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**
* [Transform](http://docs.google.com/classsf_1_1Transform.htm)

[Public Member Functions](#_gjdgxs) | [Static Public Attributes](#_30j0zll) | [Related Functions](#_1fob9te) | [List of all members](http://docs.google.com/classsf_1_1Transform-members.htm)

sf::Transform Class Reference

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

Define a 3x3 transform matrix. [More...](http://docs.google.com/classsf_1_1Transform.htm#details)

#include <[Transform.hpp](http://docs.google.com/Transform_8hpp_source.htm)>

| Public Member Functions | |
| --- | --- |
|  | [Transform](http://docs.google.com/classsf_1_1Transform.htm#ac32de51bd0b9f3d52fbe0838225ee83b) () |
|  | Default constructor. |
|  | |
|  | [Transform](http://docs.google.com/classsf_1_1Transform.htm#a78c48677712fcf41122d02f1301d71a3) (float a00, float a01, float a02, float a10, float a11, float a12, float a20, float a21, float a22) |
|  | Construct a transform from a 3x3 matrix. |
|  | |
| const float \* | [getMatrix](http://docs.google.com/classsf_1_1Transform.htm#ae4c1969c47533e2b01deb526ff73b37f) () const |
|  | Return the transform as a 4x4 matrix. |
|  | |
| [Transform](http://docs.google.com/classsf_1_1Transform.htm) | [getInverse](http://docs.google.com/classsf_1_1Transform.htm#ab1c033198b0aae8cdb9daa3d3bef3fc1) () const |
|  | Return the inverse of the transform. |
|  | |
| [Vector2f](http://docs.google.com/classsf_1_1Vector2.htm) | [transformPoint](http://docs.google.com/classsf_1_1Transform.htm#af20913c6a27087c26192c116397ab40a) (float x, float y) const |
|  | [Transform](http://docs.google.com/classsf_1_1Transform.htm) a 2D point. |
|  | |
| [Vector2f](http://docs.google.com/classsf_1_1Vector2.htm) | [transformPoint](http://docs.google.com/classsf_1_1Transform.htm#ac322cd8f6d606598d1aacc4d1d160ad6) (const [Vector2f](http://docs.google.com/classsf_1_1Vector2.htm) &point) const |
|  | [Transform](http://docs.google.com/classsf_1_1Transform.htm) a 2D point. |
|  | |
| [FloatRect](http://docs.google.com/classsf_1_1Rect.htm) | [transformRect](http://docs.google.com/classsf_1_1Transform.htm#a345112559981d988e92b54b7976fca8a) (const [FloatRect](http://docs.google.com/classsf_1_1Rect.htm) &rectangle) const |
|  | [Transform](http://docs.google.com/classsf_1_1Transform.htm) a rectangle. |
|  | |
| [Transform](http://docs.google.com/classsf_1_1Transform.htm) & | [combine](http://docs.google.com/classsf_1_1Transform.htm#acd978f60421a0f839bb9a8263e8877ff) (const [Transform](http://docs.google.com/classsf_1_1Transform.htm) &transform) |
|  | Combine the current transform with another one. |
|  | |
| [Transform](http://docs.google.com/classsf_1_1Transform.htm) & | [translate](http://docs.google.com/classsf_1_1Transform.htm#ab54f6c8070cc05e2afcb3145fbf4395a) (float x, float y) |
|  | Combine the current transform with a translation. |
|  | |
| [Transform](http://docs.google.com/classsf_1_1Transform.htm) & | [translate](http://docs.google.com/classsf_1_1Transform.htm#a452ff6e32d5120fa8c132c1bf0ad83cd) (const [Vector2f](http://docs.google.com/classsf_1_1Vector2.htm) &offset) |
|  | Combine the current transform with a translation. |
|  | |
| [Transform](http://docs.google.com/classsf_1_1Transform.htm) & | [rotate](http://docs.google.com/classsf_1_1Transform.htm#a3e548c3c9e3fb9d4bd43cf852669e555) (float angle) |
|  | Combine the current transform with a rotation. |
|  | |
| [Transform](http://docs.google.com/classsf_1_1Transform.htm) & | [rotate](http://docs.google.com/classsf_1_1Transform.htm#af0b7cc3fed36d0fa22d5d331a779eee2) (float angle, float centerX, float centerY) |
|  | Combine the current transform with a rotation. |
|  | |
| [Transform](http://docs.google.com/classsf_1_1Transform.htm) & | [rotate](http://docs.google.com/classsf_1_1Transform.htm#ad2a2520ad81724079d109d4a986f9902) (float angle, const [Vector2f](http://docs.google.com/classsf_1_1Vector2.htm) &center) |
|  | Combine the current transform with a rotation. |
|  | |
| [Transform](http://docs.google.com/classsf_1_1Transform.htm) & | [scale](http://docs.google.com/classsf_1_1Transform.htm#a3f46af807f69d74120fb836334268671) (float scaleX, float scaleY) |
|  | Combine the current transform with a scaling. |
|  | |
| [Transform](http://docs.google.com/classsf_1_1Transform.htm) & | [scale](http://docs.google.com/classsf_1_1Transform.htm#a6eaeedd35e289cb17f9bf7f24dc28daa) (float scaleX, float scaleY, float centerX, float centerY) |
|  | Combine the current transform with a scaling. |
|  | |
| [Transform](http://docs.google.com/classsf_1_1Transform.htm) & | [scale](http://docs.google.com/classsf_1_1Transform.htm#a3d57622a7ab309925c9d9887c99cc720) (const [Vector2f](http://docs.google.com/classsf_1_1Vector2.htm) &factors) |
|  | Combine the current transform with a scaling. |
|  | |
| [Transform](http://docs.google.com/classsf_1_1Transform.htm) & | [scale](http://docs.google.com/classsf_1_1Transform.htm#a9198da375173127901f3095e0165ee1b) (const [Vector2f](http://docs.google.com/classsf_1_1Vector2.htm) &factors, const [Vector2f](http://docs.google.com/classsf_1_1Vector2.htm) &center) |
|  | Combine the current transform with a scaling. |
|  | |

| Static Public Attributes | |
| --- | --- |
| static const [Transform](http://docs.google.com/classsf_1_1Transform.htm) | [Identity](http://docs.google.com/classsf_1_1Transform.htm#aa4eb1eecbcb9979d76e2543b337fdb13) |
|  | The identity transform (does nothing) |
|  | |

| Related Functions | |
| --- | --- |
| (Note that these are not member functions.) | |
| [Transform](http://docs.google.com/classsf_1_1Transform.htm) | [operator\*](http://docs.google.com/classsf_1_1Transform.htm#a85ea4e5539795f9b2ceb7d4b06736c8f) (const [Transform](http://docs.google.com/classsf_1_1Transform.htm) &left, const [Transform](http://docs.google.com/classsf_1_1Transform.htm) &right) |
|  | Overload of binary operator \* to combine two transforms. |
|  | |
| [Transform](http://docs.google.com/classsf_1_1Transform.htm) & | [operator\*=](http://docs.google.com/classsf_1_1Transform.htm#a189899674616490f6250953ac581ac30) ([Transform](http://docs.google.com/classsf_1_1Transform.htm) &left, const [Transform](http://docs.google.com/classsf_1_1Transform.htm) &right) |
|  | Overload of binary operator \*= to combine two transforms. |
|  | |
| [Vector2f](http://docs.google.com/classsf_1_1Vector2.htm) | [operator\*](http://docs.google.com/classsf_1_1Transform.htm#a4eeee49125c3c72c250062eef35ceb75) (const [Transform](http://docs.google.com/classsf_1_1Transform.htm) &left, const [Vector2f](http://docs.google.com/classsf_1_1Vector2.htm) &right) |
|  | Overload of binary operator \* to transform a point. |
|  | |

## Detailed Description

Define a 3x3 transform matrix.

A [sf::Transform](http://docs.google.com/classsf_1_1Transform.htm) specifies how to translate, rotate, scale, shear, project, whatever things.

In mathematical terms, it defines how to transform a coordinate system into another.

For example, if you apply a rotation transform to a sprite, the result will be a rotated sprite. And anything that is transformed by this rotation transform will be rotated the same way, according to its initial position.

Transforms are typically used for drawing. But they can also be used for any computation that requires to transform points between the local and global coordinate systems of an entity (like collision detection).

Example:

// define a translation transform

[sf::Transform](http://docs.google.com/classsf_1_1Transform.htm) translation;

translation.[translate](http://docs.google.com/classsf_1_1Transform.htm#ab54f6c8070cc05e2afcb3145fbf4395a)(20, 50);

// define a rotation transform

[sf::Transform](http://docs.google.com/classsf_1_1Transform.htm) rotation;

rotation.[rotate](http://docs.google.com/classsf_1_1Transform.htm#a3e548c3c9e3fb9d4bd43cf852669e555)(45);

// combine them

[sf::Transform](http://docs.google.com/classsf_1_1Transform.htm) transform = translation \* rotation;

// use the result to transform stuff...

[sf::Vector2f](http://docs.google.com/classsf_1_1Vector2.htm) point = transform.[transformPoint](http://docs.google.com/classsf_1_1Transform.htm#af20913c6a27087c26192c116397ab40a)(10, 20);

[sf::FloatRect](http://docs.google.com/classsf_1_1Rect.htm) rect = transform.[transformRect](http://docs.google.com/classsf_1_1Transform.htm#a345112559981d988e92b54b7976fca8a)([sf::FloatRect](http://docs.google.com/classsf_1_1Rect.htm)(0, 0, 10, 100));

See Also[sf::Transformable](http://docs.google.com/classsf_1_1Transformable.htm), [sf::RenderStates](http://docs.google.com/classsf_1_1RenderStates.htm)

Definition at line [42](http://docs.google.com/Transform_8hpp_source.htm#l00042) of file [Transform.hpp](http://docs.google.com/Transform_8hpp_source.htm).

## Constructor & Destructor Documentation

| sf::Transform::Transform | ( |  | ) |  |
| --- | --- | --- | --- | --- |

Default constructor.

Creates an identity transform (a transform that does nothing).

| sf::Transform::Transform | ( | float | *a00*, |
| --- | --- | --- | --- |
|  |  | float | *a01*, |
|  |  | float | *a02*, |
|  |  | float | *a10*, |
|  |  | float | *a11*, |
|  |  | float | *a12*, |
|  |  | float | *a20*, |
|  |  | float | *a21*, |
|  |  | float | *a22* |
|  | ) |  |  |

Construct a transform from a 3x3 matrix.

Parameters

| a00 | Element (0, 0) of the matrix |
| --- | --- |
| a01 | Element (0, 1) of the matrix |
| a02 | Element (0, 2) of the matrix |
| a10 | Element (1, 0) of the matrix |
| a11 | Element (1, 1) of the matrix |
| a12 | Element (1, 2) of the matrix |
| a20 | Element (2, 0) of the matrix |
| a21 | Element (2, 1) of the matrix |
| a22 | Element (2, 2) of the matrix |

## Member Function Documentation

| [Transform](http://docs.google.com/classsf_1_1Transform.htm)& sf::Transform::combine | ( | const [Transform](http://docs.google.com/classsf_1_1Transform.htm) & | *transform* | ) |  |
| --- | --- | --- | --- | --- | --- |

Combine the current transform with another one.

The result is a transform that is equivalent to applying \*this followed by *transform*. Mathematically, it is equivalent to a matrix multiplication.

Parameters

| transform | [Transform](http://docs.google.com/classsf_1_1Transform.htm) to combine with this transform |
| --- | --- |

ReturnsReference to \*this

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

Return the inverse of the transform.

If the inverse cannot be computed, an identity transform is returned.

ReturnsA new transform which is the inverse of self

| const float\* sf::Transform::getMatrix | ( |  | ) | const |
| --- | --- | --- | --- | --- |

Return the transform as a 4x4 matrix.

This function returns a pointer to an array of 16 floats containing the transform elements as a 4x4 matrix, which is directly compatible with OpenGL functions.

[sf::Transform](http://docs.google.com/classsf_1_1Transform.htm) transform = ...;

glLoadMatrixf(transform.[getMatrix](http://docs.google.com/classsf_1_1Transform.htm#ae4c1969c47533e2b01deb526ff73b37f)());

ReturnsPointer to a 4x4 matrix

| [Transform](http://docs.google.com/classsf_1_1Transform.htm)& sf::Transform::rotate | ( | float | *angle* | ) |  |
| --- | --- | --- | --- | --- | --- |

Combine the current transform with a rotation.

This function returns a reference to \*this, so that calls can be chained.

[sf::Transform](http://docs.google.com/classsf_1_1Transform.htm) transform;

transform.[rotate](http://docs.google.com/classsf_1_1Transform.htm#a3e548c3c9e3fb9d4bd43cf852669e555)(90).[translate](http://docs.google.com/classsf_1_1Transform.htm#ab54f6c8070cc05e2afcb3145fbf4395a)(50, 20);

Parameters

| angle | Rotation angle, in degrees |
| --- | --- |

ReturnsReference to \*this See Also[translate](http://docs.google.com/classsf_1_1Transform.htm#ab54f6c8070cc05e2afcb3145fbf4395a), [scale](http://docs.google.com/classsf_1_1Transform.htm#a3f46af807f69d74120fb836334268671)

| [Transform](http://docs.google.com/classsf_1_1Transform.htm)& sf::Transform::rotate | ( | float | *angle*, |
| --- | --- | --- | --- |
|  |  | float | *centerX*, |
|  |  | float | *centerY* |
|  | ) |  |  |

Combine the current transform with a rotation.

The center of rotation is provided for convenience as a second argument, so that you can build rotations around arbitrary points more easily (and efficiently) than the usual translate(-center).rotate(angle).translate(center).

This function returns a reference to \*this, so that calls can be chained.

[sf::Transform](http://docs.google.com/classsf_1_1Transform.htm) transform;

transform.[rotate](http://docs.google.com/classsf_1_1Transform.htm#a3e548c3c9e3fb9d4bd43cf852669e555)(90, 8, 3).[translate](http://docs.google.com/classsf_1_1Transform.htm#ab54f6c8070cc05e2afcb3145fbf4395a)(50, 20);

Parameters

| angle | Rotation angle, in degrees |
| --- | --- |
| centerX | X coordinate of the center of rotation |
| centerY | Y coordinate of the center of rotation |

ReturnsReference to \*this See Also[translate](http://docs.google.com/classsf_1_1Transform.htm#ab54f6c8070cc05e2afcb3145fbf4395a), [scale](http://docs.google.com/classsf_1_1Transform.htm#a3f46af807f69d74120fb836334268671)

| [Transform](http://docs.google.com/classsf_1_1Transform.htm)& sf::Transform::rotate | ( | float | *angle*, |
| --- | --- | --- | --- |
|  |  | const [Vector2f](http://docs.google.com/classsf_1_1Vector2.htm) & | *center* |
|  | ) |  |  |

Combine the current transform with a rotation.

The center of rotation is provided for convenience as a second argument, so that you can build rotations around arbitrary points more easily (and efficiently) than the usual translate(-center).rotate(angle).translate(center).

This function returns a reference to \*this, so that calls can be chained.

[sf::Transform](http://docs.google.com/classsf_1_1Transform.htm) transform;

transform.[rotate](http://docs.google.com/classsf_1_1Transform.htm#a3e548c3c9e3fb9d4bd43cf852669e555)(90, [sf::Vector2f](http://docs.google.com/classsf_1_1Vector2.htm)(8, 3)).[translate](http://docs.google.com/classsf_1_1Transform.htm#ab54f6c8070cc05e2afcb3145fbf4395a)([sf::Vector2f](http://docs.google.com/classsf_1_1Vector2.htm)(50, 20));

Parameters

| angle | Rotation angle, in degrees |
| --- | --- |
| center | Center of rotation |

ReturnsReference to \*this See Also[translate](http://docs.google.com/classsf_1_1Transform.htm#ab54f6c8070cc05e2afcb3145fbf4395a), [scale](http://docs.google.com/classsf_1_1Transform.htm#a3f46af807f69d74120fb836334268671)

| [Transform](http://docs.google.com/classsf_1_1Transform.htm)& sf::Transform::scale | ( | float | *scaleX*, |
| --- | --- | --- | --- |
|  |  | float | *scaleY* |
|  | ) |  |  |

Combine the current transform with a scaling.

This function returns a reference to \*this, so that calls can be chained.

[sf::Transform](http://docs.google.com/classsf_1_1Transform.htm) transform;

transform.[scale](http://docs.google.com/classsf_1_1Transform.htm#a3f46af807f69d74120fb836334268671)(2, 1).[rotate](http://docs.google.com/classsf_1_1Transform.htm#a3e548c3c9e3fb9d4bd43cf852669e555)(45);

Parameters

| scaleX | Scaling factor on the X axis |
| --- | --- |
| scaleY | Scaling factor on the Y axis |

ReturnsReference to \*this See Also[translate](http://docs.google.com/classsf_1_1Transform.htm#ab54f6c8070cc05e2afcb3145fbf4395a), [rotate](http://docs.google.com/classsf_1_1Transform.htm#a3e548c3c9e3fb9d4bd43cf852669e555)

| [Transform](http://docs.google.com/classsf_1_1Transform.htm)& sf::Transform::scale | ( | float | *scaleX*, |
| --- | --- | --- | --- |
|  |  | float | *scaleY*, |
|  |  | float | *centerX*, |
|  |  | float | *centerY* |
|  | ) |  |  |

Combine the current transform with a scaling.

The center of scaling is provided for convenience as a second argument, so that you can build scaling around arbitrary points more easily (and efficiently) than the usual translate(-center).scale(factors).translate(center).

This function returns a reference to \*this, so that calls can be chained.

[sf::Transform](http://docs.google.com/classsf_1_1Transform.htm) transform;

transform.[scale](http://docs.google.com/classsf_1_1Transform.htm#a3f46af807f69d74120fb836334268671)(2, 1, 8, 3).[rotate](http://docs.google.com/classsf_1_1Transform.htm#a3e548c3c9e3fb9d4bd43cf852669e555)(45);

Parameters

| scaleX | Scaling factor on X axis |
| --- | --- |
| scaleY | Scaling factor on Y axis |
| centerX | X coordinate of the center of scaling |
| centerY | Y coordinate of the center of scaling |

ReturnsReference to \*this See Also[translate](http://docs.google.com/classsf_1_1Transform.htm#ab54f6c8070cc05e2afcb3145fbf4395a), [rotate](http://docs.google.com/classsf_1_1Transform.htm#a3e548c3c9e3fb9d4bd43cf852669e555)

| [Transform](http://docs.google.com/classsf_1_1Transform.htm)& sf::Transform::scale | ( | const [Vector2f](http://docs.google.com/classsf_1_1Vector2.htm) & | *factors* | ) |  |
| --- | --- | --- | --- | --- | --- |

Combine the current transform with a scaling.

This function returns a reference to \*this, so that calls can be chained.

[sf::Transform](http://docs.google.com/classsf_1_1Transform.htm) transform;

transform.[scale](http://docs.google.com/classsf_1_1Transform.htm#a3f46af807f69d74120fb836334268671)([sf::Vector2f](http://docs.google.com/classsf_1_1Vector2.htm)(2, 1)).[rotate](http://docs.google.com/classsf_1_1Transform.htm#a3e548c3c9e3fb9d4bd43cf852669e555)(45);

Parameters

| factors | Scaling factors |
| --- | --- |

ReturnsReference to \*this See Also[translate](http://docs.google.com/classsf_1_1Transform.htm#ab54f6c8070cc05e2afcb3145fbf4395a), [rotate](http://docs.google.com/classsf_1_1Transform.htm#a3e548c3c9e3fb9d4bd43cf852669e555)

| [Transform](http://docs.google.com/classsf_1_1Transform.htm)& sf::Transform::scale | ( | const [Vector2f](http://docs.google.com/classsf_1_1Vector2.htm) & | *factors*, |
| --- | --- | --- | --- |
|  |  | const [Vector2f](http://docs.google.com/classsf_1_1Vector2.htm) & | *center* |
|  | ) |  |  |

Combine the current transform with a scaling.

The center of scaling is provided for convenience as a second argument, so that you can build scaling around arbitrary points more easily (and efficiently) than the usual translate(-center).scale(factors).translate(center).

This function returns a reference to \*this, so that calls can be chained.

[sf::Transform](http://docs.google.com/classsf_1_1Transform.htm) transform;

transform.[scale](http://docs.google.com/classsf_1_1Transform.htm#a3f46af807f69d74120fb836334268671)([sf::Vector2f](http://docs.google.com/classsf_1_1Vector2.htm)(2, 1), [sf::Vector2f](http://docs.google.com/classsf_1_1Vector2.htm)(8, 3)).[rotate](http://docs.google.com/classsf_1_1Transform.htm#a3e548c3c9e3fb9d4bd43cf852669e555)(45);

Parameters

| factors | Scaling factors |
| --- | --- |
| center | Center of scaling |

ReturnsReference to \*this See Also[translate](http://docs.google.com/classsf_1_1Transform.htm#ab54f6c8070cc05e2afcb3145fbf4395a), [rotate](http://docs.google.com/classsf_1_1Transform.htm#a3e548c3c9e3fb9d4bd43cf852669e555)

| [Vector2f](http://docs.google.com/classsf_1_1Vector2.htm) sf::Transform::transformPoint | ( | float | *x*, |
| --- | --- | --- | --- |
|  |  | float | *y* |
|  | ) |  | const |

[Transform](http://docs.google.com/classsf_1_1Transform.htm) a 2D point.

Parameters

| x | X coordinate of the point to transform |
| --- | --- |
| y | Y coordinate of the point to transform |

ReturnsTransformed point

| [Vector2f](http://docs.google.com/classsf_1_1Vector2.htm) sf::Transform::transformPoint | ( | const [Vector2f](http://docs.google.com/classsf_1_1Vector2.htm) & | *point* | ) | const |
| --- | --- | --- | --- | --- | --- |

[Transform](http://docs.google.com/classsf_1_1Transform.htm) a 2D point.

Parameters

| point | Point to transform |
| --- | --- |

ReturnsTransformed point

| [FloatRect](http://docs.google.com/classsf_1_1Rect.htm) sf::Transform::transformRect | ( | const [FloatRect](http://docs.google.com/classsf_1_1Rect.htm) & | *rectangle* | ) | const |
| --- | --- | --- | --- | --- | --- |

[Transform](http://docs.google.com/classsf_1_1Transform.htm) a rectangle.

Since SFML doesn't provide support for oriented rectangles, the result of this function is always an axis-aligned rectangle. Which means that if the transform contains a rotation, the bounding rectangle of the transformed rectangle is returned.

Parameters

| rectangle | Rectangle to transform |
| --- | --- |

ReturnsTransformed rectangle

| [Transform](http://docs.google.com/classsf_1_1Transform.htm)& sf::Transform::translate | ( | float | *x*, |
| --- | --- | --- | --- |
|  |  | float | *y* |
|  | ) |  |  |

Combine the current transform with a translation.

This function returns a reference to \*this, so that calls can be chained.

[sf::Transform](http://docs.google.com/classsf_1_1Transform.htm) transform;

transform.[translate](http://docs.google.com/classsf_1_1Transform.htm#ab54f6c8070cc05e2afcb3145fbf4395a)(100, 200).[rotate](http://docs.google.com/classsf_1_1Transform.htm#a3e548c3c9e3fb9d4bd43cf852669e555)(45);

Parameters

| x | Offset to apply on X axis |
| --- | --- |
| y | Offset to apply on Y axis |

ReturnsReference to \*this See Also[rotate](http://docs.google.com/classsf_1_1Transform.htm#a3e548c3c9e3fb9d4bd43cf852669e555), [scale](http://docs.google.com/classsf_1_1Transform.htm#a3f46af807f69d74120fb836334268671)

| [Transform](http://docs.google.com/classsf_1_1Transform.htm)& sf::Transform::translate | ( | const [Vector2f](http://docs.google.com/classsf_1_1Vector2.htm) & | *offset* | ) |  |
| --- | --- | --- | --- | --- | --- |

Combine the current transform with a translation.

This function returns a reference to \*this, so that calls can be chained.

[sf::Transform](http://docs.google.com/classsf_1_1Transform.htm) transform;

transform.[translate](http://docs.google.com/classsf_1_1Transform.htm#ab54f6c8070cc05e2afcb3145fbf4395a)([sf::Vector2f](http://docs.google.com/classsf_1_1Vector2.htm)(100, 200)).[rotate](http://docs.google.com/classsf_1_1Transform.htm#a3e548c3c9e3fb9d4bd43cf852669e555)(45);

Parameters

| offset | Translation offset to apply |
| --- | --- |

ReturnsReference to \*this See Also[rotate](http://docs.google.com/classsf_1_1Transform.htm#a3e548c3c9e3fb9d4bd43cf852669e555), [scale](http://docs.google.com/classsf_1_1Transform.htm#a3f46af807f69d74120fb836334268671)

## Friends And Related Function Documentation

| | [Transform](http://docs.google.com/classsf_1_1Transform.htm) operator\* | ( | const [Transform](http://docs.google.com/classsf_1_1Transform.htm) & | *left*, | | --- | --- | --- | --- | |  |  | const [Transform](http://docs.google.com/classsf_1_1Transform.htm) & | *right* | |  | ) |  |  | | related |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

Overload of binary operator \* to combine two transforms.

This call is equivalent to calling Transform(left).combine(right).

Parameters

| left | Left operand (the first transform) |
| --- | --- |
| right | Right operand (the second transform) |

ReturnsNew combined transform

| | [Vector2f](http://docs.google.com/classsf_1_1Vector2.htm) operator\* | ( | const [Transform](http://docs.google.com/classsf_1_1Transform.htm) & | *left*, | | --- | --- | --- | --- | |  |  | const [Vector2f](http://docs.google.com/classsf_1_1Vector2.htm) & | *right* | |  | ) |  |  | | related |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

Overload of binary operator \* to transform a point.

This call is equivalent to calling left.transformPoint(right).

Parameters

| left | Left operand (the transform) |
| --- | --- |
| right | Right operand (the point to transform) |

ReturnsNew transformed point

| | [Transform](http://docs.google.com/classsf_1_1Transform.htm) & operator\*= | ( | [Transform](http://docs.google.com/classsf_1_1Transform.htm) & | *left*, | | --- | --- | --- | --- | |  |  | const [Transform](http://docs.google.com/classsf_1_1Transform.htm) & | *right* | |  | ) |  |  | | related |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

Overload of binary operator \*= to combine two transforms.

This call is equivalent to calling left.combine(right).

Parameters

| left | Left operand (the first transform) |
| --- | --- |
| right | Right operand (the second transform) |

ReturnsThe combined transform

## Member Data Documentation

| | const [Transform](http://docs.google.com/classsf_1_1Transform.htm) sf::Transform::Identity | | --- | | static |
| --- | --- | --- |

The identity transform (does nothing)

Definition at line [354](http://docs.google.com/Transform_8hpp_source.htm#l00354) of file [Transform.hpp](http://docs.google.com/Transform_8hpp_source.htm).

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

* [Transform.hpp](http://docs.google.com/Transform_8hpp_source.htm)

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