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**
* [Rect](http://docs.google.com/classsf_1_1Rect.htm)

[Public Member Functions](#_gjdgxs) | [Public Attributes](#_30j0zll) | [Related Functions](#_1fob9te) | [List of all members](http://docs.google.com/classsf_1_1Rect-members.htm)

sf::Rect< T > Class Template Reference

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

Utility class for manipulating 2D axis aligned rectangles. [More...](http://docs.google.com/classsf_1_1Rect.htm#details)

#include <[Rect.hpp](http://docs.google.com/Rect_8hpp_source.htm)>

| Public Member Functions | |
| --- | --- |
|  | [Rect](http://docs.google.com/classsf_1_1Rect.htm#a0f87ebaef9722a6222fd2e04ce8efb37) () |
|  | Default constructor. |
|  | |
|  | [Rect](http://docs.google.com/classsf_1_1Rect.htm#a15cdbc5a1aed3a8fc7be1bd5004f19f9) (T rectLeft, T rectTop, T rectWidth, T rectHeight) |
|  | Construct the rectangle from its coordinates. |
|  | |
|  | [Rect](http://docs.google.com/classsf_1_1Rect.htm#a27fdf85caa6d12caeeff78913cc59936) (const [Vector2](http://docs.google.com/classsf_1_1Vector2.htm)< T > &position, const [Vector2](http://docs.google.com/classsf_1_1Vector2.htm)< T > &size) |
|  | Construct the rectangle from position and size. |
|  | |
| template<typename U > | |
|  | [Rect](http://docs.google.com/classsf_1_1Rect.htm#a6fff2bb7e93677839461a66bc2957de0) (const [Rect](http://docs.google.com/classsf_1_1Rect.htm)< U > &rectangle) |
|  | Construct the rectangle from another type of rectangle. |
|  | |
| bool | [contains](http://docs.google.com/classsf_1_1Rect.htm#aa8a5364c84de6dd5299f833b54e31ef1) (T x, T y) const |
|  | Check if a point is inside the rectangle's area. |
|  | |
| bool | [contains](http://docs.google.com/classsf_1_1Rect.htm#a24163acdb9b2987c0ea55c201e270d41) (const [Vector2](http://docs.google.com/classsf_1_1Vector2.htm)< T > &point) const |
|  | Check if a point is inside the rectangle's area. |
|  | |
| bool | [intersects](http://docs.google.com/classsf_1_1Rect.htm#a566740c8f58e01bb052266f47e7e1011) (const [Rect](http://docs.google.com/classsf_1_1Rect.htm)< T > &rectangle) const |
|  | Check the intersection between two rectangles. |
|  | |
| bool | [intersects](http://docs.google.com/classsf_1_1Rect.htm#a5f1874792b04c7e221bb786b31f5836e) (const [Rect](http://docs.google.com/classsf_1_1Rect.htm)< T > &rectangle, [Rect](http://docs.google.com/classsf_1_1Rect.htm)< T > &intersection) const |
|  | Check the intersection between two rectangles. |
|  | |
| template<typename T > | |
|  | **Rect** (T rectLeft, T rectTop, T rectWidth, T rectHeight) |
|  | |
| template<typename T > | |
|  | **Rect** (const [Vector2](http://docs.google.com/classsf_1_1Vector2.htm)< T > &position, const [Vector2](http://docs.google.com/classsf_1_1Vector2.htm)< T > &size) |
|  | |
| template<typename U > | |
|  | **Rect** (const [Rect](http://docs.google.com/classsf_1_1Rect.htm)< U > &rectangle) |
|  | |

| Public Attributes | |
| --- | --- |
| T | [left](http://docs.google.com/classsf_1_1Rect.htm#aa49960fa465103d9cb7069ceb25c7c32) |
|  | Left coordinate of the rectangle. |
|  | |
| T | [top](http://docs.google.com/classsf_1_1Rect.htm#abd3d3a2d0ad211ef0082bd0aa1a5c0e3) |
|  | Top coordinate of the rectangle. |
|  | |
| T | [width](http://docs.google.com/classsf_1_1Rect.htm#a4dd5b9d4333bebbc51bd309298fd500f) |
|  | Width of the rectangle. |
|  | |
| T | [height](http://docs.google.com/classsf_1_1Rect.htm#a6fa0fc7de1636d78cae1a1b54eef95cd) |
|  | Height of the rectangle. |
|  | |

| Related Functions | |
| --- | --- |
| (Note that these are not member functions.) | |
| template<typename T > | |
| bool | [operator==](http://docs.google.com/classsf_1_1Rect.htm#ab3488b5dbd0e587c4d7cb80605affc46) (const [Rect](http://docs.google.com/classsf_1_1Rect.htm)< T > &[left](http://docs.google.com/classsf_1_1Rect.htm#aa49960fa465103d9cb7069ceb25c7c32), const [Rect](http://docs.google.com/classsf_1_1Rect.htm)< T > &right) |
|  | Overload of binary operator ==. |
|  | |
| template<typename T > | |
| bool | [operator!=](http://docs.google.com/classsf_1_1Rect.htm#a03fc4c105687b7d0f07b6b4ed4b45581) (const [Rect](http://docs.google.com/classsf_1_1Rect.htm)< T > &[left](http://docs.google.com/classsf_1_1Rect.htm#aa49960fa465103d9cb7069ceb25c7c32), const [Rect](http://docs.google.com/classsf_1_1Rect.htm)< T > &right) |
|  | Overload of binary operator !=. |
|  | |

## Detailed Description

template<typename T>

class sf::Rect< T >

Utility class for manipulating 2D axis aligned rectangles.

A rectangle is defined by its top-left corner and its size.

It is a very simple class defined for convenience, so its member variables (left, top, width and height) are public and can be accessed directly, just like the vector classes ([Vector2](http://docs.google.com/classsf_1_1Vector2.htm) and [Vector3](http://docs.google.com/classsf_1_1Vector3.htm)).

To keep things simple, [sf::Rect](http://docs.google.com/classsf_1_1Rect.htm) doesn't define functions to emulate the properties that are not directly members (such as right, bottom, center, etc.), it rather only provides intersection functions.

[sf::Rect](http://docs.google.com/classsf_1_1Rect.htm) uses the usual rules for its boundaries:

* The left and top edges are included in the rectangle's area
* The right (left + width) and bottom (top + height) edges are excluded from the rectangle's area

This means that sf::IntRect(0, 0, 1, 1) and sf::IntRect(1, 1, 1, 1) don't intersect.

[sf::Rect](http://docs.google.com/classsf_1_1Rect.htm) is a template and may be used with any numeric type, but for simplicity the instanciations used by SFML are typedefed:

* sf::Rect<int> is sf::IntRect
* sf::Rect<float> is sf::FloatRect

So that you don't have to care about the template syntax.

Usage example:

// Define a rectangle, located at (0, 0) with a size of 20x5

[sf::IntRect](http://docs.google.com/classsf_1_1Rect.htm) r1(0, 0, 20, 5);

// Define another rectangle, located at (4, 2) with a size of 18x10

[sf::Vector2i](http://docs.google.com/classsf_1_1Vector2.htm) position(4, 2);

[sf::Vector2i](http://docs.google.com/classsf_1_1Vector2.htm) size(18, 10);

[sf::IntRect](http://docs.google.com/classsf_1_1Rect.htm) r2(position, size);

// Test intersections with the point (3, 1)

bool b1 = r1.contains(3, 1); // true

bool b2 = r2.contains(3, 1); // false

// Test the intersection between r1 and r2

[sf::IntRect](http://docs.google.com/classsf_1_1Rect.htm) result;

bool b3 = r1.[intersects](http://docs.google.com/classsf_1_1Rect.htm#a566740c8f58e01bb052266f47e7e1011)(r2, result); // true

// result == (4, 2, 16, 3)

Definition at line [42](http://docs.google.com/Rect_8hpp_source.htm#l00042) of file [Rect.hpp](http://docs.google.com/Rect_8hpp_source.htm).

## Constructor & Destructor Documentation

template<typename T>

| [sf::Rect](http://docs.google.com/classsf_1_1Rect.htm)< T >::[Rect](http://docs.google.com/classsf_1_1Rect.htm) | ( |  | ) |  |
| --- | --- | --- | --- | --- |

Default constructor.

Creates an empty rectangle (it is equivalent to calling Rect(0, 0, 0, 0)).

template<typename T>

| [sf::Rect](http://docs.google.com/classsf_1_1Rect.htm)< T >::[Rect](http://docs.google.com/classsf_1_1Rect.htm) | ( | T | *rectLeft*, |
| --- | --- | --- | --- |
|  |  | T | *rectTop*, |
|  |  | T | *rectWidth*, |
|  |  | T | *rectHeight* |
|  | ) |  |  |

Construct the rectangle from its coordinates.

Be careful, the last two parameters are the width and height, not the right and bottom coordinates!

Parameters

| rectLeft | Left coordinate of the rectangle |
| --- | --- |
| rectTop | Top coordinate of the rectangle |
| rectWidth | Width of the rectangle |
| rectHeight | Height of the rectangle |

template<typename T>

| [sf::Rect](http://docs.google.com/classsf_1_1Rect.htm)< T >::[Rect](http://docs.google.com/classsf_1_1Rect.htm) | ( | const [Vector2](http://docs.google.com/classsf_1_1Vector2.htm)< T > & | *position*, |
| --- | --- | --- | --- |
|  |  | const [Vector2](http://docs.google.com/classsf_1_1Vector2.htm)< T > & | *size* |
|  | ) |  |  |

Construct the rectangle from position and size.

Be careful, the last parameter is the size, not the bottom-right corner!

Parameters

| position | Position of the top-left corner of the rectangle |
| --- | --- |
| size | Size of the rectangle |

template<typename T>

template<typename U >

| | [sf::Rect](http://docs.google.com/classsf_1_1Rect.htm)< T >::[Rect](http://docs.google.com/classsf_1_1Rect.htm) | ( | const [Rect](http://docs.google.com/classsf_1_1Rect.htm)< U > & | *rectangle* | ) |  | | --- | --- | --- | --- | --- | --- | | explicit |
| --- | --- | --- | --- | --- | --- | --- | --- |

Construct the rectangle from another type of rectangle.

This constructor doesn't replace the copy constructor, it's called only when U != T. A call to this constructor will fail to compile if U is not convertible to T.

Parameters

| rectangle | Rectangle to convert |
| --- | --- |

## Member Function Documentation

template<typename T>

| bool [sf::Rect](http://docs.google.com/classsf_1_1Rect.htm)< 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 |

ReturnsTrue if the point is inside, false otherwise See Also[intersects](http://docs.google.com/classsf_1_1Rect.htm#a566740c8f58e01bb052266f47e7e1011)

template<typename T>

| bool [sf::Rect](http://docs.google.com/classsf_1_1Rect.htm)< T >::contains | ( | const [Vector2](http://docs.google.com/classsf_1_1Vector2.htm)< T > & | *point* | ) | const |
| --- | --- | --- | --- | --- | --- |

Check if a point is inside the rectangle's area.

Parameters

| point | Point to test |
| --- | --- |

ReturnsTrue if the point is inside, false otherwise See Also[intersects](http://docs.google.com/classsf_1_1Rect.htm#a566740c8f58e01bb052266f47e7e1011)

template<typename T>

| bool [sf::Rect](http://docs.google.com/classsf_1_1Rect.htm)< T >::intersects | ( | const [Rect](http://docs.google.com/classsf_1_1Rect.htm)< T > & | *rectangle* | ) | const |
| --- | --- | --- | --- | --- | --- |

Check the intersection between two rectangles.

Parameters

| rectangle | Rectangle to test |
| --- | --- |

ReturnsTrue if rectangles overlap, false otherwise See Also[contains](http://docs.google.com/classsf_1_1Rect.htm#aa8a5364c84de6dd5299f833b54e31ef1)

template<typename T>

| bool [sf::Rect](http://docs.google.com/classsf_1_1Rect.htm)< T >::intersects | ( | const [Rect](http://docs.google.com/classsf_1_1Rect.htm)< T > & | *rectangle*, |
| --- | --- | --- | --- |
|  |  | [Rect](http://docs.google.com/classsf_1_1Rect.htm)< T > & | *intersection* |
|  | ) |  | const |

Check the intersection between two rectangles.

This overload returns the overlapped rectangle in the *intersection* parameter.

Parameters

| rectangle | Rectangle to test |
| --- | --- |
| intersection | Rectangle to be filled with the intersection |

ReturnsTrue if rectangles overlap, false otherwise See Also[contains](http://docs.google.com/classsf_1_1Rect.htm#aa8a5364c84de6dd5299f833b54e31ef1)

## Friends And Related Function Documentation

template<typename T >

| | bool operator!= | ( | const [Rect](http://docs.google.com/classsf_1_1Rect.htm)< T > & | *left*, | | --- | --- | --- | --- | |  |  | const [Rect](http://docs.google.com/classsf_1_1Rect.htm)< T > & | *right* | |  | ) |  |  | | related |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

Overload of binary operator !=.

This operator compares strict difference between two rectangles.

Parameters

| left | Left operand (a rectangle) |
| --- | --- |
| right | Right operand (a rectangle) |

ReturnsTrue if *left* is not equal to *right*

template<typename T >

| | bool operator== | ( | const [Rect](http://docs.google.com/classsf_1_1Rect.htm)< T > & | *left*, | | --- | --- | --- | --- | |  |  | const [Rect](http://docs.google.com/classsf_1_1Rect.htm)< T > & | *right* | |  | ) |  |  | | related |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

Overload of binary operator ==.

This operator compares strict equality between two rectangles.

Parameters

| left | Left operand (a rectangle) |
| --- | --- |
| right | Right operand (a rectangle) |

ReturnsTrue if *left* is equal to *right*

## Member Data Documentation

template<typename T>

| T [sf::Rect](http://docs.google.com/classsf_1_1Rect.htm)< T >::height |
| --- |

Height of the rectangle.

Definition at line [154](http://docs.google.com/Rect_8hpp_source.htm#l00154) of file [Rect.hpp](http://docs.google.com/Rect_8hpp_source.htm).

template<typename T>

| T [sf::Rect](http://docs.google.com/classsf_1_1Rect.htm)< T >::left |
| --- |

Left coordinate of the rectangle.

Definition at line [151](http://docs.google.com/Rect_8hpp_source.htm#l00151) of file [Rect.hpp](http://docs.google.com/Rect_8hpp_source.htm).

template<typename T>

| T [sf::Rect](http://docs.google.com/classsf_1_1Rect.htm)< T >::top |
| --- |

Top coordinate of the rectangle.

Definition at line [152](http://docs.google.com/Rect_8hpp_source.htm#l00152) of file [Rect.hpp](http://docs.google.com/Rect_8hpp_source.htm).

template<typename T>

| T [sf::Rect](http://docs.google.com/classsf_1_1Rect.htm)< T >::width |
| --- |

Width of the rectangle.

Definition at line [153](http://docs.google.com/Rect_8hpp_source.htm#l00153) of file [Rect.hpp](http://docs.google.com/Rect_8hpp_source.htm).

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

* [Rect.hpp](http://docs.google.com/Rect_8hpp_source.htm)
* [Rect.inl](http://docs.google.com/Rect_8inl_source.htm)

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