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**
* [Sprite](http://docs.google.com/classsf_1_1Sprite.htm)

[Public Member Functions](#_gjdgxs) | [List of all members](http://docs.google.com/classsf_1_1Sprite-members.htm)

sf::Sprite Class Reference

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

[Drawable](http://docs.google.com/classsf_1_1Drawable.htm) representation of a texture, with its own transformations, color, etc. [More...](http://docs.google.com/classsf_1_1Sprite.htm#details)

#include <[Sprite.hpp](http://docs.google.com/Sprite_8hpp_source.htm)>

Inheritance diagram for sf::Sprite:



| Public Member Functions | |
| --- | --- |
|  | [Sprite](http://docs.google.com/classsf_1_1Sprite.htm#a92559fbca895a96758abf5eabab96984) () |
|  | Default constructor. |
|  | |
|  | [Sprite](http://docs.google.com/classsf_1_1Sprite.htm#a2a9fca374d7abf084bb1c143a879ff4a) (const [Texture](http://docs.google.com/classsf_1_1Texture.htm) &texture) |
|  | Construct the sprite from a source texture. |
|  | |
|  | [Sprite](http://docs.google.com/classsf_1_1Sprite.htm#a01cfe1402372d243dbaa2ffa96020206) (const [Texture](http://docs.google.com/classsf_1_1Texture.htm) &texture, const [IntRect](http://docs.google.com/classsf_1_1Rect.htm) &rectangle) |
|  | Construct the sprite from a sub-rectangle of a source texture. |
|  | |
| void | [setTexture](http://docs.google.com/classsf_1_1Sprite.htm#a3729c88d88ac38c19317c18e87242560) (const [Texture](http://docs.google.com/classsf_1_1Texture.htm) &texture, bool resetRect=false) |
|  | Change the source texture of the sprite. |
|  | |
| void | [setTextureRect](http://docs.google.com/classsf_1_1Sprite.htm#a3fefec419a4e6a90c0fd54c793d82ec2) (const [IntRect](http://docs.google.com/classsf_1_1Rect.htm) &rectangle) |
|  | Set the sub-rectangle of the texture that the sprite will display. |
|  | |
| void | [setColor](http://docs.google.com/classsf_1_1Sprite.htm#a14def44da6437bfea20c4df5e71aba4c) (const [Color](http://docs.google.com/classsf_1_1Color.htm) &color) |
|  | Set the global color of the sprite. |
|  | |
| const [Texture](http://docs.google.com/classsf_1_1Texture.htm) \* | [getTexture](http://docs.google.com/classsf_1_1Sprite.htm#a2c4fbb983b29e14f9bb69a3a75feae46) () const |
|  | Get the source texture of the sprite. |
|  | |
| const [IntRect](http://docs.google.com/classsf_1_1Rect.htm) & | [getTextureRect](http://docs.google.com/classsf_1_1Sprite.htm#a89aa58bc39e2072c6afe7c547d27b5a0) () const |
|  | Get the sub-rectangle of the texture displayed by the sprite. |
|  | |
| const [Color](http://docs.google.com/classsf_1_1Color.htm) & | [getColor](http://docs.google.com/classsf_1_1Sprite.htm#ab05fafd4e9999608a5cae4985e7b52df) () const |
|  | Get the global color of the sprite. |
|  | |
| [FloatRect](http://docs.google.com/classsf_1_1Rect.htm) | [getLocalBounds](http://docs.google.com/classsf_1_1Sprite.htm#a69557a8369bc2e26dd2e2eb2c50f5c90) () const |
|  | Get the local bounding rectangle of the entity. |
|  | |
| [FloatRect](http://docs.google.com/classsf_1_1Rect.htm) | [getGlobalBounds](http://docs.google.com/classsf_1_1Sprite.htm#a203d2d8087bfdca2ebc3c0485cdb7409) () const |
|  | Get the global bounding rectangle of the entity. |
|  | |
| void | [setPosition](http://docs.google.com/classsf_1_1Transformable.htm#a4dbfb1a7c80688b0b4c477d706550208) (float x, float y) |
|  | set the position of the object |
|  | |
| void | [setPosition](http://docs.google.com/classsf_1_1Transformable.htm#af1a42209ce2b5d3f07b00f917bcd8015) (const [Vector2f](http://docs.google.com/classsf_1_1Vector2.htm) &position) |
|  | set the position of the object |
|  | |
| void | [setRotation](http://docs.google.com/classsf_1_1Transformable.htm#a32baf2bf1a74699b03bf8c95030a38ed) (float angle) |
|  | set the orientation of the object |
|  | |
| void | [setScale](http://docs.google.com/classsf_1_1Transformable.htm#aaec50b46b3f41b054763304d1e727471) (float factorX, float factorY) |
|  | set the scale factors of the object |
|  | |
| void | [setScale](http://docs.google.com/classsf_1_1Transformable.htm#a4c48a87f1626047e448f9c1a68ff167e) (const [Vector2f](http://docs.google.com/classsf_1_1Vector2.htm) &factors) |
|  | set the scale factors of the object |
|  | |
| void | [setOrigin](http://docs.google.com/classsf_1_1Transformable.htm#a56c67bd80aae8418d13fb96c034d25ec) (float x, float y) |
|  | set the local origin of the object |
|  | |
| void | [setOrigin](http://docs.google.com/classsf_1_1Transformable.htm#aa93a835ffbf3bee2098dfbbc695a7f05) (const [Vector2f](http://docs.google.com/classsf_1_1Vector2.htm) &origin) |
|  | set the local origin of the object |
|  | |
| const [Vector2f](http://docs.google.com/classsf_1_1Vector2.htm) & | [getPosition](http://docs.google.com/classsf_1_1Transformable.htm#a6a0552d8cf155b7df25f6ceda8ee45a5) () const |
|  | get the position of the object |
|  | |
| float | [getRotation](http://docs.google.com/classsf_1_1Transformable.htm#ad783a7e9971398ec613d22455252809e) () const |
|  | get the orientation of the object |
|  | |
| const [Vector2f](http://docs.google.com/classsf_1_1Vector2.htm) & | [getScale](http://docs.google.com/classsf_1_1Transformable.htm#a3ea9639abd7a430ac99afb0aaf1ea562) () const |
|  | get the current scale of the object |
|  | |
| const [Vector2f](http://docs.google.com/classsf_1_1Vector2.htm) & | [getOrigin](http://docs.google.com/classsf_1_1Transformable.htm#a6bddc485d22bb64449d9d2d3a99a778f) () const |
|  | get the local origin of the object |
|  | |
| void | [move](http://docs.google.com/classsf_1_1Transformable.htm#a86b461d6a941ad390c2ad8b6a4a20391) (float offsetX, float offsetY) |
|  | Move the object by a given offset. |
|  | |
| void | [move](http://docs.google.com/classsf_1_1Transformable.htm#ab9ca691522f6ddc1a40406849b87c469) (const [Vector2f](http://docs.google.com/classsf_1_1Vector2.htm) &offset) |
|  | Move the object by a given offset. |
|  | |
| void | [rotate](http://docs.google.com/classsf_1_1Transformable.htm#af8a5ffddc0d93f238fee3bf8efe1ebda) (float angle) |
|  | Rotate the object. |
|  | |
| void | [scale](http://docs.google.com/classsf_1_1Transformable.htm#a3de0c6d8957f3cf318092f3f60656391) (float factorX, float factorY) |
|  | Scale the object. |
|  | |
| void | [scale](http://docs.google.com/classsf_1_1Transformable.htm#adecaa6c69b1f27dd5194b067d96bb694) (const [Vector2f](http://docs.google.com/classsf_1_1Vector2.htm) &factor) |
|  | Scale the object. |
|  | |
| const [Transform](http://docs.google.com/classsf_1_1Transform.htm) & | [getTransform](http://docs.google.com/classsf_1_1Transformable.htm#a3b48c3362e3e2c14fef7551252deb7bb) () const |
|  | get the combined transform of the object |
|  | |
| const [Transform](http://docs.google.com/classsf_1_1Transform.htm) & | [getInverseTransform](http://docs.google.com/classsf_1_1Transformable.htm#ab00de62b5d1efb2ee4cf2566dea98175) () const |
|  | get the inverse of the combined transform of the object |
|  | |

## Detailed Description

[Drawable](http://docs.google.com/classsf_1_1Drawable.htm) representation of a texture, with its own transformations, color, etc.

[sf::Sprite](http://docs.google.com/classsf_1_1Sprite.htm) is a drawable class that allows to easily display a texture (or a part of it) on a render target.

It inherits all the functions from [sf::Transformable](http://docs.google.com/classsf_1_1Transformable.htm): position, rotation, scale, origin. It also adds sprite-specific properties such as the texture to use, the part of it to display, and some convenience functions to change the overall color of the sprite, or to get its bounding rectangle.

[sf::Sprite](http://docs.google.com/classsf_1_1Sprite.htm) works in combination with the [sf::Texture](http://docs.google.com/classsf_1_1Texture.htm) class, which loads and provides the pixel data of a given texture.

The separation of [sf::Sprite](http://docs.google.com/classsf_1_1Sprite.htm) and [sf::Texture](http://docs.google.com/classsf_1_1Texture.htm) allows more flexibility and better performances: indeed a [sf::Texture](http://docs.google.com/classsf_1_1Texture.htm) is a heavy resource, and any operation on it is slow (often too slow for real-time applications). On the other side, a [sf::Sprite](http://docs.google.com/classsf_1_1Sprite.htm) is a lightweight object which can use the pixel data of a [sf::Texture](http://docs.google.com/classsf_1_1Texture.htm) and draw it with its own transformation/color/blending attributes.

It is important to note that the [sf::Sprite](http://docs.google.com/classsf_1_1Sprite.htm) instance doesn't copy the texture that it uses, it only keeps a reference to it. Thus, a [sf::Texture](http://docs.google.com/classsf_1_1Texture.htm) must not be destroyed while it is used by a [sf::Sprite](http://docs.google.com/classsf_1_1Sprite.htm) (i.e. never write a function that uses a local [sf::Texture](http://docs.google.com/classsf_1_1Texture.htm) instance for creating a sprite).

Usage example:

// Declare and load a texture

[sf::Texture](http://docs.google.com/classsf_1_1Texture.htm) texture;

texture.[loadFromFile](http://docs.google.com/classsf_1_1Texture.htm#a8e1b56eabfe33e2e0e1cb03712c7fcc7)("texture.png");

// Create a sprite

[sf::Sprite](http://docs.google.com/classsf_1_1Sprite.htm) sprite;

sprite.[setTexture](http://docs.google.com/classsf_1_1Sprite.htm#a3729c88d88ac38c19317c18e87242560)(texture);

sprite.[setTextureRect](http://docs.google.com/classsf_1_1Sprite.htm#a3fefec419a4e6a90c0fd54c793d82ec2)([sf::IntRect](http://docs.google.com/classsf_1_1Rect.htm)(10, 10, 50, 30));

sprite.[setColor](http://docs.google.com/classsf_1_1Sprite.htm#a14def44da6437bfea20c4df5e71aba4c)([sf::Color](http://docs.google.com/classsf_1_1Color.htm)(255, 255, 255, 200));

sprite.[setPosition](http://docs.google.com/classsf_1_1Transformable.htm#a4dbfb1a7c80688b0b4c477d706550208)(100, 25);

// Draw it

window.draw(sprite);

See Also[sf::Texture](http://docs.google.com/classsf_1_1Texture.htm), [sf::Transformable](http://docs.google.com/classsf_1_1Transformable.htm)

Definition at line [47](http://docs.google.com/Sprite_8hpp_source.htm#l00047) of file [Sprite.hpp](http://docs.google.com/Sprite_8hpp_source.htm).

## Constructor & Destructor Documentation

| sf::Sprite::Sprite | ( |  | ) |  |
| --- | --- | --- | --- | --- |

Default constructor.

Creates an empty sprite with no source texture.

| | sf::Sprite::Sprite | ( | const [Texture](http://docs.google.com/classsf_1_1Texture.htm) & | *texture* | ) |  | | --- | --- | --- | --- | --- | --- | | explicit |
| --- | --- | --- | --- | --- | --- | --- | --- |

Construct the sprite from a source texture.

Parameters

| texture | Source texture |
| --- | --- |

See Also[setTexture](http://docs.google.com/classsf_1_1Sprite.htm#a3729c88d88ac38c19317c18e87242560)

| sf::Sprite::Sprite | ( | const [Texture](http://docs.google.com/classsf_1_1Texture.htm) & | *texture*, |
| --- | --- | --- | --- |
|  |  | const [IntRect](http://docs.google.com/classsf_1_1Rect.htm) & | *rectangle* |
|  | ) |  |  |

Construct the sprite from a sub-rectangle of a source texture.

Parameters

| texture | Source texture |
| --- | --- |
| rectangle | Sub-rectangle of the texture to assign to the sprite |

See Also[setTexture](http://docs.google.com/classsf_1_1Sprite.htm#a3729c88d88ac38c19317c18e87242560), [setTextureRect](http://docs.google.com/classsf_1_1Sprite.htm#a3fefec419a4e6a90c0fd54c793d82ec2)

## Member Function Documentation

| const [Color](http://docs.google.com/classsf_1_1Color.htm)& sf::Sprite::getColor | ( |  | ) | const |
| --- | --- | --- | --- | --- |

Get the global color of the sprite.

ReturnsGlobal color of the sprite See Also[setColor](http://docs.google.com/classsf_1_1Sprite.htm#a14def44da6437bfea20c4df5e71aba4c)

| [FloatRect](http://docs.google.com/classsf_1_1Rect.htm) sf::Sprite::getGlobalBounds | ( |  | ) | const |
| --- | --- | --- | --- | --- |

Get the global bounding rectangle of the entity.

The returned rectangle is in global coordinates, which means that it takes in account the transformations (translation, rotation, scale, ...) that are applied to the entity. In other words, this function returns the bounds of the sprite in the global 2D world's coordinate system.

ReturnsGlobal bounding rectangle of the entity

| | const [Transform](http://docs.google.com/classsf_1_1Transform.htm)& sf::Transformable::getInverseTransform | ( |  | ) | const | | --- | --- | --- | --- | --- | | inherited |
| --- | --- | --- | --- | --- | --- | --- |

get the inverse of the combined transform of the object

ReturnsInverse of the combined transformations applied to the object See Also[getTransform](http://docs.google.com/classsf_1_1Transformable.htm#a3b48c3362e3e2c14fef7551252deb7bb)

| [FloatRect](http://docs.google.com/classsf_1_1Rect.htm) sf::Sprite::getLocalBounds | ( |  | ) | const |
| --- | --- | --- | --- | --- |

Get the local bounding rectangle of the entity.

The returned rectangle is in local coordinates, which means that it ignores the transformations (translation, rotation, scale, ...) that are applied to the entity. In other words, this function returns the bounds of the entity in the entity's coordinate system.

ReturnsLocal bounding rectangle of the entity

| | const [Vector2f](http://docs.google.com/classsf_1_1Vector2.htm)& sf::Transformable::getOrigin | ( |  | ) | const | | --- | --- | --- | --- | --- | | inherited |
| --- | --- | --- | --- | --- | --- | --- |

get the local origin of the object

ReturnsCurrent origin See Also[setOrigin](http://docs.google.com/classsf_1_1Transformable.htm#a56c67bd80aae8418d13fb96c034d25ec)

| | const [Vector2f](http://docs.google.com/classsf_1_1Vector2.htm)& sf::Transformable::getPosition | ( |  | ) | const | | --- | --- | --- | --- | --- | | inherited |
| --- | --- | --- | --- | --- | --- | --- |

get the position of the object

ReturnsCurrent position See Also[setPosition](http://docs.google.com/classsf_1_1Transformable.htm#a4dbfb1a7c80688b0b4c477d706550208)

| | float sf::Transformable::getRotation | ( |  | ) | const | | --- | --- | --- | --- | --- | | inherited |
| --- | --- | --- | --- | --- | --- | --- |

get the orientation of the object

The rotation is always in the range [0, 360].

ReturnsCurrent rotation, in degrees See Also[setRotation](http://docs.google.com/classsf_1_1Transformable.htm#a32baf2bf1a74699b03bf8c95030a38ed)

| | const [Vector2f](http://docs.google.com/classsf_1_1Vector2.htm)& sf::Transformable::getScale | ( |  | ) | const | | --- | --- | --- | --- | --- | | inherited |
| --- | --- | --- | --- | --- | --- | --- |

get the current scale of the object

ReturnsCurrent scale factors See Also[setScale](http://docs.google.com/classsf_1_1Transformable.htm#aaec50b46b3f41b054763304d1e727471)

| const [Texture](http://docs.google.com/classsf_1_1Texture.htm)\* sf::Sprite::getTexture | ( |  | ) | const |
| --- | --- | --- | --- | --- |

Get the source texture of the sprite.

If the sprite has no source texture, a NULL pointer is returned. The returned pointer is const, which means that you can't modify the texture when you retrieve it with this function.

ReturnsPointer to the sprite's texture See Also[setTexture](http://docs.google.com/classsf_1_1Sprite.htm#a3729c88d88ac38c19317c18e87242560)

| const [IntRect](http://docs.google.com/classsf_1_1Rect.htm)& sf::Sprite::getTextureRect | ( |  | ) | const |
| --- | --- | --- | --- | --- |

Get the sub-rectangle of the texture displayed by the sprite.

Returns[Texture](http://docs.google.com/classsf_1_1Texture.htm) rectangle of the sprite See Also[setTextureRect](http://docs.google.com/classsf_1_1Sprite.htm#a3fefec419a4e6a90c0fd54c793d82ec2)

| | const [Transform](http://docs.google.com/classsf_1_1Transform.htm)& sf::Transformable::getTransform | ( |  | ) | const | | --- | --- | --- | --- | --- | | inherited |
| --- | --- | --- | --- | --- | --- | --- |

get the combined transform of the object

Returns[Transform](http://docs.google.com/classsf_1_1Transform.htm) combining the position/rotation/scale/origin of the object See Also[getInverseTransform](http://docs.google.com/classsf_1_1Transformable.htm#ab00de62b5d1efb2ee4cf2566dea98175)

| | void sf::Transformable::move | ( | float | *offsetX*, | | --- | --- | --- | --- | |  |  | float | *offsetY* | |  | ) |  |  | | inherited |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

Move the object by a given offset.

This function adds to the current position of the object, unlike setPosition which overwrites it. Thus, it is equivalent to the following code:

[sf::Vector2f](http://docs.google.com/classsf_1_1Vector2.htm) pos = object.getPosition();

object.setPosition(pos.[x](http://docs.google.com/classsf_1_1Vector2.htm#a1e6ad77fa155f3753bfb92699bd28141) + offsetX, pos.[y](http://docs.google.com/classsf_1_1Vector2.htm#a420f2481b015f4eb929c75f2af564299) + offsetY);

Parameters

| offsetX | X offset |
| --- | --- |
| offsetY | Y offset |

See Also[setPosition](http://docs.google.com/classsf_1_1Transformable.htm#a4dbfb1a7c80688b0b4c477d706550208)

| | void sf::Transformable::move | ( | const [Vector2f](http://docs.google.com/classsf_1_1Vector2.htm) & | *offset* | ) |  | | --- | --- | --- | --- | --- | --- | | inherited |
| --- | --- | --- | --- | --- | --- | --- | --- |

Move the object by a given offset.

This function adds to the current position of the object, unlike setPosition which overwrites it. Thus, it is equivalent to the following code:

object.setPosition(object.[getPosition](http://docs.google.com/classsf_1_1Transformable.htm#a6a0552d8cf155b7df25f6ceda8ee45a5)() + offset);

Parameters

| offset | Offset |
| --- | --- |

See Also[setPosition](http://docs.google.com/classsf_1_1Transformable.htm#a4dbfb1a7c80688b0b4c477d706550208)

| | void sf::Transformable::rotate | ( | float | *angle* | ) |  | | --- | --- | --- | --- | --- | --- | | inherited |
| --- | --- | --- | --- | --- | --- | --- | --- |

Rotate the object.

This function adds to the current rotation of the object, unlike setRotation which overwrites it. Thus, it is equivalent to the following code:

object.setRotation(object.[getRotation](http://docs.google.com/classsf_1_1Transformable.htm#ad783a7e9971398ec613d22455252809e)() + angle);

Parameters

| angle | Angle of rotation, in degrees |
| --- | --- |

| | void sf::Transformable::scale | ( | float | *factorX*, | | --- | --- | --- | --- | |  |  | float | *factorY* | |  | ) |  |  | | inherited |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

Scale the object.

This function multiplies the current scale of the object, unlike setScale which overwrites it. Thus, it is equivalent to the following code:

[sf::Vector2f](http://docs.google.com/classsf_1_1Vector2.htm) scale = object.getScale();

object.setScale(scale.[x](http://docs.google.com/classsf_1_1Vector2.htm#a1e6ad77fa155f3753bfb92699bd28141) \* factorX, scale.[y](http://docs.google.com/classsf_1_1Vector2.htm#a420f2481b015f4eb929c75f2af564299) \* factorY);

Parameters

| factorX | Horizontal scale factor |
| --- | --- |
| factorY | Vertical scale factor |

See Also[setScale](http://docs.google.com/classsf_1_1Transformable.htm#aaec50b46b3f41b054763304d1e727471)

| | void sf::Transformable::scale | ( | const [Vector2f](http://docs.google.com/classsf_1_1Vector2.htm) & | *factor* | ) |  | | --- | --- | --- | --- | --- | --- | | inherited |
| --- | --- | --- | --- | --- | --- | --- | --- |

Scale the object.

This function multiplies the current scale of the object, unlike setScale which overwrites it. Thus, it is equivalent to the following code:

[sf::Vector2f](http://docs.google.com/classsf_1_1Vector2.htm) scale = object.getScale();

object.setScale(scale.[x](http://docs.google.com/classsf_1_1Vector2.htm#a1e6ad77fa155f3753bfb92699bd28141) \* factor.x, scale.[y](http://docs.google.com/classsf_1_1Vector2.htm#a420f2481b015f4eb929c75f2af564299) \* factor.y);

Parameters

| factor | Scale factors |
| --- | --- |

See Also[setScale](http://docs.google.com/classsf_1_1Transformable.htm#aaec50b46b3f41b054763304d1e727471)

| void sf::Sprite::setColor | ( | const [Color](http://docs.google.com/classsf_1_1Color.htm) & | *color* | ) |  |
| --- | --- | --- | --- | --- | --- |

Set the global color of the sprite.

This color is modulated (multiplied) with the sprite's texture. It can be used to colorize the sprite, or change its global opacity. By default, the sprite's color is opaque white.

Parameters

| color | New color of the sprite |
| --- | --- |

See Also[getColor](http://docs.google.com/classsf_1_1Sprite.htm#ab05fafd4e9999608a5cae4985e7b52df)

| | void sf::Transformable::setOrigin | ( | float | *x*, | | --- | --- | --- | --- | |  |  | float | *y* | |  | ) |  |  | | inherited |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

set the local origin of the object

The origin of an object defines the center point for all transformations (position, scale, rotation). The coordinates of this point must be relative to the top-left corner of the object, and ignore all transformations (position, scale, rotation). The default origin of a transformable object is (0, 0).

Parameters

| x | X coordinate of the new origin |
| --- | --- |
| y | Y coordinate of the new origin |

See Also[getOrigin](http://docs.google.com/classsf_1_1Transformable.htm#a6bddc485d22bb64449d9d2d3a99a778f)

| | void sf::Transformable::setOrigin | ( | const [Vector2f](http://docs.google.com/classsf_1_1Vector2.htm) & | *origin* | ) |  | | --- | --- | --- | --- | --- | --- | | inherited |
| --- | --- | --- | --- | --- | --- | --- | --- |

set the local origin of the object

The origin of an object defines the center point for all transformations (position, scale, rotation). The coordinates of this point must be relative to the top-left corner of the object, and ignore all transformations (position, scale, rotation). The default origin of a transformable object is (0, 0).

Parameters

| origin | New origin |
| --- | --- |

See Also[getOrigin](http://docs.google.com/classsf_1_1Transformable.htm#a6bddc485d22bb64449d9d2d3a99a778f)

| | void sf::Transformable::setPosition | ( | float | *x*, | | --- | --- | --- | --- | |  |  | float | *y* | |  | ) |  |  | | inherited |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

set the position of the object

This function completely overwrites the previous position. See the move function to apply an offset based on the previous position instead. The default position of a transformable object is (0, 0).

Parameters

| x | X coordinate of the new position |
| --- | --- |
| y | Y coordinate of the new position |

See Also[move](http://docs.google.com/classsf_1_1Transformable.htm#a86b461d6a941ad390c2ad8b6a4a20391), [getPosition](http://docs.google.com/classsf_1_1Transformable.htm#a6a0552d8cf155b7df25f6ceda8ee45a5)

| | void sf::Transformable::setPosition | ( | const [Vector2f](http://docs.google.com/classsf_1_1Vector2.htm) & | *position* | ) |  | | --- | --- | --- | --- | --- | --- | | inherited |
| --- | --- | --- | --- | --- | --- | --- | --- |

set the position of the object

This function completely overwrites the previous position. See the move function to apply an offset based on the previous position instead. The default position of a transformable object is (0, 0).

Parameters

| position | New position |
| --- | --- |

See Also[move](http://docs.google.com/classsf_1_1Transformable.htm#a86b461d6a941ad390c2ad8b6a4a20391), [getPosition](http://docs.google.com/classsf_1_1Transformable.htm#a6a0552d8cf155b7df25f6ceda8ee45a5)

| | void sf::Transformable::setRotation | ( | float | *angle* | ) |  | | --- | --- | --- | --- | --- | --- | | inherited |
| --- | --- | --- | --- | --- | --- | --- | --- |

set the orientation of the object

This function completely overwrites the previous rotation. See the rotate function to add an angle based on the previous rotation instead. The default rotation of a transformable object is 0.

Parameters

| angle | New rotation, in degrees |
| --- | --- |

See Also[rotate](http://docs.google.com/classsf_1_1Transformable.htm#af8a5ffddc0d93f238fee3bf8efe1ebda), [getRotation](http://docs.google.com/classsf_1_1Transformable.htm#ad783a7e9971398ec613d22455252809e)

| | void sf::Transformable::setScale | ( | float | *factorX*, | | --- | --- | --- | --- | |  |  | float | *factorY* | |  | ) |  |  | | inherited |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

set the scale factors of the object

This function completely overwrites the previous scale. See the scale function to add a factor based on the previous scale instead. The default scale of a transformable object is (1, 1).

Parameters

| factorX | New horizontal scale factor |
| --- | --- |
| factorY | New vertical scale factor |

See Also[scale](http://docs.google.com/classsf_1_1Transformable.htm#a3de0c6d8957f3cf318092f3f60656391), [getScale](http://docs.google.com/classsf_1_1Transformable.htm#a3ea9639abd7a430ac99afb0aaf1ea562)

| | void sf::Transformable::setScale | ( | const [Vector2f](http://docs.google.com/classsf_1_1Vector2.htm) & | *factors* | ) |  | | --- | --- | --- | --- | --- | --- | | inherited |
| --- | --- | --- | --- | --- | --- | --- | --- |

set the scale factors of the object

This function completely overwrites the previous scale. See the scale function to add a factor based on the previous scale instead. The default scale of a transformable object is (1, 1).

Parameters

| factors | New scale factors |
| --- | --- |

See Also[scale](http://docs.google.com/classsf_1_1Transformable.htm#a3de0c6d8957f3cf318092f3f60656391), [getScale](http://docs.google.com/classsf_1_1Transformable.htm#a3ea9639abd7a430ac99afb0aaf1ea562)

| void sf::Sprite::setTexture | ( | const [Texture](http://docs.google.com/classsf_1_1Texture.htm) & | *texture*, |
| --- | --- | --- | --- |
|  |  | bool | *resetRect* = false |
|  | ) |  |  |

Change the source texture of the sprite.

The *texture* argument refers to a texture that must exist as long as the sprite uses it. Indeed, the sprite doesn't store its own copy of the texture, but rather keeps a pointer to the one that you passed to this function. If the source texture is destroyed and the sprite tries to use it, the behaviour is undefined. If *resetRect* is true, the TextureRect property of the sprite is automatically adjusted to the size of the new texture. If it is false, the texture rect is left unchanged.

Parameters

| texture | New texture |
| --- | --- |
| resetRect | Should the texture rect be reset to the size of the new texture? |

See Also[getTexture](http://docs.google.com/classsf_1_1Sprite.htm#a2c4fbb983b29e14f9bb69a3a75feae46), [setTextureRect](http://docs.google.com/classsf_1_1Sprite.htm#a3fefec419a4e6a90c0fd54c793d82ec2)

| void sf::Sprite::setTextureRect | ( | const [IntRect](http://docs.google.com/classsf_1_1Rect.htm) & | *rectangle* | ) |  |
| --- | --- | --- | --- | --- | --- |

Set the sub-rectangle of the texture that the sprite will display.

The texture rect is useful when you don't want to display the whole texture, but rather a part of it. By default, the texture rect covers the entire texture.

Parameters

| rectangle | Rectangle defining the region of the texture to display |
| --- | --- |

See Also[getTextureRect](http://docs.google.com/classsf_1_1Sprite.htm#a89aa58bc39e2072c6afe7c547d27b5a0), [setTexture](http://docs.google.com/classsf_1_1Sprite.htm#a3729c88d88ac38c19317c18e87242560)

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

* [Sprite.hpp](http://docs.google.com/Sprite_8hpp_source.htm)

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