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**
* [Shape](http://docs.google.com/classsf_1_1Shape.htm)

[Public Member Functions](#_gjdgxs) | [Protected Member Functions](#_30j0zll) | [List of all members](http://docs.google.com/classsf_1_1Shape-members.htm)

sf::Shape Class Referenceabstract

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

Base class for textured shapes with outline. [More...](http://docs.google.com/classsf_1_1Shape.htm#details)

#include <[Shape.hpp](http://docs.google.com/Shape_8hpp_source.htm)>

Inheritance diagram for sf::Shape:



| Public Member Functions | |
| --- | --- |
| virtual | [~Shape](http://docs.google.com/classsf_1_1Shape.htm#a2262aceb9df52d4275c19633592f19bf) () |
|  | Virtual destructor. |
|  | |
| void | [setTexture](http://docs.google.com/classsf_1_1Shape.htm#af8fb22bab1956325be5d62282711e3b6) (const [Texture](http://docs.google.com/classsf_1_1Texture.htm) \*texture, bool resetRect=false) |
|  | Change the source texture of the shape. |
|  | |
| void | [setTextureRect](http://docs.google.com/classsf_1_1Shape.htm#a2029cc820d1740d14ac794b82525e157) (const [IntRect](http://docs.google.com/classsf_1_1Rect.htm) &rect) |
|  | Set the sub-rectangle of the texture that the shape will display. |
|  | |
| void | [setFillColor](http://docs.google.com/classsf_1_1Shape.htm#a3506f9b5d916fec14d583d16f23c2485) (const [Color](http://docs.google.com/classsf_1_1Color.htm) &color) |
|  | Set the fill color of the shape. |
|  | |
| void | [setOutlineColor](http://docs.google.com/classsf_1_1Shape.htm#a5978f41ee349ac3c52942996dcb184f7) (const [Color](http://docs.google.com/classsf_1_1Color.htm) &color) |
|  | Set the outline color of the shape. |
|  | |
| void | [setOutlineThickness](http://docs.google.com/classsf_1_1Shape.htm#a5ad336ad74fc1f567fce3b7e44cf87dc) (float thickness) |
|  | Set the thickness of the shape's outline. |
|  | |
| const [Texture](http://docs.google.com/classsf_1_1Texture.htm) \* | [getTexture](http://docs.google.com/classsf_1_1Shape.htm#a1bf27ac425fcce36efd0eed67531a403) () const |
|  | Get the source texture of the shape. |
|  | |
| const [IntRect](http://docs.google.com/classsf_1_1Rect.htm) & | [getTextureRect](http://docs.google.com/classsf_1_1Shape.htm#af7c4c80a435b85a622812711cf510439) () const |
|  | Get the sub-rectangle of the texture displayed by the shape. |
|  | |
| const [Color](http://docs.google.com/classsf_1_1Color.htm) & | [getFillColor](http://docs.google.com/classsf_1_1Shape.htm#ad7f7fe601a8bb24efe9aa77809a35c12) () const |
|  | Get the fill color of the shape. |
|  | |
| const [Color](http://docs.google.com/classsf_1_1Color.htm) & | [getOutlineColor](http://docs.google.com/classsf_1_1Shape.htm#a4fa7d3bf5ee2332f6b9d9bebe9b1e2fd) () const |
|  | Get the outline color of the shape. |
|  | |
| float | [getOutlineThickness](http://docs.google.com/classsf_1_1Shape.htm#ac66f917b41eda6159a8ba6571d77f2ad) () const |
|  | Get the outline thickness of the shape. |
|  | |
| virtual unsigned int | [getPointCount](http://docs.google.com/classsf_1_1Shape.htm#ad84e1b675ecd270ad8151aea4e271a78) () const =0 |
|  | Get the total number of points of the shape. |
|  | |
| virtual [Vector2f](http://docs.google.com/classsf_1_1Vector2.htm) | [getPoint](http://docs.google.com/classsf_1_1Shape.htm#a397f3b4cdb7ad98cdc6c034816c652d2) (unsigned int index) const =0 |
|  | Get a point of the shape. |
|  | |
| [FloatRect](http://docs.google.com/classsf_1_1Rect.htm) | [getLocalBounds](http://docs.google.com/classsf_1_1Shape.htm#a5d26a18ccfe850ff8d327ca97edbc34a) () 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_1Shape.htm#a5257341fe832884dbba6b9dc855e33cc) () 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 |
|  | |

| Protected Member Functions | |
| --- | --- |
|  | [Shape](http://docs.google.com/classsf_1_1Shape.htm#a413a457f720835b9f5d8e97ca8b80960) () |
|  | Default constructor. |
|  | |
| void | [update](http://docs.google.com/classsf_1_1Shape.htm#adfb2bd966c8edbc5d6c92ebc375e4ac1) () |
|  | Recompute the internal geometry of the shape. |
|  | |

## Detailed Description

Base class for textured shapes with outline.

[sf::Shape](http://docs.google.com/classsf_1_1Shape.htm) is a drawable class that allows to define and display a custom convex shape on a render target.

It's only an abstract base, it needs to be specialized for concrete types of shapes (circle, rectangle, convex polygon, star, ...).

In addition to the attributes provided by the specialized shape classes, a shape always has the following attributes:

* a texture
* a texture rectangle
* a fill color
* an outline color
* an outline thickness

Each feature is optional, and can be disabled easily:

* the texture can be null
* the fill/outline colors can be [sf::Color::Transparent](http://docs.google.com/classsf_1_1Color.htm#a569b45471737f770656f50ae7bbac292)
* the outline thickness can be zero

You can write your own derived shape class, there are only two virtual functions to override:

* getPointCount must return the number of points of the shape
* getPoint must return the points of the shape

See Also[sf::RectangleShape](http://docs.google.com/classsf_1_1RectangleShape.htm), [sf::CircleShape](http://docs.google.com/classsf_1_1CircleShape.htm), [sf::ConvexShape](http://docs.google.com/classsf_1_1ConvexShape.htm), [sf::Transformable](http://docs.google.com/classsf_1_1Transformable.htm)

Definition at line [44](http://docs.google.com/Shape_8hpp_source.htm#l00044) of file [Shape.hpp](http://docs.google.com/Shape_8hpp_source.htm).

## Constructor & Destructor Documentation

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

Virtual destructor.

| | sf::Shape::Shape | ( |  | ) |  | | --- | --- | --- | --- | --- | | protected |
| --- | --- | --- | --- | --- | --- | --- |

Default constructor.

## Member Function Documentation

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

Get the fill color of the shape.

ReturnsFill color of the shape See Also[setFillColor](http://docs.google.com/classsf_1_1Shape.htm#a3506f9b5d916fec14d583d16f23c2485)

| [FloatRect](http://docs.google.com/classsf_1_1Rect.htm) sf::Shape::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::Shape::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 [Color](http://docs.google.com/classsf_1_1Color.htm)& sf::Shape::getOutlineColor | ( |  | ) | const |
| --- | --- | --- | --- | --- |

Get the outline color of the shape.

ReturnsOutline color of the shape See Also[setOutlineColor](http://docs.google.com/classsf_1_1Shape.htm#a5978f41ee349ac3c52942996dcb184f7)

| float sf::Shape::getOutlineThickness | ( |  | ) | const |
| --- | --- | --- | --- | --- |

Get the outline thickness of the shape.

ReturnsOutline thickness of the shape See Also[setOutlineThickness](http://docs.google.com/classsf_1_1Shape.htm#a5ad336ad74fc1f567fce3b7e44cf87dc)

| | virtual [Vector2f](http://docs.google.com/classsf_1_1Vector2.htm) sf::Shape::getPoint | ( | unsigned int | *index* | ) | const | | --- | --- | --- | --- | --- | --- | | pure virtual |
| --- | --- | --- | --- | --- | --- | --- | --- |

Get a point of the shape.

The result is undefined if *index* is out of the valid range.

Parameters

| index | Index of the point to get, in range [0 .. [getPointCount()](http://docs.google.com/classsf_1_1Shape.htm#ad84e1b675ecd270ad8151aea4e271a78) - 1] |
| --- | --- |

ReturnsIndex-th point of the shape See Also[getPointCount](http://docs.google.com/classsf_1_1Shape.htm#ad84e1b675ecd270ad8151aea4e271a78)

Implemented in [sf::ConvexShape](http://docs.google.com/classsf_1_1ConvexShape.htm#ae2a18b837cd4454e340599a220c09a34), [sf::CircleShape](http://docs.google.com/classsf_1_1CircleShape.htm#a05139deaef220ed3d5a3bc4ca9aa9dbe), and [sf::RectangleShape](http://docs.google.com/classsf_1_1RectangleShape.htm#a3994f7f937d6332fe64b6990d5bc43a1).

| | virtual unsigned int sf::Shape::getPointCount | ( |  | ) | const | | --- | --- | --- | --- | --- | | pure virtual |
| --- | --- | --- | --- | --- | --- | --- |

Get the total number of points of the shape.

ReturnsNumber of points of the shape See Also[getPoint](http://docs.google.com/classsf_1_1Shape.htm#a397f3b4cdb7ad98cdc6c034816c652d2)

Implemented in [sf::CircleShape](http://docs.google.com/classsf_1_1CircleShape.htm#ae41ed830ca8f459e88ea6f125c240949), [sf::RectangleShape](http://docs.google.com/classsf_1_1RectangleShape.htm#a439f5a92583baf972878c836b73bf955), and [sf::ConvexShape](http://docs.google.com/classsf_1_1ConvexShape.htm#af81b86134fe54f2d50d9fab0db065ef1).

| | 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::Shape::getTexture | ( |  | ) | const |
| --- | --- | --- | --- | --- |

Get the source texture of the shape.

If the shape 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 shape's texture See Also[setTexture](http://docs.google.com/classsf_1_1Shape.htm#af8fb22bab1956325be5d62282711e3b6)

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

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

Returns[Texture](http://docs.google.com/classsf_1_1Texture.htm) rectangle of the shape See Also[setTextureRect](http://docs.google.com/classsf_1_1Shape.htm#a2029cc820d1740d14ac794b82525e157)

| | 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::Shape::setFillColor | ( | const [Color](http://docs.google.com/classsf_1_1Color.htm) & | *color* | ) |  |
| --- | --- | --- | --- | --- | --- |

Set the fill color of the shape.

This color is modulated (multiplied) with the shape's texture if any. It can be used to colorize the shape, or change its global opacity. You can use [sf::Color::Transparent](http://docs.google.com/classsf_1_1Color.htm#a569b45471737f770656f50ae7bbac292) to make the inside of the shape transparent, and have the outline alone. By default, the shape's fill color is opaque white.

Parameters

| color | New color of the shape |
| --- | --- |

See Also[getFillColor](http://docs.google.com/classsf_1_1Shape.htm#ad7f7fe601a8bb24efe9aa77809a35c12), [setOutlineColor](http://docs.google.com/classsf_1_1Shape.htm#a5978f41ee349ac3c52942996dcb184f7)

| | 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::Shape::setOutlineColor | ( | const [Color](http://docs.google.com/classsf_1_1Color.htm) & | *color* | ) |  |
| --- | --- | --- | --- | --- | --- |

Set the outline color of the shape.

By default, the shape's outline color is opaque white.

Parameters

| color | New outline color of the shape |
| --- | --- |

See Also[getOutlineColor](http://docs.google.com/classsf_1_1Shape.htm#a4fa7d3bf5ee2332f6b9d9bebe9b1e2fd), [setFillColor](http://docs.google.com/classsf_1_1Shape.htm#a3506f9b5d916fec14d583d16f23c2485)

| void sf::Shape::setOutlineThickness | ( | float | *thickness* | ) |  |
| --- | --- | --- | --- | --- | --- |

Set the thickness of the shape's outline.

Note that negative values are allowed (so that the outline expands towards the center of the shape), and using zero disables the outline. By default, the outline thickness is 0.

Parameters

| thickness | New outline thickness |
| --- | --- |

See Also[getOutlineThickness](http://docs.google.com/classsf_1_1Shape.htm#ac66f917b41eda6159a8ba6571d77f2ad)

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

Change the source texture of the shape.

The *texture* argument refers to a texture that must exist as long as the shape uses it. Indeed, the shape 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 shape tries to use it, the behaviour is undefined. *texture* can be NULL to disable texturing. If *resetRect* is true, the TextureRect property of the shape 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_1Shape.htm#a1bf27ac425fcce36efd0eed67531a403), [setTextureRect](http://docs.google.com/classsf_1_1Shape.htm#a2029cc820d1740d14ac794b82525e157)

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

Set the sub-rectangle of the texture that the shape 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

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

See Also[getTextureRect](http://docs.google.com/classsf_1_1Shape.htm#af7c4c80a435b85a622812711cf510439), [setTexture](http://docs.google.com/classsf_1_1Shape.htm#af8fb22bab1956325be5d62282711e3b6)

| | void sf::Shape::update | ( |  | ) |  | | --- | --- | --- | --- | --- | | protected |
| --- | --- | --- | --- | --- | --- | --- |

Recompute the internal geometry of the shape.

This function must be called by the derived class everytime the shape's points change (ie. the result of either getPointCount or getPoint is different).

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

* [Shape.hpp](http://docs.google.com/Shape_8hpp_source.htm)

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