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**
* [Text](http://docs.google.com/classsf_1_1Text.htm)

[Public Types](#_gjdgxs) | [Public Member Functions](#_30j0zll) | [List of all members](http://docs.google.com/classsf_1_1Text-members.htm)

sf::Text Class Reference

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

Graphical text that can be drawn to a render target. [More...](http://docs.google.com/classsf_1_1Text.htm#details)

#include <[Text.hpp](http://docs.google.com/Text_8hpp_source.htm)>

Inheritance diagram for sf::Text:



| Public Types | |
| --- | --- |
| enum | [Style](http://docs.google.com/classsf_1_1Text.htm#aa8add4aef484c6e6b20faff07452bd82) {  [Regular](http://docs.google.com/classsf_1_1Text.htm#aa8add4aef484c6e6b20faff07452bd82a2af9ae5e1cda126570f744448e0caa32) = 0,  [Bold](http://docs.google.com/classsf_1_1Text.htm#aa8add4aef484c6e6b20faff07452bd82af1b47f98fb1e10509ba930a596987171) = 1 << 0,  [Italic](http://docs.google.com/classsf_1_1Text.htm#aa8add4aef484c6e6b20faff07452bd82aee249eb803848723c542c2062ebe69d8) = 1 << 1,  [Underlined](http://docs.google.com/classsf_1_1Text.htm#aa8add4aef484c6e6b20faff07452bd82a664bd143f92b6e8c709d7f788e8b20df) = 1 << 2  } |
|  | Enumeration of the string drawing styles. [More...](http://docs.google.com/classsf_1_1Text.htm#aa8add4aef484c6e6b20faff07452bd82) |
|  | |

| Public Member Functions | |
| --- | --- |
|  | [Text](http://docs.google.com/classsf_1_1Text.htm#aff7cab6a92e5948c9d1481cb2d87eb84) () |
|  | Default constructor. |
|  | |
|  | [Text](http://docs.google.com/classsf_1_1Text.htm#a614019e0b5c0ed39a99d32483a51f2c5) (const [String](http://docs.google.com/classsf_1_1String.htm) &string, const [Font](http://docs.google.com/classsf_1_1Font.htm) &font, unsigned int characterSize=30) |
|  | Construct the text from a string, font and size. |
|  | |
| void | [setString](http://docs.google.com/classsf_1_1Text.htm#a7d3b3359f286fd9503d1ced25b7b6c33) (const [String](http://docs.google.com/classsf_1_1String.htm) &string) |
|  | Set the text's string. |
|  | |
| void | [setFont](http://docs.google.com/classsf_1_1Text.htm#a2927805d1ae92d57f15034ea34756b81) (const [Font](http://docs.google.com/classsf_1_1Font.htm) &font) |
|  | Set the text's font. |
|  | |
| void | [setCharacterSize](http://docs.google.com/classsf_1_1Text.htm#ae96f835fc1bff858f8a23c5b01eaaf7e) (unsigned int size) |
|  | Set the character size. |
|  | |
| void | [setStyle](http://docs.google.com/classsf_1_1Text.htm#ad791702bc2d1b6590a1719aa60635edf) (Uint32 style) |
|  | Set the text's style. |
|  | |
| void | [setColor](http://docs.google.com/classsf_1_1Text.htm#afd1742fca1adb6b0ea98357250ffb634) (const [Color](http://docs.google.com/classsf_1_1Color.htm) &color) |
|  | Set the global color of the text. |
|  | |
| const [String](http://docs.google.com/classsf_1_1String.htm) & | [getString](http://docs.google.com/classsf_1_1Text.htm#a14d580e8afdd43c210429505310ecc95) () const |
|  | Get the text's string. |
|  | |
| const [Font](http://docs.google.com/classsf_1_1Font.htm) \* | [getFont](http://docs.google.com/classsf_1_1Text.htm#ab831de193307ab591b34221440613aa1) () const |
|  | Get the text's font. |
|  | |
| unsigned int | [getCharacterSize](http://docs.google.com/classsf_1_1Text.htm#a9abb85c6966c9879f6ba4d6e47be1dd5) () const |
|  | Get the character size. |
|  | |
| Uint32 | [getStyle](http://docs.google.com/classsf_1_1Text.htm#a3f7483a48faf66378da19d36ff6145cf) () const |
|  | Get the text's style. |
|  | |
| const [Color](http://docs.google.com/classsf_1_1Color.htm) & | [getColor](http://docs.google.com/classsf_1_1Text.htm#ae42818342a74a9d04644e2fbbd4ca29a) () const |
|  | Get the global color of the text. |
|  | |
| [Vector2f](http://docs.google.com/classsf_1_1Vector2.htm) | [findCharacterPos](http://docs.google.com/classsf_1_1Text.htm#adf120ed4159e43daa782b17fea23ea82) (std::size\_t index) const |
|  | Return the position of the *index-th* character. |
|  | |
| [FloatRect](http://docs.google.com/classsf_1_1Rect.htm) | [getLocalBounds](http://docs.google.com/classsf_1_1Text.htm#a8a766ea03a1b8899cd1542765771a4ae) () 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_1Text.htm#a95d732f58bd12bf7ec388b106f3729ba) () 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

Graphical text that can be drawn to a render target.

[sf::Text](http://docs.google.com/classsf_1_1Text.htm) is a drawable class that allows to easily display some text with custom style and color 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 text-specific properties such as the font to use, the character size, the font style (bold, italic, underlined), the global color and the text to display of course. It also provides convenience functions to calculate the graphical size of the text, or to get the global position of a given character.

[sf::Text](http://docs.google.com/classsf_1_1Text.htm) works in combination with the [sf::Font](http://docs.google.com/classsf_1_1Font.htm) class, which loads and provides the glyphs (visual characters) of a given font.

The separation of [sf::Font](http://docs.google.com/classsf_1_1Font.htm) and [sf::Text](http://docs.google.com/classsf_1_1Text.htm) allows more flexibility and better performances: indeed a [sf::Font](http://docs.google.com/classsf_1_1Font.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::Text](http://docs.google.com/classsf_1_1Text.htm) is a lightweight object which can combine the glyphs data and metrics of a [sf::Font](http://docs.google.com/classsf_1_1Font.htm) to display any text on a render target.

It is important to note that the [sf::Text](http://docs.google.com/classsf_1_1Text.htm) instance doesn't copy the font that it uses, it only keeps a reference to it. Thus, a [sf::Font](http://docs.google.com/classsf_1_1Font.htm) must not be destructed while it is used by a [sf::Text](http://docs.google.com/classsf_1_1Text.htm) (i.e. never write a function that uses a local [sf::Font](http://docs.google.com/classsf_1_1Font.htm) instance for creating a text).

Usage example:

// Declare and load a font

[sf::Font](http://docs.google.com/classsf_1_1Font.htm) font;

font.[loadFromFile](http://docs.google.com/classsf_1_1Font.htm#ab020052ef4e01f6c749a85571c0f3fd1)("arial.ttf");

// Create a text

[sf::Text](http://docs.google.com/classsf_1_1Text.htm) text("hello", font);

text.[setCharacterSize](http://docs.google.com/classsf_1_1Text.htm#ae96f835fc1bff858f8a23c5b01eaaf7e)(30);

text.[setStyle](http://docs.google.com/classsf_1_1Text.htm#ad791702bc2d1b6590a1719aa60635edf)([sf::Text::Bold](http://docs.google.com/classsf_1_1Text.htm#aa8add4aef484c6e6b20faff07452bd82af1b47f98fb1e10509ba930a596987171));

text.[setColor](http://docs.google.com/classsf_1_1Text.htm#afd1742fca1adb6b0ea98357250ffb634)([sf::Color::Red](http://docs.google.com/classsf_1_1Color.htm#a127dbf55db9c07d0fa8f4bfcbb97594a));

// Draw it

window.draw(text);

See Also[sf::Font](http://docs.google.com/classsf_1_1Font.htm), [sf::Transformable](http://docs.google.com/classsf_1_1Transformable.htm)

Definition at line [48](http://docs.google.com/Text_8hpp_source.htm#l00048) of file [Text.hpp](http://docs.google.com/Text_8hpp_source.htm).

## Member Enumeration Documentation

| enum [sf::Text::Style](http://docs.google.com/classsf_1_1Text.htm#aa8add4aef484c6e6b20faff07452bd82) |
| --- |

Enumeration of the string drawing styles.

**Enumerator:**

| *Regular* | Regular characters, no style. |
| --- | --- |
| *Bold* | Bold characters. |
| *Italic* | Italic characters. |
| *Underlined* | Underlined characters. |

Definition at line [56](http://docs.google.com/Text_8hpp_source.htm#l00056) of file [Text.hpp](http://docs.google.com/Text_8hpp_source.htm).

## Constructor & Destructor Documentation

| sf::Text::Text | ( |  | ) |  |
| --- | --- | --- | --- | --- |

Default constructor.

Creates an empty text.

| sf::Text::Text | ( | const [String](http://docs.google.com/classsf_1_1String.htm) & | *string*, |
| --- | --- | --- | --- |
|  |  | const [Font](http://docs.google.com/classsf_1_1Font.htm) & | *font*, |
|  |  | unsigned int | *characterSize* = 30 |
|  | ) |  |  |

Construct the text from a string, font and size.

Parameters

| string | [Text](http://docs.google.com/classsf_1_1Text.htm) assigned to the string |
| --- | --- |
| font | [Font](http://docs.google.com/classsf_1_1Font.htm) used to draw the string |
| characterSize | Base size of characters, in pixels |

## Member Function Documentation

| [Vector2f](http://docs.google.com/classsf_1_1Vector2.htm) sf::Text::findCharacterPos | ( | std::size\_t | *index* | ) | const |
| --- | --- | --- | --- | --- | --- |

Return the position of the *index-th* character.

This function computes the visual position of a character from its index in the string. The returned position is in global coordinates (translation, rotation, scale and origin are applied). If *index* is out of range, the position of the end of the string is returned.

Parameters

| index | Index of the character |
| --- | --- |

ReturnsPosition of the character

| unsigned int sf::Text::getCharacterSize | ( |  | ) | const |
| --- | --- | --- | --- | --- |

Get the character size.

ReturnsSize of the characters, in pixels See Also[setCharacterSize](http://docs.google.com/classsf_1_1Text.htm#ae96f835fc1bff858f8a23c5b01eaaf7e)

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

Get the global color of the text.

ReturnsGlobal color of the text See Also[setColor](http://docs.google.com/classsf_1_1Text.htm#afd1742fca1adb6b0ea98357250ffb634)

| const [Font](http://docs.google.com/classsf_1_1Font.htm)\* sf::Text::getFont | ( |  | ) | const |
| --- | --- | --- | --- | --- |

Get the text's font.

If the text has no font attached, a NULL pointer is returned. The returned reference is const, which means that you cannot modify the font when you get it from this function.

ReturnsPointer to the text's font See Also[setFont](http://docs.google.com/classsf_1_1Text.htm#a2927805d1ae92d57f15034ea34756b81)

| [FloatRect](http://docs.google.com/classsf_1_1Rect.htm) sf::Text::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::Text::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 [String](http://docs.google.com/classsf_1_1String.htm)& sf::Text::getString | ( |  | ) | const |
| --- | --- | --- | --- | --- |

Get the text's string.

The returned string is a [sf::String](http://docs.google.com/classsf_1_1String.htm), which can automatically be converted to standard string types. So, the following lines of code are all valid:

[sf::String](http://docs.google.com/classsf_1_1String.htm) s1 = text.[getString](http://docs.google.com/classsf_1_1Text.htm#a14d580e8afdd43c210429505310ecc95)();

std::string s2 = text.[getString](http://docs.google.com/classsf_1_1Text.htm#a14d580e8afdd43c210429505310ecc95)();

std::wstring s3 = text.[getString](http://docs.google.com/classsf_1_1Text.htm#a14d580e8afdd43c210429505310ecc95)();

Returns[Text](http://docs.google.com/classsf_1_1Text.htm)'s string See Also[setString](http://docs.google.com/classsf_1_1Text.htm#a7d3b3359f286fd9503d1ced25b7b6c33)

| Uint32 sf::Text::getStyle | ( |  | ) | const |
| --- | --- | --- | --- | --- |

Get the text's style.

Returns[Text](http://docs.google.com/classsf_1_1Text.htm)'s style See Also[setStyle](http://docs.google.com/classsf_1_1Text.htm#ad791702bc2d1b6590a1719aa60635edf)

| | 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::Text::setCharacterSize | ( | unsigned int | *size* | ) |  |
| --- | --- | --- | --- | --- | --- |

Set the character size.

The default size is 30.

Parameters

| size | New character size, in pixels |
| --- | --- |

See Also[getCharacterSize](http://docs.google.com/classsf_1_1Text.htm#a9abb85c6966c9879f6ba4d6e47be1dd5)

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

Set the global color of the text.

By default, the text's color is opaque white.

Parameters

| color | New color of the text |
| --- | --- |

See Also[getColor](http://docs.google.com/classsf_1_1Text.htm#ae42818342a74a9d04644e2fbbd4ca29a)

| void sf::Text::setFont | ( | const [Font](http://docs.google.com/classsf_1_1Font.htm) & | *font* | ) |  |
| --- | --- | --- | --- | --- | --- |

Set the text's font.

The *font* argument refers to a font that must exist as long as the text uses it. Indeed, the text doesn't store its own copy of the font, but rather keeps a pointer to the one that you passed to this function. If the font is destroyed and the text tries to use it, the behaviour is undefined.

Parameters

| font | New font |
| --- | --- |

See Also[getFont](http://docs.google.com/classsf_1_1Text.htm#ab831de193307ab591b34221440613aa1)

| | 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::Text::setString | ( | const [String](http://docs.google.com/classsf_1_1String.htm) & | *string* | ) |  |
| --- | --- | --- | --- | --- | --- |

Set the text's string.

The *string* argument is a [sf::String](http://docs.google.com/classsf_1_1String.htm), which can automatically be constructed from standard string types. So, the following calls are all valid:

text.[setString](http://docs.google.com/classsf_1_1Text.htm#a7d3b3359f286fd9503d1ced25b7b6c33)("hello");

text.[setString](http://docs.google.com/classsf_1_1Text.htm#a7d3b3359f286fd9503d1ced25b7b6c33)(L"hello");

text.[setString](http://docs.google.com/classsf_1_1Text.htm#a7d3b3359f286fd9503d1ced25b7b6c33)(std::string("hello"));

text.[setString](http://docs.google.com/classsf_1_1Text.htm#a7d3b3359f286fd9503d1ced25b7b6c33)(std::wstring(L"hello"));

A text's string is empty by default.

Parameters

| string | New string |
| --- | --- |

See Also[getString](http://docs.google.com/classsf_1_1Text.htm#a14d580e8afdd43c210429505310ecc95)

| void sf::Text::setStyle | ( | Uint32 | *style* | ) |  |
| --- | --- | --- | --- | --- | --- |

Set the text's style.

You can pass a combination of one or more styles, for example [sf::Text::Bold](http://docs.google.com/classsf_1_1Text.htm#aa8add4aef484c6e6b20faff07452bd82af1b47f98fb1e10509ba930a596987171) | [sf::Text::Italic](http://docs.google.com/classsf_1_1Text.htm#aa8add4aef484c6e6b20faff07452bd82aee249eb803848723c542c2062ebe69d8). The default style is [sf::Text::Regular](http://docs.google.com/classsf_1_1Text.htm#aa8add4aef484c6e6b20faff07452bd82a2af9ae5e1cda126570f744448e0caa32).

Parameters

| style | New style |
| --- | --- |

See Also[getStyle](http://docs.google.com/classsf_1_1Text.htm#a3f7483a48faf66378da19d36ff6145cf)

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

* [Text.hpp](http://docs.google.com/Text_8hpp_source.htm)

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