepresented as being the original software.

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

#ifndef SFML_WINDOWIMPLCOCOA_HPP
#define SFML_WINDOWIMPLCOCOA_HPP

// Headers
#include <SFML/Window/Event.hpp>
#include <SFML/Window/WindowImpl.hpp>
#include <string>

#ifdef __OBJC__
@class sfPrivWindow;
typedef sfPrivWindow* sfPrivWindowRef;
#endif

typedef void* sfPrivWindowRef;
#endif

namespace sf {
namespace priv {

class WindowImplCocoa : public WindowImpl {

public:

WindowImplCocoa();
WindowImplCocoa(WindowHandle Handle, WindowSettings& params);
WindowImplCocoa(VideoMode Mode, const std::string& Title);
~WindowImplCocoa();

static bool IsContextActive();

void HandleNotifiedEvent(Event& eventRef);
void HandleKeyDown(void *eventRef);
void HandleKeyUp(void *eventRef);
void HandleModifierKey(void *eventRef);
void HandleMouseDown(void *eventRef);
void HandleMouseUp(void *eventRef);
void HandleMouseMove(void *eventRef);
void HandleMouseWheel(void *eventRef);

static bool IsTextEvent(void *event);

private :

virtual void Display();
virtual void ProcessEvents();
virtual void SetActive(bool Active = true);
virtual void UseVerticalSync(bool Enabled);
virtual void ShowMouseCursor(bool Show);
virtual void SetCursorPosition(unsigned int Left);
virtual void SetPosition(int Left, int Top);
virtual void SetSize(unsigned int Width, unsigned int Height);
virtual void Show(bool State);
virtual void EnableKeyRepeat(bool Enabled);
virtual void SetIcon(unsigned int Width, unsigned int Height);

// Member data
sfPrivWindowRef myWrapper;
bool myUseKeyRepeat;
bool myMouseIn;
float myWheelStatus;
};
} // namespace priv
} // namespace sf
#endif // SFML_WINDOWIMPLCOCOA_HPP
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.

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

// Headers
#define _WIN32_WINDOWS 0x0501
#define _WIN32_WINNT			0x0501
#include <SFML/Window/Win32/WindowImplWin32.hpp>
#include <SFML/Window/WindowSettings.hpp>
#include <SFML/Window/WindowStyle.hpp>
#include <GL/gl.h>
#include <SFML/Window/glext/wglext.h>
#include <SFML/Window/glext/glext.h>
#include <iostream>
#include <vector>
// MinGW lacks the definition of some Win32 constants
#ifndef XBUTTON1
#define XBUTTON1 0x0001
#endif
#ifndef XBUTTON2
#define XBUTTON2 0x0002
#endif
#ifndef MAPVK_VK_TO_VSC
#define MAPVK_VK_TO_VSC (0)
#endif

namespace sf
{
namespace priv
{
// Static member data
unsigned int WindowImplWin32::ourWindowCount = 0;
const char* WindowImplWin32::ourClassNameA = NULL;
const wchar_t* WindowImplWin32::ourClassNameW = LWindowImplWin32*
WindowImplWin32::ourFullscreenWindow = NULL;

WindowImplWin32::WindowImplWin32() :
myHandle (NULL),
myCallback (0),
myCursor (NULL),
myIcon (NULL),
myKeyRepeatEnabled(true),
myIsCursorIn (false)
{
    // Register the window class at first call
    if (ourWindowCount == 0)
    RegisterWindowClass();

    // Use small dimensions
myWidth = 1;

myHeight = 1;

// Create a dummy window (disabled and hidden)
if (HasUnicodeSupport())
{
    myHandle = CreateWindowW(ourClassNameW,
}
else
{
    myHandle = CreateWindowA(ourClassNameA,
}
ShowWindow(myHandle, SW_HIDE);

// Create the rendering context
if (myHandle)
{
    WindowSettings Params(0, 0, 0);
    CreateContext(VideoMode(myWidth, myHeight),
}
    // Don't activate by default
    SetActive(false);

}
if (myHandle) {
    // Get window client size
    RECT Rect;
    GetClientRect(myHandle, &Rect);
    myWidth = Rect.right - Rect.left;
    myHeight = Rect.bottom - Rect.top;

    // Create the rendering context
    VideoMode Mode(myWidth, myHeight, VideoMode::GetDesktopMode
    CreateContext(Mode, Params);
    // We change the event procedure of the
    SetWindowLongPtr(myHandle, GWLP_USERDATA,
    myCallback = SetWindowLongPtr(myHandle, GWLP_WNDPROC,
    myCallback = SetWindowLongPtr(myHandle, GWLP_WNDPROC,
    myCallback = SetWindowLongPtr(myHandle, GWLP_WNDPROC,
    myCallback = SetWindowLongPtr(myHandle, GWLP_WNDPROC,
    WindowImplWin32::WindowImplWin32(VideoMode Mode,
    myHandle (NULL),
    myCallback (0),
    myCursor (NULL),
    myIcon (NULL),
    myKeyRepeatEnabled(true),
    myIsCursorIn (false)
    { // Register the window class at first call
        if (ourWindowCount == 0)
            RegisterWindowClass();
    // Compute position and size
    HDC ScreenDC = GetDC(NULL);
    int Left = (GetDeviceCaps(ScreenDC, HORZRES
    int Top = (GetDeviceCaps(ScreenDC, VERTRES
    int Width = myWidth = Mode.Width;
    int Height = myHeight = Mode.Height;
ReleaseDC(NULL, ScreenDC);

// Choose the window style according to the Style parameter
DWORD Win32Style = WS_VISIBLE;

if (WindowStyle == Style::None)
{
    Win32Style |= WS_POPUP;
}
else
{
    if (WindowStyle & Style::Titlebar) Win32Style |= WS_CAPTION |
    if (WindowStyle & Style::Resize) Win32Style |= WS_THICKFRAME |
    if (WindowStyle & Style::Close) Win32Style |= WS_SYSMENU;
}

// In windowed mode, adjust width and height
bool Fullscreen = (WindowStyle & Style::Fullscreen)
if (!Fullscreen)
{
    RECT Rect = {0, 0, Width, Height};
    AdjustWindowRect(&Rect, Win32Style, false
    Width = Rect.right - Rect.left;
    Height = Rect.bottom - Rect.top;
}

// Create the window
if (HasUnicodeSupport())
{
    wchar_t WTitle[256];
    int NbChars = MultiByteToWideChar(CP_ACP, WTitle[NbChars] = L'\0';
    myHandle = CreateWindowW(ourClassNameW, myHandle = CreateWindowA(ourClassNameA,
// Switch to fullscreen if requested
if (Fullscreen)
    SwitchToFullscreen(Mode);

// Create the rendering context
if (myHandle)
    CreateContext(Mode, Params);

// Increment window count
ourWindowCount++;

// Get the actual size of the window, which
// This happens when the window is bigger than
RECT ActualRect;
GetClientRect(myHandle, &ActualRect);
myWidth = ActualRect.right - ActualRect.left;
myHeight = ActualRect.bottom - ActualRect.top;

WindowImplWin32::~WindowImplWin32()
{
    // Destroy the custom icon, if any
    if (myIcon)
        DestroyIcon(myIcon);
    if (!myCallback)
    {
        // Destroy the window
        if (myHandle)
            DestroyWindow(myHandle);
        // Decrement the window count
        ourWindowCount--;
        // Unregister window class if we were th
if (ourWindowCount == 0)
{
    if (HasUnicodeSupport())
    {
        UnregisterClassW(ourClassNameW, GetModuleHandle(NULL));
    }
    else
    {
        UnregisterClassA(ourClassNameA, GetModuleHandle(NULL));
    }
}
else
{
    // The window is external : remove the hook
    SetWindowLongPtr(myHandle, GWLP_WNDPROC, (long) (long) myCallback);
}

bool WindowImplWin32::IsContextActive()
{
    return wglGetCurrentContext() != NULL;
}

void WindowImplWin32::ProcessEvents()
{
    // We update the window only if we own it
    if (!myCallback)
    {
        MSG Message;
        while (PeekMessage(&Message, NULL, 0, 0, PM_REMOVE))
        {
            TranslateMessage(&Message);
            DispatchMessage(&Message);
        }
    }
void WindowImplWin32::Display()
{
    if (myDeviceContext && myGLContext)
    {
        SwapBuffers(myDeviceContext);
    }
}

void WindowImplWin32::SetActive(bool Active) const
{
    if (Active)
    {
        if (myDeviceContext && myGLContext && (wglGetCurrentContext() != myGLContext))
            wglMakeCurrent(myDeviceContext, myGLContext);
    }
    else
    {
        if (wglGetCurrentContext() == myGLContext)
            wglMakeCurrent(NULL, NULL);
    }
}

void WindowImplWin32::UseVerticalSync(bool Enabled)
{
    if (wglSwapIntervalEXT)
        wglSwapIntervalEXT(Enabled ? 1 : 0);
}

void WindowImplWin32::ShowMouseCursor(bool Show)
{
    if (Show)
myCursor = LoadCursor(NULL, IDC_ARROW);
else
    myCursor = NULL;
SetCursor(myCursor);
}
void WindowImplWin32::SetCursorPosition(unsigned)
{
    POINT Pos = {Left, Top};
    ClientToScreen(myHandle, &Pos);
    SetCursorPos(Pos.x, Pos.y);
}
void WindowImplWin32::SetPosition(int Left, int)
{
    SetWindowPos(myHandle, NULL, Left, Top, 0, 0, SWP_NOSIZE | SWP_NOZORDER);
}
void WindowImplWin32::SetSize(unsigned int Width)
{
    RECT Rect = {0, 0, Width, Height};
    AdjustWindowRect(&Rect, GetWindowLong(myHandle, GWL_STYLE), Width = Rect.right - Rect.left;
    Height = Rect.bottom - Rect.top;
    SetWindowPos(myHandle, NULL, 0, 0, Width, Height);
}
void WindowImplWin32::Show(bool State)
{
ShowWindow(myHandle, State ? SW_SHOW : SW_HIDE);
#endif

void WindowImplWin32::EnableKeyRepeat(bool Enabled)
{
    myKeyRepeatEnabled = Enabled;
}

void WindowImplWin32::SetIcon(unsigned int Width, unsigned int Height)
{
    // First destroy the previous one
    if (myIcon)
        DestroyIcon(myIcon);

    // Windows wants BGRA pixels : swap red and blue channels
    std::vector<Uint8> IconPixels(Width * Height);
    for (std::size_t i = 0; i < IconPixels.size() / 4; ++i)
    {
        IconPixels[i * 4 + 0] = Pixels[i * 4 + 2];
        IconPixels[i * 4 + 1] = Pixels[i * 4 + 1];
        IconPixels[i * 4 + 2] = Pixels[i * 4 + 0];
        IconPixels[i * 4 + 3] = Pixels[i * 4 + 3];
    }

    // Create the icon from the pixels array
    myIcon = CreateIcon(GetModuleHandle(NULL), Width, Height, 1, 32, NULL, &IconPixels[0]);

    // Set it as both big and small icon of the window
    if (myIcon)
        { 
            SendMessage(myHandle, WM_SETICON, ICON_BIG, (LPARAM)myIcon);
            SendMessage(myHandle, WM_SETICON, ICON_SMALL, (LPARAM)myIcon);
        }
    else
    {

void WindowImplWin32::RegisterWindowClass()
{
    if (HasUnicodeSupport())
    {
        WNDCLASSW WindowClass;
        WindowClass.style = 0;
        WindowClass.lpfnWndProc = &WindowImplWin32::GlobalOnEvent;
        WindowClass.cbClsExtra = 0;
        WindowClass.cbWndExtra = 0;
        WindowClass.hInstance = GetModuleHandle(NULL);
        WindowClass.hIcon = NULL;
        WindowClass.hCursor = 0;
        WindowClass.hbrBackground = 0;
        WindowClass.lpszMenuName = NULL;
        WindowClass.lpszClassName = ourClassNameW;
        RegisterClassW(&WindowClass);
    }
    else
    {
        WNDCLASSA WindowClass;
        WindowClass.style = 0;
        WindowClass.lpfnWndProc = &WindowImplWin32::GlobalOnEvent;
        WindowClass.cbClsExtra = 0;
        WindowClass.cbWndExtra = 0;
        WindowClass.hInstance = GetModuleHandle(NULL);
        WindowClass.hIcon = NULL;
        WindowClass.hCursor = 0;
        WindowClass.hbrBackground = 0;
        WindowClass.lpszMenuName = NULL;
        WindowClass.lpszClassName = ourClassNameA;
        RegisterClassA(&WindowClass);
    }
}
void WindowImplWin32::SwitchToFullscreen(const VideoMode& Mode) {
    DEVMODE DevMode;
    DevMode.dmSize = sizeof(DEVMODE);
    DevMode.dmPelsWidth = Mode.Width;
    DevMode.dmPelsHeight = Mode.Height;
    DevMode.dmBitsPerPel = Mode.BitsPerPixel;
    DevMode.dmFields = DM_PELSWIDTH | DM_PELSHIGHT;

    // Apply fullscreen mode
    if (ChangeDisplaySettings(&DevMode, CDS_FULLSCREEN) != DISP_CHANGE_SUCCESSFUL) {
        std::cerr << "Failed to change display mode for fullscreen"
        return;
    }

    // Make the window flags compatible with fullscreen mode
    SetWindowLong(myHandle, GWL_STYLE, WS_POPUP | WS_CLIPCHILDREN | WS_CLIPSIBLINGS);
    SetWindowLong(myHandle, GWL_EXSTYLE, WS_EX_APPWINDOW);

    // Resize the window so that it fits the entire screen
    SetWindowPos(myHandle, HWND_TOP, 0, 0, Mode.Width, Mode.Height, SWP_FRAMECHANGED);
    ShowWindow(myHandle, SW_SHOW);

    // Set "this" as the current fullscreen window
    ourFullscreenWindow = this;
}

void WindowImplWin32::CreateContext(const VideoMode& Mode, WindowSettings& Params) {
    // Get the device context attached to the window
    myDeviceContext = GetDC(myHandle);

    if (myDeviceContext == NULL)
```cpp
std::cerr << "Failed to get device context -- cannot create OpenGL context"

return;

// Let's find a suitable pixel format -- first try with antialiasing
int BestFormat = 0;
if (Params.AntialiasingLevel > 0)
{
    // Get the wglChoosePixelFormatARB function (it is an extension)
    PFNWGLCHOOSEPIXELFORMATARBPROC wglChoosePixelFormatARB =
        (PFNWGLCHOOSEPIXELFORMATARBPROC)wglGetProcAddress(OpenGLProcTable, "wglChoosePixelFormatARB");
    if (wglChoosePixelFormatARB)
    {
        // Define the basic attributes we want for our window
        int IntAttributes[] = {
            WGL_DRAW_TO_WINDOW_ARB, GL_TRUE,
            WGL_SUPPORT_OPENGL_ARB, GL_TRUE,
            WGL_ACCELERATION_ARB, WGL_FULL ACCELERATION_ARB,
            WGL_DOUBLE_BUFFER_ARB, GL_TRUE,
            WGL_SAMPLE_BUFFERS_ARB, (Params.AntialiasingLevel ? GL_TRUE : GL_FALSE),
            WGL_SAMPLES_ARB, Params.AntialiasingLevel,
            0, 0
        };

        // Let's check how many formats are supporting our requirements
        int Formats[128];
        UINT NbFormats;
        float FloatAttributes[] = {0, 0};
        bool IsValid = wglChoosePixelFormatARB(myDeviceContext, IntAttributes, FloatAttributes,
            &NbFormats, Formats);
        if (!IsValid || (NbFormats == 0))
        {
            if (Params.AntialiasingLevel > 2)
            {
                // No format matching our needs: reduce the multisampling level
                std::cerr << "Failed to find a pixel format supporting " << Params.AntialiasingLevel
                    << " antialiasing levels."
            }
        }
    }
}
```
00542
00543  Params.AntialiasingLevel = IntAttributes[11];
00544  IsValid = wglChoosePixelFormatARB(myDeviceContext,
00545                    IntAttributes,
00546                    FloatAttributes,
00547  };
00548
00549  if (!IsValid || (NbFormats == 0))
00550  {
00551    // Cannot find any pixel format supporting multisampling;
disabling antialiasing
00552    std::cerr << "Failed to find a pixel format supporting antialiasing; antialiasing will be disabled";
00553    Params.AntialiasingLevel = 0;
00554  }
00555  // Get the best format among the returned ones
00556  if (IsValid && (NbFormats > 0))
00557  {
00558    int BestScore = 0xFFFF;
00559    for (UINT i = 0; i < NbFormats; ++i)
00560      {
00561        // Get the current format's attributes
00562        PIXELFORMATDESCRIPTOR Attributes;
00563        Attributes.nSize = sizeof(PIXELFORMATDESCRIPTOR);
00564        Attributes.nVersion = 1;
00565        DescribePixelFormat(myDeviceContext, Formats[i],
00566        // Evaluate the current configuration
00567        int Color = Attributes.cRedBits + Attributes.cGreenBits + Attributes.cBlueBits + Attributes.cAlphaBits;
00568        int Score = EvaluateConfig(Mode, Params, Color, Attributes.cDepthBits, Attributes.cStencilBits, Params.AntialiasingLevel);
00569        int BestFormat = Formats[i];
00570        // Keep it if it's better than the current best
00571        if (Score < BestScore)
00572          {
00573            BestScore = Score;
00574            BestFormat = Formats[i];
00575          }
00576      }
00577  }
else {
    // wglChoosePixelFormatARB not supported
    std::cerr << "Antialiasing is not supported; disabling antialiasing
    Params.AntialiasingLevel = 0;
}

// Find a pixel format with no antialiasing, if not needed or not supported
if (BestFormat == 0)
{
    // Setup a pixel format descriptor from rendering settings
    PIXELFORMATDESCRIPTOR PixelDescriptor;
    ZeroMemory(&PixelDescriptor, sizeof(PIXELFORMATDESCRIPTOR));
    PixelDescriptor.nSize = sizeof(PIXELFORMATDESCRIPTOR);
    PixelDescriptor.nVersion = 1;
    PixelDescriptor.iLayerType = PFD_MAIN_PLANE;
    PixelDescriptor.dwFlags = PFD_DRAW_TO_WINDOW | PFD_SUPPORT_OPENGL | PFD_DOUBLEBUFFER;
    PixelDescriptor.iPixelType = PFD_TYPE_RGBA;
    PixelDescriptor.cColorBits = static_cast<
    PixelDescriptor.cDepthBits = static_cast<
    PixelDescriptor.cStencilBits = static_cast<
    PixelDescriptor.cAlphaBits = Mode.BitsPerPixel == 32 ? 8 : 0;

    // Get the pixel format that best matches our requirements
    BestFormat = ChoosePixelFormat(myDeviceContext, &PixelDescriptor);
    if (BestFormat == 0)
    {
        std::cerr << "Failed to find a suitable pixel format for device context -- cannot create OpenGL context"
        return;
    }

    // Extract the depth and stencil bits from the chosen format
    PIXELFORMATDESCRIPTOR ActualFormat;
    ActualFormat.nSize = sizeof(PIXELFORMATDESCRIPTOR);
ActualFormat.nVersion = 1;
DescribePixelFormat(myDeviceContext, BestFormat,
Params.DepthBits = ActualFormat.cDepthBits;
Params.StencilBits = ActualFormat.cStencilBits;

// Set the chosen pixel format
if (!SetPixelFormat(myDeviceContext, BestFormat)) {
    std::cerr << "Failed to set pixel format for device context -- cannot create OpenGL context"
    return;
}

// Create the OpenGL context from the device
myGLContext = wglCreateContext(myDeviceContext);
if (myGLContext == NULL) {
    std::cerr << "Failed to create an OpenGL context for this window"
    return;
}

// Share display lists with other contexts
HGLRC CurrentContext = wglGetCurrentContext();
if (CurrentContext)
    wglShareLists(CurrentContext, myGLContext);

// Activate the context
SetActive(true);

// Enable multisampling
if (Params.AntialiasingLevel > 0)
    glEnable(GL_MULTISAMPLE_ARB);

void WindowImplWin32::Cleanup()
{
    // Restore the previous video mode (in case
if (ourFullscreenWindow == this)
{
    ChangeDisplaySettings(NULL, 0);
    ourFullscreenWindow = NULL;
}

// Unhide the mouse cursor (in case it was hidden)
ShowMouseCursor(true);

// Destroy the OpenGL context
if (myGLContext)
{
    // Unbind the context before destroying
    SetActive(false);

    wglDeleteContext(myGLContext);
    myGLContext = NULL;
}
if (myDeviceContext)
{
    ReleaseDC(myHandle, myDeviceContext);
    myDeviceContext = NULL;
}

void WindowImplWin32::ProcessEvent(UINT Message, WPARAM WParam, LPARAM LParam)
{
    // Don't process any message until window is created
    if (myHandle == NULL)
        return;

    switch (Message)
    {
    case WM_DESTROY :
        // Destroy event
        case WM_CLOSE :
        {
// Here we must cleanup resources!
Cleanup();
break;
}

// Set cursor event
case WM_SETCURSOR :
{
    // The mouse has moved, if the cursor is in our window we must refresh the cursor
    if (LOWORD(LParam) == HTCLIENT)
        SetCursor(myCursor);
break;
}

// Close event
case WM_CLOSE :
{
    Event Evt;
    Evt.Type = Event::Closed;
    SendEvent(Evt);
break;
}

// Resize event
case WM_SIZE :
{
    // Update window size
    RECT Rect;
    GetClientRect(myHandle, &Rect);
    myWidth = Rect.right - Rect.left;
    myHeight = Rect.bottom - Rect.top;
    Event Evt;
    Evt.Type = Event::Resized;
    Evt.Size.Width = myWidth;
    Evt.Size.Height = myHeight;
SendEvent(Evt);
break;
}

// Gain focus event
case WM_SETFOCUS :
{
    Event Evt;
    Evt.Type = Event::GainedFocus;
    SendEvent(Evt);
    break;
}

// Lost focus event
case WM_KILLFOCUS :
{
    Event Evt;
    Evt.Type = Event::LostFocus;
    SendEvent(Evt);
    break;
}

// Text event
case WM_CHAR :
{
    if (myKeyRepeatEnabled || ((LPARAM &
    
    )
    {
        Event Evt;
        Evt.Type = Event::TextEntered;
        Evt.Text.Unicode = static_cast
        
        } SendEvent(Evt);
    
    break;
    
    }

    // Keydown event
    case WM_KEYDOWN :
    

case WM_SYSKEYDOWN :
{
    if (myKeyRepeatEnabled || ((HIWORD(LParam) & KF_REPEAT) == 0))
    {
        Event Evt;
        Evt.Type = Event::KeyPressed;
        Evt.Key.Alt = HIWORD(GetAsyncKeyState(VK_MENU)) != 0;
        Evt.Key.Control = HIWORD(GetAsyncKeyState(VK_CONTROL)) != 0;
        Evt.Key.Shift = HIWORD(GetAsyncKeyState(VK_SHIFT)) != 0;
        Evt.Key.Code = VirtualKeyCodeToSF(WParam, LParam);
        SendEvent(Evt);
    }
    break;
}

// Keyup event
case WM_KEYUP :
case WM_SYSKEYUP :
{
    Event Evt;
    Evt.Type = Event::KeyReleased;
    Evt.Key.Alt = HIWORD(GetAsyncKeyState(VK_MENU)) != 0;
    Evt.Key.Control = HIWORD(GetAsyncKeyState(VK_CONTROL)) != 0;
    Evt.Key.Shift = HIWORD(GetAsyncKeyState(VK_SHIFT)) != 0;
    Evt.Key.Code = VirtualKeyCodeToSF(WParam, LParam);
    SendEvent(Evt);
    break;
}

// Mouse wheel event
case WM_MOUSEWHEEL :
{
    Event Evt;
    Evt.Type = Event::MouseWheelMoved;
    Evt.MouseWheel.Delta = static_cast<int>[
    SendEvent(Evt);
break;
}

// Mouse left button down event
case WM_LBUTTONDOWN :
{
    Event Evt;
    Evt.Type = Event::MouseButtonPressed;
    Evt.MouseButton.Button = Mouse::Left;
    Evt.MouseButton.X = LOWORD(LParam);
    Evt.MouseButton.Y = HIWORD(LParam);
    SendEvent(Evt);
    break;
}

// Mouse left button up event
case WM_LBUTTONUP :
{
    Event Evt;
    Evt.Type = Event::MouseButtonReleased;
    Evt.MouseButton.Button = Mouse::Left;
    Evt.MouseButton.X = LOWORD(LParam);
    Evt.MouseButton.Y = HIWORD(LParam);
    SendEvent(Evt);
    break;
}

// Mouse right button down event
case WM_RBUTTONDOWN :
{
    Event Evt;
    Evt.Type = Event::MouseButtonPressed;
    Evt.MouseButton.Button = Mouse::Right;
    Evt.MouseButton.X = LOWORD(LParam);
    Evt.MouseButton.Y = HIWORD(LParam);
    SendEvent(Evt);
    break;
// Mouse right button up event
case WM_RBUTTONUP :
{
    Event Evt;
    Evt.Type = Event::MouseButtonReleased;
    Evt.MouseButton.Button = Mouse::Right;
    Evt.MouseButton.X = LOWORD(LParam);
    Evt.MouseButton.Y = HIWORD(LParam);
    SendEvent(Evt);
    break;
}
// Mouse wheel button down event
case WM_MBUTTONDOWN :
{
    Event Evt;
    Evt.Type = Event::MouseButtonPressed;
    Evt.MouseButton.Button = Mouse::Middle;
    Evt.MouseButton.X = LOWORD(LParam);
    Evt.MouseButton.Y = HIWORD(LParam);
    SendEvent(Evt);
    break;
}
// Mouse wheel button up event
case WM_MBUTTONUP :
{
    Event Evt;
    Evt.Type = Event::MouseButtonReleased;
    Evt.MouseButton.Button = Mouse::Middle;
    Evt.MouseButton.X = LOWORD(LParam);
    Evt.MouseButton.Y = HIWORD(LParam);
    SendEvent(Evt);
    break;
}
00881  // Mouse X button down event
00882  case WM_XBUTTONDOWN :
00883  {
00884      Event Evt;
00885      Evt.Type = Event::MouseButtonPressed;
00886      Evt.MouseButton.Button = HIWORD(WParam) == XBUTTON1 ? Mouse::XButton1 : Mouse::XButton2;
00887      Evt.MouseButton.X = LOWORD(LParam);
00888      Evt.MouseButton.Y = HIWORD(LParam);
00889      SendEvent(Evt);
00890      break;
00891  }
00892
00893  // Mouse X button up event
00894  case WM_XBUTTONUP :
00895  {
00896      Event Evt;
00897      Evt.Type = Event::MouseButtonReleased;
00898      Evt.MouseButton.Button = HIWORD(WParam) == XBUTTON1 ? Mouse::XButton1 : Mouse::XButton2;
00899      Evt.MouseButton.X = LOWORD(LParam);
00900      Evt.MouseButton.Y = HIWORD(LParam);
00901      SendEvent(Evt);
00902      break;
00903  }
00904
00905  // Mouse move event
00906  case WM_MOUSEMOVE :
00907  {
00908      // Check if we need to generate a MouseEntered event
00909      if (!myIsCursorIn)
00910      {
00911          TRACKMOUSEEVENT MouseEventArgs;
00912          MouseEventArgs.cbSize = sizeof(TRACKMOUSEEVENT);
00913          MouseEventArgs.hwndTrack = myHandle;
00914          MouseEventArgs.dwFlags = TME_LEAVE;
00915          TrackMouseEvent(&MouseEventArgs);
00916      }
myIsCursorIn = true;

Event Evt;
Evt.Type = Event::MouseEntered;
SendEvent(Evt);
}

Event Evt;
Evt.Type = Event::MouseMoved;
Evt.MouseMove.X = LOWORD(LParam);
Evt.MouseMove.Y = HIWORD(LParam);
SendEvent(Evt);
break;

// Mouse leave event

Key::Code WindowImplWin32::VirtualKeyCodeToSF(WPARAM VirtualKey, LPARAM Flags)
{
    switch (VirtualKey) {
        case VK_SHIFT :
            static UINT LShift = MapVirtualKey(VK_LSHIFT, MAPVK_VK_TO_VSC);
UINT scancode = (Flags & (0xFF << 16)) >> 16;

return scancode == LShift ? Key::LShift :

// Check the "extended" flag to distinguish between left and right alt
// Check the "extended" flag to distinguish between left and right control
// Other keys are reported properly

case VK_APPS : return Key::Menu;
case VK_OEM_1 : return Key::SemiColon;
case VK_OEM_2 : return Key::Equal;
case VK_OEM_PLUS : return Key::LBracket;
case VK_OEM_6 : return Key::RBracket;
case VK_OEM_COMMA : return Key::Comma;
case VK_OEM_PERIOD : return Key::Period;
case VK_OEM_7 : return Key::Quote;
case VK_OEM_5 : return Key::BackSlash;
case VK_OEM_3 : return Key::Tilde;
case VK.Escape : return Key::Escape;
case VK_SPACE : return Key::Space;
case VK_Return : return Key::Return;
case VK_BACK : return Key::Back;
case VK_TAB : return Key::Tab;
case VK_PRIOR : return Key::PageUp;
case VK_NEXT : return Key::PageDown;
case VK_END : return Key::End;
case VK_HOME : return Key::Home;
case VK_INSERT : return Key::Insert;
case VK_DELETE : return Key::Delete;
case VK_ADD : return Key::Add;
case VK_SUBTRACT : return Key::Subtract;
case VK_MULTIPLY : return Key::Multiply;
case VK_DIVIDE : return Key::Divide;
case VK_PAUSE : return Key::Pause;
case VK_F1 : return Key::F1;
case VK_F2 : return Key::F2;
case VK_F3 : return Key::F3;
case VK_F4 : return Key::F4;
case VK_F5 : return Key::F5;
case VK_F6 : return Key::F6;
case VK_F7 : return Key::F7;
case VK_F8 : return Key::F8;
case VK_F9 : return Key::F9;
case VK_F10 : return Key::F10;
case VK_F11 : return Key::F11;
case VK_F12 : return Key::F12;
case VK_F13 : return Key::F13;
case VK_F14 : return Key::F14;
case VK_F15 : return Key::F15;
case VK_LEFT : return Key::Left;
case VK_RIGHT : return Key::Right;
case VK_UP : return Key::Up;
case VK_DOWN : return Key::Down;
case VK_NUMPAD0 : return Key::Numpad0;
case VK_NUMPAD1 : return Key::Numpad1;
case VK_NUMPAD2 : return Key::Numpad2;
case VK_NUMPAD3 : return Key::Numpad3;
case VK_NUMPAD4 : return Key::Numpad4;
case VK_NUMPAD5 : return Key::Numpad5;
case VK_NUMPAD6 : return Key::Numpad6;
case VK_NUMPAD7 : return Key::Numpad7;
case VK_NUMPAD8 : return Key::Numpad8;
case VK_NUMPAD9 : return Key::Numpad9;
case 'A' : return Key::A;
case 'Z' : return Key::Z;
case 'E' : return Key::E;
case 'R' : return Key::R;
01032     case 'T':       return Key::T;
01033     case 'Y':       return Key::Y;
01034     case 'U':       return Key::U;
01035     case 'I':       return Key::I;
01036     case 'O':       return Key::O;
01037     case 'P':       return Key::P;
01038     case 'Q':       return Key::Q;
01039     case 'S':       return Key::S;
01040     case 'D':       return Key::D;
01041     case 'F':       return Key::F;
01042     case 'G':       return Key::G;
01043     case 'H':       return Key::H;
01044     case 'J':       return Key::J;
01045     case 'K':       return Key::K;
01046     case 'L':       return Key::L;
01047     case 'M':       return Key::M;
01048     case 'W':       return Key::W;
01049     case 'X':       return Key::X;
01050     case 'C':       return Key::C;
01051     case 'V':       return Key::V;
01052     case 'B':       return Key::B;
01053     case 'N':       return Key::N;
01054     case '0':       return Key::Num0;
01055     case '1':       return Key::Num1;
01056     case '2':       return Key::Num2;
01057     case '3':       return Key::Num3;
01058     case '4':       return Key::Num4;
01059     case '5':       return Key::Num5;
01060     case '6':       return Key::Num6;
01061     case '7':       return Key::Num7;
01062     case '8':       return Key::Num8;
01063     case '9':       return Key::Num9;
01064           }
01065
01066     return Key::Code(0);
01067   }
01068
bool WindowImplWin32::HasUnicodeSupport()
{
    OSVERSIONINFO VersionInfo;
    ZeroMemory(&VersionInfo, sizeof(VersionInfo));
    VersionInfo.dwOSVersionInfoSize = sizeof(VersionInfo);

    if (GetVersionEx(&VersionInfo))
    {
        return VersionInfo.dwPlatformId == VER_PLATFORM_WIN32_NT;
    }
    else
    {
        return false;
    }
}

LRESULT CALLBACK WindowImplWin32::GlobalOnEvent(HWND Handle, UINT Message, WPARAM WParam, LPARAM LParam)
{
    // Associate handle and Window instance when the creation message is received
    if (Message == WM_CREATE)
    {
        // Get WindowImplWin32 instance (it was passed as the last argument of CreateWindow)
        long This = reinterpret_cast<long>(reinterpret_cast<long>(Handle));
        // Set as the "user data" parameter of the window
        SetWindowLongPtr(Handle, GWLP_USERDATA, This);
    }

    // Get the WindowImpl instance corresponding to the window handle
    WindowImplWin32* Window = reinterpret_cast<WindowImplWin32*>(Handle);
    // Forward the event to the appropriate function
    if (Window)
    {
        Window->ProcessEvent(Message, WParam, LParam);
    }
}
if (Window->myCallback)
    return CallWindowProc(reinterpret_cast<WNDPROC>(Window->myCallback), Handle, Message, WParam, LParam);
}

// We don't forward the WM_CLOSE message to
if (Message == WM_CLOSE)
    return 0;

static const bool HasUnicode = HasUnicodeSupport();
return HasUnicode ? DefWindowProcW(Handle, Message, WParam, LParam) :
                    DefWindowProcA(Handle, Message, WParam, LParam);

} // namespace priv
} // namespace sf

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)
//
// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.
//
// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
//   you must not claim that you wrote the original software.
//   If you use this software in a product, an acknowledgment
//   in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
//   and must not be misrepresented as being the original software.

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

#ifndef SFML_WINDOWIMPLWIN32_HPP
#define SFML_WINDOWIMPLWIN32_HPP

// Headers
#include <SFML/Window/Event.hpp>
#include <SFML/Window/WindowImpl.hpp>
#include <windows.h>
#include <string>

namespace sf
namespace priv
{
class WindowImplWin32 : public WindowImpl
{
public :
    WindowImplWin32();
    WindowImplWin32(WindowHandle Handle,
                     WindowSettings& Params);
    WindowImplWin32(VideoMode Mode,
                    const std::string& Title,
                    ~WindowImplWin32();
    static bool IsContextActive();
private :
    virtual void ProcessEvents();
    virtual void Display();
    virtual void SetActive(bool Active = true) const;
    virtual void UseVerticalSync(bool Enabled);
    virtual void ShowMouseCursor(bool Show);
    virtual void SetCursorPosition(unsigned int Left,
                                     unsigned int Top);
    virtual void SetPosition(int Left, int Top);
    virtual void SetSize(unsigned int Width,
                          unsigned int Height);
    virtual void Show(bool State);
virtual void EnableKeyRepeat(bool Enabled);

virtual void SetIcon(unsigned int Width, unsigned int Height);

void RegisterWindowClass();

void SwitchToFullscreen(const VideoMode& Mode);

void CreateContext(const VideoMode& Mode, WindowSettings& Params);

void Cleanup();

void ProcessEvent(UINT Message, WPARAM WParam, LPARAM LParam);

static Key::Code VirtualKeyCodeToSF(WPARAM VirtualKey, LPARAM Flags);

static bool HasUnicodeSupport();

static LRESULT CALLBACK GlobalOnEvent(HWND Handle, UINT Message, WPARAM WParam, LPARAM LParam);

// Static member data
static unsigned int ourWindowCount;
static const char* ourClassNameA;
static const wchar_t* ourClassNameW;
static WindowImplWin32* ourFullscreenWindow;

// Member data
HWND myHandle;
long myCallback;
HCURSOR myCursor;
HICON myIcon;
bool myKeyRepeatEnabled;
bool myIsCursorIn;
HDC myDeviceContext;
HGLRC myGLContext;
00251 } // namespace priv
00252
00253 } // namespace sf
00254
00255 #endif // SFML_WINDOWIMPLWIN32_HPP
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gom@gmail.com)

// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented;
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.

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

#include <SFML/Window/XXX/WindowImplXXX.hpp>
#include <SFML/Window/WindowStyle.hpp>
#include <GL/gl.h>

namespace sf
{
namespace priv
{
WindowImplXXX::WindowImplXXX();
}
{
  // Create a dummy window (with the fewest attributes -- it's just to have a valid support for an OpenGL context)

  // Initialize myWidth and myHeight members from base class with the window size

  // Create an OpenGL context in this window and make it active
}

WindowImplXXX::WindowImplXXX(WindowHandle Handle)
{
  // Make sure we'll be able to catch all the events of the given window

  // Initialize myWidth and myHeight members from base class with the window size

  // Create an OpenGL context in this window and make it active
}

WindowImplXXX::WindowImplXXX(VideoMode Mode, const
{
  // Create a new window with given size, title

  // Initialize myWidth and myHeight members from base class with the window size

  // Create an OpenGL context in this window and make it active
}

WindowImplXXX::~WindowImplXXX()
{
  // Destroy the OpenGL context, the window and every resource allocated by this class
}

bool WindowImplXXX::IsContextActive()
{

void WindowImplXXX::Display()
{
    // Swap OpenGL buffers (should be a call to xxxSwapBuffers)
}

void WindowImplXXX::ProcessEvents()
{
    // Process every event for this window
    // Generate a sf::Event and call SendEvent()
}

void WindowImplXXX::SetActive(bool Active) const
{
    // Bind / unbind OpenGL context (should be a call to xxxMakeCurrent)
}

void WindowImplXXX::UseVerticalSync(bool Enabled)
{
    // Activate / deactivate vertical synchronization
    // usually using an OpenGL extension (should
}

void WindowImplXXX::ShowMouseCursor(bool Show)
{
    // Show or hide the system cursor in this window
}
void WindowImplXXX::SetCursorPosition(unsigned int)
{
    // Change the cursor position (Left and Top are relative to this window)
}

void WindowImplXXX::SetPosition(int Left, int Top)
{
    // Change the window position
}

void WindowImplWin32::SetSize(unsigned int Width)
{
    // Change the window size
}

void WindowImplXXX::Show(bool State)
{
    // Show or hide the window
}

void WindowImplXXX::EnableKeyRepeat(bool Enabled)
{
    // Enable or disable automatic key-repeat for keydown events
}

void WindowImplXXX::SetIcon(unsigned int Width, unsigned int Height)
{
    // Change all the necessary icons of the window with the provided array of 32 bits RGBA pixels
}

/*================================*/
- If the requested level of anti-aliasing is not supported and is greater than 2, try with 2
  --> if level 2 fails, disable anti-aliasing
  --> it's important not to generate an error if anti-aliasing is not supported
- Use a matching pixel mode, or the best of all
  You should use the function EvaluateConfig to
- Don't forget to fill Params (see constructors)
- IMPORTANT: all OpenGL contexts must be shared (usually a call to xxxShareLists)

- Process any event matching with the ones in sf::Event::EventType
  --> Create a sf::Event, fill the members corresponding to the event type
  --> No need to handle joystick events, they are handled by WindowImpl::ProcessJoystickEvents
  --> Event::TextEntered must provide UTF-16 characters (see http://www.unicode.org/Public/PROGRAMS/CVTUTF/ for unicode conversions)
  --> Don't forget to process any destroy-like event
- Use SendEvent function from base class to propagate the created events

} // namespace priv
} // namespace sf

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
// SFML - Simple and Fast Multimedia Library
// Copyright (C) 2007-2009 Laurent Gomila (laurent.gomila@gmail.com)

// This software is provided 'as-is', without any implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.

// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:

// 1. The origin of this software must not be misrepresented,
//    you must not claim that you wrote the original software.
//    If you use this software in a product, an acknowledgment
//    in the product documentation would be appreciated but is not required.

// 2. Altered source versions must be plainly marked as such,
//    and must not be misrepresented as being the original software.

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

#ifndef SFML_WINDOWIMPLXXX_HPP
#define SFML_WINDOWIMPLXXX_HPP

// Headers
#include <SFML/Window/Event.hpp>
#include <SFML/Window/WindowImpl.hpp>
#include <string>

namespace sf {


namespace priv {

class WindowImplXXX : public WindowImpl {

public:

    WindowImplXXX();

    WindowImplXXX(WindowHandle Handle, WindowSettings& Params);

    WindowImplXXX(VideoMode Mode, const std::string& Title);

    ~WindowImplXXX();

    static bool IsContextActive();

private:

    virtual void Display();

    virtual void ProcessEvents();

    virtual void SetActive(bool Active = true);

    virtual bool IsActive() const;

    virtual void UseVerticalSync(bool Enabled);

    virtual void ShowMouseCursor(bool Show);

    virtual void SetCursorPosition(unsigned int Left, unsigned int Top);

    virtual void SetPosition(int Left, int Top);

    virtual void SetSize(unsigned int Width, unsigned int Height);

    virtual void Show(bool State);
}
virtual void EnableKeyRepeat(bool Enabled);

virtual void SetIcon(unsigned int Width, unsigned int Height);

// namespace priv

} // namespace sf

# endif // SFML_WINDOWIMPLXXX_HPP
- b -

- **Bind()** : `sf::Image`, `sf::SocketUDP`
- c -

- CanCapture() : sf::SoundRecorder
- CanUsePostFX() : sf::PostFX
- Capture() : sf::RenderWindow
- ChangeDirectory() : sf::Ftp
- Circle() : sf::Shape
- Clear() : sf::Packet, sf::Selector< Type >, sf::SelectorBase, sf::RenderTarget
- Clock() : sf::Clock
- Close() : sf::SocketTCP, sf::SocketUDP, sf::SocketHelper, sf::Window
- Color() : sf::Color
- Connect() : sf::SocketTCP, sf::Ftp
- Contains() : sf::Rect< T >
- Context() : sf::Context
- ConvertCoords() : sf::RenderWindow
- Copy() : sf::Image
- CopyScreen() : sf::Image
- Create() : sf::Window, sf::Image, sf::Window
- CreateMaskFromColor() : sf::Image
### Class Members

- **DeleteDirectory()**: `sf::Ftp`
- **DeleteFile()**: `sf::Ftp`
- **DirectoryResponse()**: `sf::Ftp::Ftp::DirectoryResponse`
- **Disconnect()**: `sf::Ftp`
- **Display()**: `sf::Window`
- **Download()**: `sf::Ftp`
- **Draw()**: `sf::RenderTarget`
- **Drawable()**: `sf::Drawable`
- e -

- **EnableFill**() : *sf::Shape*
- **EnableKeyRepeat**() : *sf::Window*
- **EnableOutline**() : *sf::Shape*
- **EndOfPacket**() : *sf::Packet*
- f -

- **FlipX()** : `sf::Sprite`
- **FlipY()** : `sf::Sprite`
- **Font()** : `sf::Font`

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by [doxygen 1.5.2](http://www.doxygen.org) ::
- g -

- Get4x4Elements() : `sf::Matrix3`
- GetAttenuation() : `sf::Sound`
- GetBlendMode() : `sf::Drawable`
- GetBody() : `sf::Http::Http::Response`
- GetBuffer() : `sf::Sound`, `sf::SoundBufferRecorder`
- GetCenter() : `sf::View`, `sf::Drawable`
- GetChannelsCount() : `sf::SoundBuffer`, `sf::SoundStream`
- GetCharacterPos() : `sf::String`
- GetCharacterSize() : `sf::Font`
- GetColor() : `sf::Drawable`
- GetCount() : `sf::Ftp::Ftp::ListingResponse`
- GetData() : `sf::Packet`
- GetDataSize() : `sf::Packet`
- GetDefaultFont() : `sf::Font`
- GetDefaultView() : `sf::RenderTarget`
- GetDesktopMode() : `sf::VideoMode`
- GetDirectory() : sf::Ftp::Ftp::DirectoryResponse
- GetDirectoryListing() : sf::Ftp
- GetDuration() : sf::Music, sf::SoundBuffer
- GetElapsedTime() : sf::Clock
- GetErrorStatus() : sf::SocketHelper
- GetEvent() : sf::Window
- GetField() : sf::Http::Http::Response
- GetFilename() : sf::Ftp::Ftp::ListingResponse
- GetFont() : sf::String
- GetFrameTime() : sf::Window
- GetGlobal() : sf::Context
- GetGlobalVolume() : sf::Listener
- GetGlyph() : sf::Font
- GetHalfSize() : sf::View
- GetHeight() : sf::Image, sf::Rect<T>, sf::RenderTarget, sf::RenderWindow, sf::Window
- GetImage() : sf::Font, sf::Sprite
- GetInput() : sf::Window
- GetInverse() : sf::Matrix3
- GetInverseMatrix() : sf::Drawable
- GetJoystickAxis() : sf::Input
- GetLocalAddress() : sf::IPAddress
- GetLoop() : sf::Sound, sf::SoundStream
- GetMajorHttpVersion() : sf::Http::Http::Response
- GetMatrix() : sf::Drawable
- GetMessage() : sf::Ftp::Ftp::Response
- GetMinDistance() : sf::Sound
- GetMinorHttpVersion() : sf::Http::Http::Response
- GetMode() : sf::VideoMode
- GetModesCount() : sf::VideoMode
- GetMouseX() : sf::Input
- GetMouseY() : sf::Input
- GetNbPoints() : sf::Shape
- GetOutlineWidth() : sf::Shape
- GetPitch() : sf::Sound
- GetPixel() : sf::Image, sf::Sprite
- GetPixelsPtr() : sf::Image
- GetPlayingOffset() : sf::Sound, sf::SoundStream
- GetPointColor() : sf::Shape
- GetPointOutlineColor() : sf::Shape
- GetPointPosition() : sf::Shape
- GetPort() : sf::SocketUDP
- GetPosition() : sf::Listener, sf::Drawable, sf::Sound
- GetPublicAddress() : sf::IPAddress
- GetRect() : sf::String, sf::View
- GetRotation() : sf::Drawable
- GetSampleRate() : sf::SoundBuffer, sf::SoundStream, sf::SoundRecorder
- GetSamples() : sf::SoundBuffer
- GetSamplesCount() : sf::SoundBuffer
- GetScale() : sf::Drawable
- GetSeed() : sf::Randomizer
- GetSettings() : sf::Window
- GetSize() : sf::String, sf::Sprite
- GetSocketReady() : sf::SelectorBase, sf::Selector< Type >
- GetStatus() : sf::SoundStream, sf::Http::Http::Response, sf::Sound, sf::Ftp::Ftp::Response
- GetStyle() : sf::String
- GetSubRect() : sf::Sprite
- GetTarget() : sf::Listener
- GetTexCoords() : sf::Image
- GetText() : sf::String
- GetUTF16Length() : sf::Unicode
- GetUTF32Length() : sf::Unicode
- GetUTF8Length() : sf::Unicode
- GetValidTextureSize(): sf::Image
- GetView(): sf::RenderTarget
- GetVolume(): sf::Sound
- GetWidth(): sf::Window, sf::RenderTarget, sf::Rect< T >, sf::Image, sf::RenderWindow
- GetWorkingDirectory(): sf::Ftp
- Glyph(): sf::Glyph
- h -

- **Http() : sf::Http**
- i -

- Image() : sf::Image
- Initialize() : sf::SoundStream, sf::RenderTarget
- Input() : sf::Input
- Intersects() : sf::Rect< T >
- InvalidSocket() : sf::SocketHelper
- IPAddress() : sf::IPAddress
- IsContextActive() : sf::Context
- IsJoystickButtonDown() : sf::Input
- IsKeyDown() : sf::Input
- IsMouseButtonDown() : sf::Input
- IsOk() : sf::Ftp::Ftp::Response
- IsOpened() : sf::Window
- IsRelativeToListener() : sf::Sound
- IsSmooth() : sf::Image
- IsValid() : sf::SocketTCP, sf::IPAddress, sf::VideoMode, sf::SocketUDP
- k -

- KeepAlive() : sf::Ftp

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
- I -

- Launch(): `sf::Thread`
- Line(): `sf::Shape`
- Listen(): `sf::SocketTCP`
- ListingResponse(): `sf::Ftp::Ftp::ListingResponse`
- LoadFromFile(): `sf::Image`, `sf::PostFX`, `sf::SoundBuffer`
  , `sf::Font`
- LoadFromMemory(): `sf::SoundBuffer`, `sf::Font`,
  `sf::Image`, `sf::PostFX`
- LoadFromPixels(): `sf::Image`
- LoadFromSamples(): `sf::SoundBuffer`
- Lock(): `sf::Mutex`, `sf::Lock`
- Login(): `sf::Ftp`

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
- m -

- **MakeDirectory()** : `sf::Ftp`
- **Matrix3()** : `sf::Matrix3`
- **Move()** : `sf::View`, `sf::Drawable`, `sf::View`
- **Music()** : `sf::Music`
- **Mutex()** : `sf::Mutex`
- n -

- NonCopyable() : sf::NonCopyable
- o -

- Offset() : `sf::Rect< T >`
- OnEvent() : `sf::WindowListener`
- OnResourceDestroyed() : `sf::ResourcePtr< T >`
- OpenFromFile() : `sf::Music`
- OpenFromMemory() : `sf::Music`
- operator bool() : `sf::Packet`
- operator const T *() : `sf::ResourcePtr< T >`
- operator std::string() : `sf::Unicode::Unicode::Text`
- operator!=( ) : `sf::Color`, `sf::IPAddress`, `sf::SocketTCP`, `sf::SocketUDP`, `sf::VideoMode`
- operator() : `sf::Matrix3`
- operator*() : `sf::ResourcePtr< T >`, `sf::Matrix3`
- operator*=( ) : `sf::Color`, `sf::Matrix3`
- operator+=() : `sf::Color`
- operator->() : `sf::ResourcePtr< T >`
- operator<() : `sf::SocketUDP`, `sf::IPAddress`, 
sf::SocketTCP

- operator<<() : sf::Packet
- operator<=( ) : sf::IPAddress
- operator=( ) : sf::ResourcePtr< T >, sf::PostFX, sf::ResourcePtr< T >, sf::SoundBuffer, sf::Resource< T >, sf::Sound, sf::Image
- operator==() : sf::IPAddress, sf::SocketUDP, sf::SocketTCP, sf::Color, sf::VideoMode
- operator>() : sf::IPAddress
- operator>=( ) : sf::IPAddress
- operator>>() : sf::Packet
- p -

- Packet() : `sf::Packet`
- ParentDirectory() : `sf::Ftp`
- Pause() : `sf::Sound`
- Play() : `sf::Sound, sf::SoundStream`
- PostFX() : `sf::PostFX`
- PreserveOpenGLStates() : `sf::RenderTarget`

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::

Documentation generated by doxygen 1.5.2 ::
- r -

- Random() : **sf::Randomizer**
- Receive() : **sf::SocketTCP**, **sf::SocketUDP**, **sf::SocketTCP**, **sf::SocketUDP**
- Rect() : **sf::Rect<T>**
- Rectangle() : **sf::Shape**
- Remove() : **sf::Selector<Type>**, **sf::SelectorBase**
- RenameFile() : **sf::Ftp**
- Render() : **sf::Sprite**, **sf::String**, **sf::PostFX**, **sf::Shape**
- RenderTarget() : **sf::RenderTarget**
- RenderWindow() : **sf::RenderWindow**
- Request() : **sf::Http::Http::Request**
- Reset() : **sf::Clock**
- ResetBuffer() : **sf::Sound**
- Resize() : **sf::Sprite**
- Resource() : **sf::Resource<T>**
- ResourcePtr() : **sf::ResourcePtr<T>**
- **Response()**: `sf::Ftp::Ftp::Response`, `sf::Http::Http::Response`
- **Rotate()**: `sf::Drawable`
- S -

- SaveToFile() : `sf::SoundBuffer`, `sf::Image`
- Scale() : `sf::Drawable`
- SelectorBase() : `sf::SelectorBase`
- Send() : `sf::SocketUDP`, `sf::SocketTCP`
- SendRequest() : `sf::Http`
- SetActive() : `sf::Context`, `sf::Window`
- SetAttenuation() : `sf::Sound`
- SetBlendMode() : `sf::Drawable`
- SetBlocking() : `sf::SocketUDP`, `sf::SocketHelper`, `sf::SocketTCP`
- SetBody() : `sf::Http::Http::Request`
- SetBuffer() : `sf::Sound`
- SetCenter() : `sf::Drawable`, `sf::View`
- SetColor() : `sf::Drawable`
- SetCursorPosition() : `sf::Window`
- SetField() : `sf::Http::Http::Request`
- SetFont() : sf::String
- SetFramerateLimit() : sf::Window
- SetFromRect() : sf::View
- SetFromTransformations() : sf::Matrix3
- SetGlobalVolume() : sf::Listener
- SetHalfSize() : sf::View
- SetHost() : sf::Http
- SetHttpVersion() : sf::Http::Http::Request
- SetIcon() : sf::Window
- SetImage() : sf::Sprite
- SetJoystickThreshold() : sf::Window
- SetLoop() : sf::Sound, sf::SoundStream
- SetMethod() : sf::Http::Http::Request
- SetMinDistance() : sf::Sound
- SetOutlineWidth() : sf::Shape
- SetParameter() : sf::PostFX
- SetPitch() : sf::Sound
- SetPixel() : sf::Image
- SetPlayingOffset() : sf::Sound
- SetPointColor() : sf::Shape
- SetPointOutlineColor() : sf::Shape
- SetPointPosition() : sf::Shape
- SetPosition() : sf::Drawable, sf::Listener, sf::Sound, sf::Drawable, sf::Window
- SetRelativeToListener() : sf::Sound
- SetRotation() : sf::Drawable
- SetScale() : sf::Drawable
- SetScaleX() : sf::Drawable
- SetScaleY() : sf::Drawable
- SetSeed() : sf::Randomizer
- SetSize() : sf::String, sf::Window
- SetSmooth() : sf::Image
- SetStyle() : sf::String
- SetSubRect() : sf::Sprite
- SetTarget() : sf::Listener
- SetText() : sf::String
- SetTexture() : sf::PostFX
- SetURI() : sf::Http::Http::Request
- SetView() : sf::RenderTarget
- SetVolume() : sf::Sound
- SetX() : sf::Drawable
- SetY() : sf::Drawable
- Shape() : sf::Shape
- Show() : sf::Window
- ShowMouseCursor() : sf::Window
- SocketTCP() : sf::SocketTCP
- SocketUDP() : sf::SocketUDP
- Sound() : sf::Sound
- SoundBuffer() : sf::SoundBuffer
- SoundRecorder() : sf::SoundRecorder
- SoundStream() : sf::SoundStream
- Sprite() : sf::Sprite
- Start() : sf::SoundRecorder
- Stop() : sf::Sound, sf::SoundStream, sf::SoundRecorder
- String() : sf::String
- t -

- Terminate() : \texttt{sf::Thread}
- Text() : \texttt{sf::Unicode::Unicode::Text}
- Thread() : \texttt{sf::Thread}
- ToInteger() : \texttt{sf::IPAddress}
- ToString() : \texttt{sf::IPAddress}
- Transform() : \texttt{sf::Matrix3}
- TransformToGlobal() : \texttt{sf::Drawable}
- TransformToLocal() : \texttt{sf::Drawable}
- u -

- **Unbind()**: `sf::SocketUDP`
- **Unlock()**: `sf::Mutex`
- **Upload()**: `sf::Ftp`
- **UseVerticalSync()**: `sf::Window`
- **UTF16ToUTF32()**: `sf::Unicode`
- **UTF16ToUTF8()**: `sf::Unicode`
- **UTF32ToANSI()**: `sf::Unicode`
- **UTF32ToUTF16()**: `sf::Unicode`
- **UTF32ToUTF8()**: `sf::Unicode`
- **UTF8ToUTF16()**: `sf::Unicode`
- **UTF8ToUTF32()**: `sf::Unicode`
- V -

- Vector2() : sf::Vector2< T >
- Vector3() : sf::Vector3< T >
- VideoMode() : sf::VideoMode
- View() : sf::View
- W -

- Wait() : sf::Selector< Type >, sf::SelectorBase, sf::Thread
- Window() : sf::Window
- WindowSettings() : sf::WindowSettings
- Z -

- Zoom() : sf::View

:: Copyright © 2007-2008 Laurent Gomila, all rights reserved ::
Documentation generated by doxygen 1.5.2 ::
• ~AudioResource() : sf::AudioResource
• ~Context() : sf::Context
• ~Drawable() : sf::Drawable
• ~Ftp() : sf::Ftp
• ~Image() : sf::Image
• ~Lock() : sf::Lock
• ~Music() : sf::Music
• ~Mutex() : sf::Mutex
• ~Packet() : sf::Packet
• ~PostFX() : sf::PostFX
• ~RenderTarget() : sf::RenderTarget
• ~RenderWindow() : sf::RenderWindow
• ~Resource() : sf::Resource< T >
• ~ResourcePtr() : sf::ResourcePtr< T >
• ~Sound() : sf::Sound
• ~SoundBuffer() : sf::SoundBuffer
• ~SoundRecorder() : sf::SoundRecorder
• ~SoundStream() : sf::SoundStream
• ~Thread() : sf::Thread
• ~Window() : sf::Window
• ~WindowListener() : sf::WindowListener