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**
* [RenderTarget](http://docs.google.com/classsf_1_1RenderTarget.htm)

[Classes](#3znysh7) | [Public Member Functions](#_gjdgxs) | [Protected Member Functions](#_30j0zll) | [List of all members](http://docs.google.com/classsf_1_1RenderTarget-members.htm)

sf::RenderTarget Class Referenceabstract

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

Base class for all render targets (window, texture, ...) [More...](http://docs.google.com/classsf_1_1RenderTarget.htm#details)

#include <[RenderTarget.hpp](http://docs.google.com/RenderTarget_8hpp_source.htm)>

Inheritance diagram for sf::RenderTarget:



| Public Member Functions | |
| --- | --- |
| virtual | [~RenderTarget](http://docs.google.com/classsf_1_1RenderTarget.htm#a9abd1654a99fba46f6887b9c625b9b06) () |
|  | Destructor. |
|  | |
| 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. |
|  | |
| virtual [Vector2u](http://docs.google.com/classsf_1_1Vector2.htm) | [getSize](http://docs.google.com/classsf_1_1RenderTarget.htm#a2e5ade2457d9fb4c4907ae5b3d9e94a5) () const =0 |
|  | Return the size of the rendering region of the target. |
|  | |
| 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 | |
| --- | --- |
|  | [RenderTarget](http://docs.google.com/classsf_1_1RenderTarget.htm#a2997c96cbd93cb8ce0aba2ddae35b86f) () |
|  | Default constructor. |
|  | |
| void | [initialize](http://docs.google.com/classsf_1_1RenderTarget.htm#af530274b34159d644e509b4b4dc43eb7) () |
|  | Performs the common initialization step after creation. |
|  | |

## Detailed Description

Base class for all render targets (window, texture, ...)

[sf::RenderTarget](http://docs.google.com/classsf_1_1RenderTarget.htm) defines the common behaviour of all the 2D render targets usable in the graphics module.

It makes it possible to draw 2D entities like sprites, shapes, text without using any OpenGL command directly.

A [sf::RenderTarget](http://docs.google.com/classsf_1_1RenderTarget.htm) is also able to use views ([sf::View](http://docs.google.com/classsf_1_1View.htm)), which are a kind of 2D cameras. With views you can globally scroll, rotate or zoom everything that is drawn, without having to transform every single entity. See the documentation of [sf::View](http://docs.google.com/classsf_1_1View.htm) for more details and sample pieces of code about this class.

On top of that, render targets are still able to render direct OpenGL stuff. It is even possible to mix together OpenGL calls and regular SFML drawing commands. When doing so, make sure that OpenGL states are not messed up by calling the pushGLStates/popGLStates functions.

See Also[sf::RenderWindow](http://docs.google.com/classsf_1_1RenderWindow.htm), [sf::RenderTexture](http://docs.google.com/classsf_1_1RenderTexture.htm), [sf::View](http://docs.google.com/classsf_1_1View.htm)

Definition at line [51](http://docs.google.com/RenderTarget_8hpp_source.htm#l00051) of file [RenderTarget.hpp](http://docs.google.com/RenderTarget_8hpp_source.htm).

## Constructor & Destructor Documentation

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

Destructor.

| | sf::RenderTarget::RenderTarget | ( |  | ) |  | | --- | --- | --- | --- | --- | | protected |
| --- | --- | --- | --- | --- | --- | --- |

Default constructor.

## 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) | ) |  |
| --- | --- | --- | --- | --- | --- |

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 |
| --- | --- |

| 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) |
|  | ) |  |  |

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) |
|  | ) |  |  |

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 |
| --- | --- | --- | --- | --- |

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::RenderTarget::getSize | ( |  | ) | const | | --- | --- | --- | --- | --- | | pure virtual |
| --- | --- | --- | --- | --- | --- | --- |

Return the size of the rendering region of the target.

ReturnsSize in pixels

Implemented in [sf::RenderTexture](http://docs.google.com/classsf_1_1RenderTexture.htm#a757ba45ec7a7deefcaef717049b00b8c), and [sf::RenderWindow](http://docs.google.com/classsf_1_1RenderWindow.htm#a2c7ff414be32621a453745cf2a0f8a3e).

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

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 |
| --- | --- | --- | --- | --- | --- |

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 | ( |  | ) |  | | --- | --- | --- | --- | --- | | protected |
| --- | --- | --- | --- | --- | --- | --- |

Performs the common initialization step after creation.

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

| [Vector2i](http://docs.google.com/classsf_1_1Vector2.htm) sf::RenderTarget::mapCoordsToPixel | ( | 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.

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 |

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 |
| --- | --- | --- | --- | --- | --- |

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 |

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 | ( |  | ) |  |
| --- | --- | --- | --- | --- |

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 | ( |  | ) |  |
| --- | --- | --- | --- | --- |

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 | ( |  | ) |  |
| --- | --- | --- | --- | --- |

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...

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

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:

* [RenderTarget.hpp](http://docs.google.com/RenderTarget_8hpp_source.htm)

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