SFML

* [Main Page](http://docs.google.com/index.htm)
* [Modules](http://docs.google.com/modules.htm)
* [Classes](http://docs.google.com/annotated.htm)
* [Files](http://docs.google.com/files.htm)
* [Class List](http://docs.google.com/annotated.htm)
* [Class Index](http://docs.google.com/classes.htm)
* [Class Hierarchy](http://docs.google.com/hierarchy.htm)
* [Class Members](http://docs.google.com/functions.htm)
* **sf**
* [RenderTexture](http://docs.google.com/classsf_1_1RenderTexture.htm)

[Public Member Functions](#_gjdgxs) | [Protected Member Functions](#_30j0zll) | [List of all members](http://docs.google.com/classsf_1_1RenderTexture-members.htm)

sf::RenderTexture Class Reference

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

Target for off-screen 2D rendering into a texture. [More...](http://docs.google.com/classsf_1_1RenderTexture.htm#details)

#include <[RenderTexture.hpp](http://docs.google.com/RenderTexture_8hpp_source.htm)>

Inheritance diagram for sf::RenderTexture:



| Public Member Functions | |
| --- | --- |
|  | [RenderTexture](http://docs.google.com/classsf_1_1RenderTexture.htm#a19ee6e5b4c40ad251803389b3953a9c6) () |
|  | Default constructor. |
|  | |
| virtual | [~RenderTexture](http://docs.google.com/classsf_1_1RenderTexture.htm#a94b84ab9335be84d2a014c964d973304) () |
|  | Destructor. |
|  | |
| bool | [create](http://docs.google.com/classsf_1_1RenderTexture.htm#aefbb76eb3b87e368ab974b2660931ccb) (unsigned int width, unsigned int height, bool depthBuffer=false) |
|  | Create the render-texture. |
|  | |
| void | [setSmooth](http://docs.google.com/classsf_1_1RenderTexture.htm#af08991e63c6020865dd07b20e27305b6) (bool smooth) |
|  | Enable or disable texture smoothing. |
|  | |
| bool | [isSmooth](http://docs.google.com/classsf_1_1RenderTexture.htm#ae385f4f4dbd2af50fb11947bf0bcb83d) () const |
|  | Tell whether the smooth filtering is enabled or not. |
|  | |
| void | [setRepeated](http://docs.google.com/classsf_1_1RenderTexture.htm#af8f97b33512bf7d5b6be3da6f65f7365) (bool repeated) |
|  | Enable or disable texture repeating. |
|  | |
| bool | [isRepeated](http://docs.google.com/classsf_1_1RenderTexture.htm#ae480a2ec7ee166afa50232e634d2668c) () const |
|  | Tell whether the texture is repeated or not. |
|  | |
| bool | [setActive](http://docs.google.com/classsf_1_1RenderTexture.htm#a5da95ecdbce615a80bb78399012508cf) (bool active=true) |
|  | Activate of deactivate the render-texture for rendering. |
|  | |
| void | [display](http://docs.google.com/classsf_1_1RenderTexture.htm#af92886d5faef3916caff9fa9ab32c555) () |
|  | Update the contents of the target texture. |
|  | |
| virtual [Vector2u](http://docs.google.com/classsf_1_1Vector2.htm) | [getSize](http://docs.google.com/classsf_1_1RenderTexture.htm#a757ba45ec7a7deefcaef717049b00b8c) () const |
|  | Return the size of the rendering region of the texture. |
|  | |
| const [Texture](http://docs.google.com/classsf_1_1Texture.htm) & | [getTexture](http://docs.google.com/classsf_1_1RenderTexture.htm#a95bc5152c497066d31fdc57da8e17678) () const |
|  | Get a read-only reference to the target texture. |
|  | |
| void | [clear](http://docs.google.com/classsf_1_1RenderTarget.htm#a6bb6f0ba348f2b1e2f46114aeaf60f26) (const [Color](http://docs.google.com/classsf_1_1Color.htm) &color=[Color](http://docs.google.com/classsf_1_1Color.htm)(0, 0, 0, 255)) |
|  | Clear the entire target with a single color. |
|  | |
| void | [setView](http://docs.google.com/classsf_1_1RenderTarget.htm#a063db6dd0a14913504af30e50cb6d946) (const [View](http://docs.google.com/classsf_1_1View.htm) &view) |
|  | Change the current active view. |
|  | |
| const [View](http://docs.google.com/classsf_1_1View.htm) & | [getView](http://docs.google.com/classsf_1_1RenderTarget.htm#a98f721cc6dc11478922427fedfb2288b) () const |
|  | Get the view currently in use in the render target. |
|  | |
| const [View](http://docs.google.com/classsf_1_1View.htm) & | [getDefaultView](http://docs.google.com/classsf_1_1RenderTarget.htm#a718b1aa6296bf855171699cc18251ced) () const |
|  | Get the default view of the render target. |
|  | |
| [IntRect](http://docs.google.com/classsf_1_1Rect.htm) | [getViewport](http://docs.google.com/classsf_1_1RenderTarget.htm#aae035b0d45f87a0da2a28a0de6ba1086) (const [View](http://docs.google.com/classsf_1_1View.htm) &view) const |
|  | Get the viewport of a view, applied to this render target. |
|  | |
| [Vector2f](http://docs.google.com/classsf_1_1Vector2.htm) | [mapPixelToCoords](http://docs.google.com/classsf_1_1RenderTarget.htm#a2b0cab0e4c6af29d4efaba149d28116d) (const [Vector2i](http://docs.google.com/classsf_1_1Vector2.htm) &point) const |
|  | Convert a point from target coordinates to world coordinates, using the current view. |
|  | |
| [Vector2f](http://docs.google.com/classsf_1_1Vector2.htm) | [mapPixelToCoords](http://docs.google.com/classsf_1_1RenderTarget.htm#a46eb08f775dd1420d6207ea87dde6e54) (const [Vector2i](http://docs.google.com/classsf_1_1Vector2.htm) &point, const [View](http://docs.google.com/classsf_1_1View.htm) &view) const |
|  | Convert a point from target coordinates to world coordinates. |
|  | |
| [Vector2i](http://docs.google.com/classsf_1_1Vector2.htm) | [mapCoordsToPixel](http://docs.google.com/classsf_1_1RenderTarget.htm#aa0c11e1989573f2cce64c621205f8e83) (const [Vector2f](http://docs.google.com/classsf_1_1Vector2.htm) &point) const |
|  | Convert a point from world coordinates to target coordinates, using the current view. |
|  | |
| [Vector2i](http://docs.google.com/classsf_1_1Vector2.htm) | [mapCoordsToPixel](http://docs.google.com/classsf_1_1RenderTarget.htm#a7a2d427bdb9bd8f9f456fcf82813aa60) (const [Vector2f](http://docs.google.com/classsf_1_1Vector2.htm) &point, const [View](http://docs.google.com/classsf_1_1View.htm) &view) const |
|  | Convert a point from world coordinates to target coordinates. |
|  | |
| void | [draw](http://docs.google.com/classsf_1_1RenderTarget.htm#a12417a3bcc245c41d957b29583556f39) (const [Drawable](http://docs.google.com/classsf_1_1Drawable.htm) &drawable, const [RenderStates](http://docs.google.com/classsf_1_1RenderStates.htm) &states=[RenderStates::Default](http://docs.google.com/classsf_1_1RenderStates.htm#ad29672df29f19ce50c3021d95f2bb062)) |
|  | Draw a drawable object to the render-target. |
|  | |
| void | [draw](http://docs.google.com/classsf_1_1RenderTarget.htm#ab636d7363f6681077361ee274ba89a8d) (const [Vertex](http://docs.google.com/classsf_1_1Vertex.htm) \*vertices, unsigned int vertexCount, [PrimitiveType](http://docs.google.com/group__graphics.htm#ga5ee56ac1339984909610713096283b1b) type, const [RenderStates](http://docs.google.com/classsf_1_1RenderStates.htm) &states=[RenderStates::Default](http://docs.google.com/classsf_1_1RenderStates.htm#ad29672df29f19ce50c3021d95f2bb062)) |
|  | Draw primitives defined by an array of vertices. |
|  | |
| void | [pushGLStates](http://docs.google.com/classsf_1_1RenderTarget.htm#a8d1998464ccc54e789aaf990242b47f7) () |
|  | Save the current OpenGL render states and matrices. |
|  | |
| void | [popGLStates](http://docs.google.com/classsf_1_1RenderTarget.htm#ad5a98401113df931ddcd54c080f7aa8e) () |
|  | Restore the previously saved OpenGL render states and matrices. |
|  | |
| void | [resetGLStates](http://docs.google.com/classsf_1_1RenderTarget.htm#aac7504990d27dada4bfe3c7866920765) () |
|  | Reset the internal OpenGL states so that the target is ready for drawing. |
|  | |

| Protected Member Functions | |
| --- | --- |
| void | [initialize](http://docs.google.com/classsf_1_1RenderTarget.htm#af530274b34159d644e509b4b4dc43eb7) () |
|  | Performs the common initialization step after creation. |
|  | |

## Detailed Description

Target for off-screen 2D rendering into a texture.

[sf::RenderTexture](http://docs.google.com/classsf_1_1RenderTexture.htm) is the little brother of [sf::RenderWindow](http://docs.google.com/classsf_1_1RenderWindow.htm).

It implements the same 2D drawing and OpenGL-related functions (see their base class [sf::RenderTarget](http://docs.google.com/classsf_1_1RenderTarget.htm) for more details), the difference is that the result is stored in an off-screen texture rather than being show in a window.

Rendering to a texture can be useful in a variety of situations:

* precomputing a complex static texture (like a level's background from multiple tiles)
* applying post-effects to the whole scene with shaders
* creating a sprite from a 3D object rendered with OpenGL
* etc.

Usage example:

// Create a new render-window

[sf::RenderWindow](http://docs.google.com/classsf_1_1RenderWindow.htm) window([sf::VideoMode](http://docs.google.com/classsf_1_1VideoMode.htm)(800, 600), "SFML window");

// Create a new render-texture

[sf::RenderTexture](http://docs.google.com/classsf_1_1RenderTexture.htm) texture;

if (!texture.[create](http://docs.google.com/classsf_1_1RenderTexture.htm#aefbb76eb3b87e368ab974b2660931ccb)(500, 500))

return -1;

// The main loop

while (window.isOpen())

{

// Event processing

// ...

// Clear the whole texture with red color

texture.[clear](http://docs.google.com/classsf_1_1RenderTarget.htm#a6bb6f0ba348f2b1e2f46114aeaf60f26)([sf::Color::Red](http://docs.google.com/classsf_1_1Color.htm#a127dbf55db9c07d0fa8f4bfcbb97594a));

// Draw stuff to the texture

texture.[draw](http://docs.google.com/classsf_1_1RenderTarget.htm#a12417a3bcc245c41d957b29583556f39)(sprite); // sprite is a sf::Sprite

texture.[draw](http://docs.google.com/classsf_1_1RenderTarget.htm#a12417a3bcc245c41d957b29583556f39)(shape); // shape is a sf::Shape

texture.[draw](http://docs.google.com/classsf_1_1RenderTarget.htm#a12417a3bcc245c41d957b29583556f39)(text); // text is a sf::Text

// We're done drawing to the texture

texture.[display](http://docs.google.com/classsf_1_1RenderTexture.htm#af92886d5faef3916caff9fa9ab32c555)();

// Now we start rendering to the window, clear it first

window.clear();

// Draw the texture

[sf::Sprite](http://docs.google.com/classsf_1_1Sprite.htm) sprite(texture.[getTexture](http://docs.google.com/classsf_1_1RenderTexture.htm#a95bc5152c497066d31fdc57da8e17678)());

window.draw(sprite);

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

window.display();

}

Like [sf::RenderWindow](http://docs.google.com/classsf_1_1RenderWindow.htm), [sf::RenderTexture](http://docs.google.com/classsf_1_1RenderTexture.htm) is still able to render direct OpenGL stuff. It is even possible to mix together OpenGL calls and regular SFML drawing commands. If you need a depth buffer for 3D rendering, don't forget to request it when calling [RenderTexture::create](http://docs.google.com/classsf_1_1RenderTexture.htm#aefbb76eb3b87e368ab974b2660931ccb).

See Also[sf::RenderTarget](http://docs.google.com/classsf_1_1RenderTarget.htm), [sf::RenderWindow](http://docs.google.com/classsf_1_1RenderWindow.htm), [sf::View](http://docs.google.com/classsf_1_1View.htm), [sf::Texture](http://docs.google.com/classsf_1_1Texture.htm)

Definition at line [47](http://docs.google.com/RenderTexture_8hpp_source.htm#l00047) of file [RenderTexture.hpp](http://docs.google.com/RenderTexture_8hpp_source.htm).

## Constructor & Destructor Documentation

| sf::RenderTexture::RenderTexture | ( |  | ) |  |
| --- | --- | --- | --- | --- |

Default constructor.

Constructs an empty, invalid render-texture. You must call create to have a valid render-texture.

See Also[create](http://docs.google.com/classsf_1_1RenderTexture.htm#aefbb76eb3b87e368ab974b2660931ccb)

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

Destructor.

## Member Function Documentation

| | void sf::RenderTarget::clear | ( | const [Color](http://docs.google.com/classsf_1_1Color.htm) & | *color* = [Color](http://docs.google.com/classsf_1_1Color.htm)(0, 0, 0, 255) | ) |  | | --- | --- | --- | --- | --- | --- | | inherited |
| --- | --- | --- | --- | --- | --- | --- | --- |

Clear the entire target with a single color.

This function is usually called once every frame, to clear the previous contents of the target.

Parameters

| color | Fill color to use to clear the render target |
| --- | --- |

| bool sf::RenderTexture::create | ( | unsigned int | *width*, |
| --- | --- | --- | --- |
|  |  | unsigned int | *height*, |
|  |  | bool | *depthBuffer* = false |
|  | ) |  |  |

Create the render-texture.

Before calling this function, the render-texture is in an invalid state, thus it is mandatory to call it before doing anything with the render-texture. The last parameter, *depthBuffer*, is useful if you want to use the render-texture for 3D OpenGL rendering that requires a depth-buffer. Otherwise it is unnecessary, and you should leave this parameter to false (which is its default value).

Parameters

| width | Width of the render-texture |
| --- | --- |
| height | Height of the render-texture |
| depthBuffer | Do you want this render-texture to have a depth buffer? |

ReturnsTrue if creation has been successful

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

Update the contents of the target texture.

This function updates the target texture with what has been drawn so far. Like for windows, calling this function is mandatory at the end of rendering. Not calling it may leave the texture in an undefined state.

| | void sf::RenderTarget::draw | ( | const [Drawable](http://docs.google.com/classsf_1_1Drawable.htm) & | *drawable*, | | --- | --- | --- | --- | |  |  | const [RenderStates](http://docs.google.com/classsf_1_1RenderStates.htm) & | *states* = [RenderStates::Default](http://docs.google.com/classsf_1_1RenderStates.htm#ad29672df29f19ce50c3021d95f2bb062) | |  | ) |  |  | | inherited |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

Draw a drawable object to the render-target.

Parameters

| drawable | Object to draw |
| --- | --- |
| states | Render states to use for drawing |

| | void sf::RenderTarget::draw | ( | const [Vertex](http://docs.google.com/classsf_1_1Vertex.htm) \* | *vertices*, | | --- | --- | --- | --- | |  |  | unsigned int | *vertexCount*, | |  |  | [PrimitiveType](http://docs.google.com/group__graphics.htm#ga5ee56ac1339984909610713096283b1b) | *type*, | |  |  | const [RenderStates](http://docs.google.com/classsf_1_1RenderStates.htm) & | *states* = [RenderStates::Default](http://docs.google.com/classsf_1_1RenderStates.htm#ad29672df29f19ce50c3021d95f2bb062) | |  | ) |  |  | | inherited |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

Draw primitives defined by an array of vertices.

Parameters

| vertices | Pointer to the vertices |
| --- | --- |
| vertexCount | Number of vertices in the array |
| type | Type of primitives to draw |
| states | Render states to use for drawing |

| | const [View](http://docs.google.com/classsf_1_1View.htm)& sf::RenderTarget::getDefaultView | ( |  | ) | const | | --- | --- | --- | --- | --- | | inherited |
| --- | --- | --- | --- | --- | --- | --- |

Get the default view of the render target.

The default view has the initial size of the render target, and never changes after the target has been created.

ReturnsThe default view of the render target See Also[setView](http://docs.google.com/classsf_1_1RenderTarget.htm#a063db6dd0a14913504af30e50cb6d946), [getView](http://docs.google.com/classsf_1_1RenderTarget.htm#a98f721cc6dc11478922427fedfb2288b)

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

Return the size of the rendering region of the texture.

The returned value is the size that you passed to the create function.

ReturnsSize in pixels

Implements [sf::RenderTarget](http://docs.google.com/classsf_1_1RenderTarget.htm#a2e5ade2457d9fb4c4907ae5b3d9e94a5).

| const [Texture](http://docs.google.com/classsf_1_1Texture.htm)& sf::RenderTexture::getTexture | ( |  | ) | const |
| --- | --- | --- | --- | --- |

Get a read-only reference to the target texture.

After drawing to the render-texture and calling Display, you can retrieve the updated texture using this function, and draw it using a sprite (for example). The internal [sf::Texture](http://docs.google.com/classsf_1_1Texture.htm) of a render-texture is always the same instance, so that it is possible to call this function once and keep a reference to the texture even after it is modified.

ReturnsConst reference to the texture

| | const [View](http://docs.google.com/classsf_1_1View.htm)& sf::RenderTarget::getView | ( |  | ) | const | | --- | --- | --- | --- | --- | | inherited |
| --- | --- | --- | --- | --- | --- | --- |

Get the view currently in use in the render target.

ReturnsThe view object that is currently used See Also[setView](http://docs.google.com/classsf_1_1RenderTarget.htm#a063db6dd0a14913504af30e50cb6d946), [getDefaultView](http://docs.google.com/classsf_1_1RenderTarget.htm#a718b1aa6296bf855171699cc18251ced)

| | [IntRect](http://docs.google.com/classsf_1_1Rect.htm) sf::RenderTarget::getViewport | ( | const [View](http://docs.google.com/classsf_1_1View.htm) & | *view* | ) | const | | --- | --- | --- | --- | --- | --- | | inherited |
| --- | --- | --- | --- | --- | --- | --- | --- |

Get the viewport of a view, applied to this render target.

The viewport is defined in the view as a ratio, this function simply applies this ratio to the current dimensions of the render target to calculate the pixels rectangle that the viewport actually covers in the target.

Parameters

| view | The view for which we want to compute the viewport |
| --- | --- |

ReturnsViewport rectangle, expressed in pixels

| | void sf::RenderTarget::initialize | ( |  | ) |  | | --- | --- | --- | --- | --- | | protectedinherited |
| --- | --- | --- | --- | --- | --- | --- |

Performs the common initialization step after creation.

The derived classes must call this function after the target is created and ready for drawing.

| bool sf::RenderTexture::isRepeated | ( |  | ) | const |
| --- | --- | --- | --- | --- |

Tell whether the texture is repeated or not.

ReturnsTrue if texture is repeated See Also[setRepeated](http://docs.google.com/classsf_1_1RenderTexture.htm#af8f97b33512bf7d5b6be3da6f65f7365)

| bool sf::RenderTexture::isSmooth | ( |  | ) | const |
| --- | --- | --- | --- | --- |

Tell whether the smooth filtering is enabled or not.

ReturnsTrue if texture smoothing is enabled See Also[setSmooth](http://docs.google.com/classsf_1_1RenderTexture.htm#af08991e63c6020865dd07b20e27305b6)

| | [Vector2i](http://docs.google.com/classsf_1_1Vector2.htm) sf::RenderTarget::mapCoordsToPixel | ( | const [Vector2f](http://docs.google.com/classsf_1_1Vector2.htm) & | *point* | ) | const | | --- | --- | --- | --- | --- | --- | | inherited |
| --- | --- | --- | --- | --- | --- | --- | --- |

Convert a point from world coordinates to target coordinates, using the current view.

This function is an overload of the mapCoordsToPixel function that implicitely uses the current view. It is equivalent to:

target.[mapCoordsToPixel](http://docs.google.com/classsf_1_1RenderTarget.htm#aa0c11e1989573f2cce64c621205f8e83)(point, target.[getView](http://docs.google.com/classsf_1_1RenderTarget.htm#a98f721cc6dc11478922427fedfb2288b)());

Parameters

| point | Point to convert |
| --- | --- |

ReturnsThe converted point, in target coordinates (pixels) See Also[mapPixelToCoords](http://docs.google.com/classsf_1_1RenderTarget.htm#a2b0cab0e4c6af29d4efaba149d28116d)

| | [Vector2i](http://docs.google.com/classsf_1_1Vector2.htm) sf::RenderTarget::mapCoordsToPixel | ( | const [Vector2f](http://docs.google.com/classsf_1_1Vector2.htm) & | *point*, | | --- | --- | --- | --- | |  |  | const [View](http://docs.google.com/classsf_1_1View.htm) & | *view* | |  | ) |  | const | | inherited |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

Convert a point from world coordinates to target coordinates.

This function finds the pixel of the render-target that matches the given 2D point. In other words, it goes through the same process as the graphics card, to compute the final position of a rendered point.

Initially, both coordinate systems (world units and target pixels) match perfectly. But if you define a custom view or resize your render-target, this assertion is not true anymore, ie. a point located at (150, 75) in your 2D world may map to the pixel (10, 50) of your render-target – if the view is translated by (140, 25).

This version uses a custom view for calculations, see the other overload of the function if you want to use the current view of the render-target.

Parameters

| point | Point to convert |
| --- | --- |
| view | The view to use for converting the point |

ReturnsThe converted point, in target coordinates (pixels) See Also[mapPixelToCoords](http://docs.google.com/classsf_1_1RenderTarget.htm#a2b0cab0e4c6af29d4efaba149d28116d)

| | [Vector2f](http://docs.google.com/classsf_1_1Vector2.htm) sf::RenderTarget::mapPixelToCoords | ( | const [Vector2i](http://docs.google.com/classsf_1_1Vector2.htm) & | *point* | ) | const | | --- | --- | --- | --- | --- | --- | | inherited |
| --- | --- | --- | --- | --- | --- | --- | --- |

Convert a point from target coordinates to world coordinates, using the current view.

This function is an overload of the mapPixelToCoords function that implicitely uses the current view. It is equivalent to:

target.[mapPixelToCoords](http://docs.google.com/classsf_1_1RenderTarget.htm#a2b0cab0e4c6af29d4efaba149d28116d)(point, target.[getView](http://docs.google.com/classsf_1_1RenderTarget.htm#a98f721cc6dc11478922427fedfb2288b)());

Parameters

| point | Pixel to convert |
| --- | --- |

ReturnsThe converted point, in "world" coordinates See Also[mapCoordsToPixel](http://docs.google.com/classsf_1_1RenderTarget.htm#aa0c11e1989573f2cce64c621205f8e83)

| | [Vector2f](http://docs.google.com/classsf_1_1Vector2.htm) sf::RenderTarget::mapPixelToCoords | ( | const [Vector2i](http://docs.google.com/classsf_1_1Vector2.htm) & | *point*, | | --- | --- | --- | --- | |  |  | const [View](http://docs.google.com/classsf_1_1View.htm) & | *view* | |  | ) |  | const | | inherited |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

Convert a point from target coordinates to world coordinates.

This function finds the 2D position that matches the given pixel of the render-target. In other words, it does the inverse of what the graphics card does, to find the initial position of a rendered pixel.

Initially, both coordinate systems (world units and target pixels) match perfectly. But if you define a custom view or resize your render-target, this assertion is not true anymore, ie. a point located at (10, 50) in your render-target may map to the point (150, 75) in your 2D world – if the view is translated by (140, 25).

For render-windows, this function is typically used to find which point (or object) is located below the mouse cursor.

This version uses a custom view for calculations, see the other overload of the function if you want to use the current view of the render-target.

Parameters

| point | Pixel to convert |
| --- | --- |
| view | The view to use for converting the point |

ReturnsThe converted point, in "world" units See Also[mapCoordsToPixel](http://docs.google.com/classsf_1_1RenderTarget.htm#aa0c11e1989573f2cce64c621205f8e83)

| | void sf::RenderTarget::popGLStates | ( |  | ) |  | | --- | --- | --- | --- | --- | | inherited |
| --- | --- | --- | --- | --- | --- | --- |

Restore the previously saved OpenGL render states and matrices.

See the description of pushGLStates to get a detailed description of these functions.

See Also[pushGLStates](http://docs.google.com/classsf_1_1RenderTarget.htm#a8d1998464ccc54e789aaf990242b47f7)

| | void sf::RenderTarget::pushGLStates | ( |  | ) |  | | --- | --- | --- | --- | --- | | inherited |
| --- | --- | --- | --- | --- | --- | --- |

Save the current OpenGL render states and matrices.

This function can be used when you mix SFML drawing and direct OpenGL rendering. Combined with PopGLStates, it ensures that:

* SFML's internal states are not messed up by your OpenGL code
* your OpenGL states are not modified by a call to a SFML function

More specifically, it must be used around code that calls Draw functions. Example:

// OpenGL code here...

window.pushGLStates();

window.draw(...);

window.draw(...);

window.popGLStates();

// OpenGL code here...

Note that this function is quite expensive: it saves all the possible OpenGL states and matrices, even the ones you don't care about. Therefore it should be used wisely. It is provided for convenience, but the best results will be achieved if you handle OpenGL states yourself (because you know which states have really changed, and need to be saved and restored). Take a look at the ResetGLStates function if you do so.

See Also[popGLStates](http://docs.google.com/classsf_1_1RenderTarget.htm#ad5a98401113df931ddcd54c080f7aa8e)

| | void sf::RenderTarget::resetGLStates | ( |  | ) |  | | --- | --- | --- | --- | --- | | inherited |
| --- | --- | --- | --- | --- | --- | --- |

Reset the internal OpenGL states so that the target is ready for drawing.

This function can be used when you mix SFML drawing and direct OpenGL rendering, if you choose not to use pushGLStates/popGLStates. It makes sure that all OpenGL states needed by SFML are set, so that subsequent [draw()](http://docs.google.com/classsf_1_1RenderTarget.htm#a12417a3bcc245c41d957b29583556f39) calls will work as expected.

Example:

// OpenGL code here...

glPushAttrib(...);

window.resetGLStates();

window.draw(...);

window.draw(...);

glPopAttrib(...);

// OpenGL code here...

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

Activate of deactivate the render-texture for rendering.

This function makes the render-texture's context current for future OpenGL rendering operations (so you shouldn't care about it if you're not doing direct OpenGL stuff). Only one context can be current in a thread, so if you want to draw OpenGL geometry to another render target (like a [RenderWindow](http://docs.google.com/classsf_1_1RenderWindow.htm)) don't forget to activate it again.

Parameters

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

ReturnsTrue if operation was successful, false otherwise

| void sf::RenderTexture::setRepeated | ( | bool | *repeated* | ) |  |
| --- | --- | --- | --- | --- | --- |

Enable or disable texture repeating.

This function is similar to [Texture::setRepeated](http://docs.google.com/classsf_1_1Texture.htm#aaa87d1eff053b9d4d34a24c784a28658). This parameter is disabled by default.

Parameters

| repeated | True to enable repeating, false to disable it |
| --- | --- |

See Also[isRepeated](http://docs.google.com/classsf_1_1RenderTexture.htm#ae480a2ec7ee166afa50232e634d2668c)

| void sf::RenderTexture::setSmooth | ( | bool | *smooth* | ) |  |
| --- | --- | --- | --- | --- | --- |

Enable or disable texture smoothing.

This function is similar to [Texture::setSmooth](http://docs.google.com/classsf_1_1Texture.htm#a0c3bd6825b9a99714f10d44179d74324). This parameter is disabled by default.

Parameters

| smooth | True to enable smoothing, false to disable it |
| --- | --- |

See Also[isSmooth](http://docs.google.com/classsf_1_1RenderTexture.htm#ae385f4f4dbd2af50fb11947bf0bcb83d)

| | void sf::RenderTarget::setView | ( | const [View](http://docs.google.com/classsf_1_1View.htm) & | *view* | ) |  | | --- | --- | --- | --- | --- | --- | | inherited |
| --- | --- | --- | --- | --- | --- | --- | --- |

Change the current active view.

The view is like a 2D camera, it controls which part of the 2D scene is visible, and how it is viewed in the render-target. The new view will affect everything that is drawn, until another view is set. The render target keeps its own copy of the view object, so it is not necessary to keep the original one alive after calling this function. To restore the original view of the target, you can pass the result of [getDefaultView()](http://docs.google.com/classsf_1_1RenderTarget.htm#a718b1aa6296bf855171699cc18251ced) to this function.

Parameters

| view | New view to use |
| --- | --- |

See Also[getView](http://docs.google.com/classsf_1_1RenderTarget.htm#a98f721cc6dc11478922427fedfb2288b), [getDefaultView](http://docs.google.com/classsf_1_1RenderTarget.htm#a718b1aa6296bf855171699cc18251ced)

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

* [RenderTexture.hpp](http://docs.google.com/RenderTexture_8hpp_source.htm)

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