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**
* [Transformable](http://docs.google.com/classsf_1_1Transformable.htm)

[Public Member Functions](#_gjdgxs) | [List of all members](http://docs.google.com/classsf_1_1Transformable-members.htm)

sf::Transformable Class Reference

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

Decomposed transform defined by a position, a rotation and a scale. [More...](http://docs.google.com/classsf_1_1Transformable.htm#details)

#include <[Transformable.hpp](http://docs.google.com/Transformable_8hpp_source.htm)>

Inheritance diagram for sf::Transformable:



| Public Member Functions | |
| --- | --- |
|  | [Transformable](http://docs.google.com/classsf_1_1Transformable.htm#ae71710de0fef423121bab1c684954a2e) () |
|  | Default constructor. |
|  | |
| virtual | [~Transformable](http://docs.google.com/classsf_1_1Transformable.htm#a43253abcb863195a673c2a347a7425cc) () |
|  | Virtual destructor. |
|  | |
| 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

Decomposed transform defined by a position, a rotation and a scale.

This class is provided for convenience, on top of [sf::Transform](http://docs.google.com/classsf_1_1Transform.htm).

[sf::Transform](http://docs.google.com/classsf_1_1Transform.htm), as a low-level class, offers a great level of flexibility but it is not always convenient to manage. Indeed, one can easily combine any kind of operation, such as a translation followed by a rotation followed by a scaling, but once the result transform is built, there's no way to go backward and, let's say, change only the rotation without modifying the translation and scaling. The entire transform must be recomputed, which means that you need to retrieve the initial translation and scale factors as well, and combine them the same way you did before updating the rotation. This is a tedious operation, and it requires to store all the individual components of the final transform.

That's exactly what [sf::Transformable](http://docs.google.com/classsf_1_1Transformable.htm) was written for: it hides these variables and the composed transform behind an easy to use interface. You can set or get any of the individual components without worrying about the others. It also provides the composed transform (as a [sf::Transform](http://docs.google.com/classsf_1_1Transform.htm)), and keeps it up-to-date.

In addition to the position, rotation and scale, [sf::Transformable](http://docs.google.com/classsf_1_1Transformable.htm) provides an "origin" component, which represents the local origin of the three other components. Let's take an example with a 10x10 pixels sprite. By default, the sprite is positionned/rotated/scaled relatively to its top-left corner, because it is the local point (0, 0). But if we change the origin to be (5, 5), the sprite will be positionned/rotated/scaled around its center instead. And if we set the origin to (10, 10), it will be transformed around its bottom-right corner.

To keep the [sf::Transformable](http://docs.google.com/classsf_1_1Transformable.htm) class simple, there's only one origin for all the components. You cannot position the sprite relatively to its top-left corner while rotating it around its center, for example. To do such things, use [sf::Transform](http://docs.google.com/classsf_1_1Transform.htm) directly.

[sf::Transformable](http://docs.google.com/classsf_1_1Transformable.htm) can be used as a base class. It is often combined with [sf::Drawable](http://docs.google.com/classsf_1_1Drawable.htm) – that's what SFML's sprites, texts and shapes do.

class MyEntity : public [sf::Transformable](http://docs.google.com/classsf_1_1Transformable.htm), public [sf::Drawable](http://docs.google.com/classsf_1_1Drawable.htm)

{

virtual void draw([sf::RenderTarget](http://docs.google.com/classsf_1_1RenderTarget.htm)& target, [sf::RenderStates](http://docs.google.com/classsf_1_1RenderStates.htm) states) const

{

states.[transform](http://docs.google.com/classsf_1_1RenderStates.htm#a1f737981a0f2f0d4bb8dac866a8d1149) \*= [getTransform](http://docs.google.com/classsf_1_1Transformable.htm#a3b48c3362e3e2c14fef7551252deb7bb)();

target.[draw](http://docs.google.com/classsf_1_1RenderTarget.htm#a12417a3bcc245c41d957b29583556f39)(..., states);

}

};

MyEntity entity;

entity.setPosition(10, 20);

entity.setRotation(45);

window.draw(entity);

It can also be used as a member, if you don't want to use its API directly (because you don't need all its functions, or you have different naming conventions for example).

class MyEntity

{

public :

void SetPosition(const MyVector& v)

{

myTransform.setPosition(v.x(), v.y());

}

void Draw([sf::RenderTarget](http://docs.google.com/classsf_1_1RenderTarget.htm)& target) const

{

target.[draw](http://docs.google.com/classsf_1_1RenderTarget.htm#a12417a3bcc245c41d957b29583556f39)(..., myTransform.getTransform());

}

private :

[sf::Transformable](http://docs.google.com/classsf_1_1Transformable.htm) myTransform;

};

See Also[sf::Transform](http://docs.google.com/classsf_1_1Transform.htm)

Definition at line [41](http://docs.google.com/Transformable_8hpp_source.htm#l00041) of file [Transformable.hpp](http://docs.google.com/Transformable_8hpp_source.htm).

## Constructor & Destructor Documentation

| sf::Transformable::Transformable | ( |  | ) |  |
| --- | --- | --- | --- | --- |

Default constructor.

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

Virtual destructor.

## Member Function Documentation

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

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)

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

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

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

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

get the current scale of the object

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

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

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* |
|  | ) |  |  |

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* | ) |  |
| --- | --- | --- | --- | --- | --- |

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* | ) |  |
| --- | --- | --- | --- | --- | --- |

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* |
|  | ) |  |  |

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* | ) |  |
| --- | --- | --- | --- | --- | --- |

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

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* | ) |  |
| --- | --- | --- | --- | --- | --- |

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* |
|  | ) |  |  |

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* | ) |  |
| --- | --- | --- | --- | --- | --- |

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* | ) |  |
| --- | --- | --- | --- | --- | --- |

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* |
|  | ) |  |  |

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* | ) |  |
| --- | --- | --- | --- | --- | --- |

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)

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

* [Transformable.hpp](http://docs.google.com/Transformable_8hpp_source.htm)

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