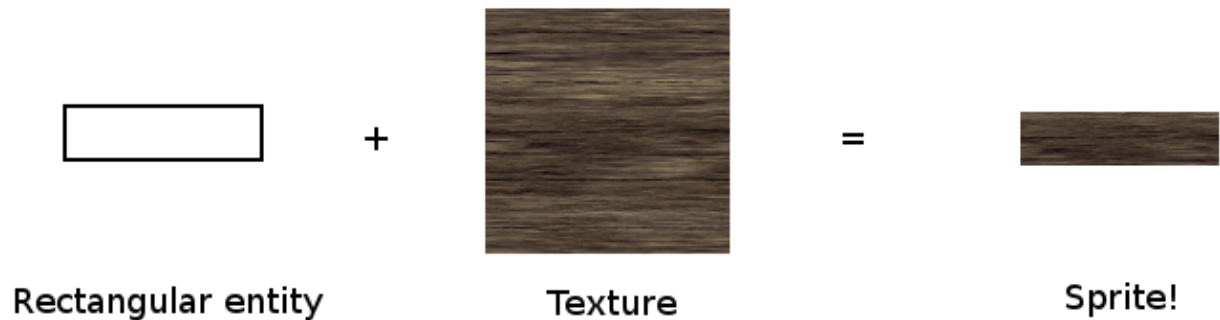


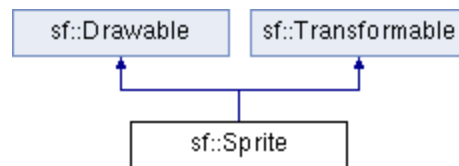
# Sprites

In this tutorial we will see how to create and draw a sprite and much more.

The object that's formed by mapping the texture into a object is known as sprite.



## Inheritance diagram of sf::Sprite



## Create and Draw Sprite

```
// create
sf::Sprite sprite;
sprite.setTexture(texture);
```

```
// Draw
// inside the main loop, between window.clear() and window.display()
window.draw(sprite);
```

## Use the specific portion of the texture into a sprite

```
// first two arguments are the coordinates and last two are the dimensions of rectangle
sprite.setTextureRect(sf::IntRect(10, 10, 32, 32));
```

---

## Change the color of sprite

You can also change the color of a sprite. The color that you set is modulated (multiplied) with the texture of the sprite. This can also be used to change the global transparency (alpha) of the sprite.

```
sprite.setColor(sf::Color(0, 255, 0)); // green
sprite.setColor(sf::Color(255, 255, 255, 128)); // half transparent
```

## Transformed a sprite

Sprites can also be transformed: They have a position, an orientation and a scale.

```
// position
sprite.setPosition(sf::Vector2f(10.f, 50.f)); // absolute position
sprite.move(sf::Vector2f(5.f, 10.f)); // offset relative to the current position

// rotation
sprite.setRotation(90.f); // absolute angle
sprite.rotate(15.f); // offset relative to the current angle

// scale
sprite.setScale(sf::Vector2f(0.5f, 2.f)); // absolute scale factor
sprite.scale(sf::Vector2f(1.5f, 3.f)); // factor relative to the current scale
```

By default, the origin for these three transformations is the top-left corner of the sprite. If you want to set the origin to a different point (for example the center of the sprite, or another corner), you can use the `setOrigin` function.

```
sprite.setOrigin(sf::Vector2f(25.f, 25.f));
```

## The white square problem

You successfully loaded a texture, constructed a sprite correctly, and... all you see on your screen now is a white square. What happened ?

This is a common mistake. When you set the texture of a sprite, all it does internally is store a **pointer** to the texture instance. Therefore, if the texture is destroyed or moves elsewhere in memory, the sprite ends up with an invalid texture pointer.

This problem occurs when you write this kind of function :

```
// white_square_prob.hpp

#include <SFML/Graphics.hpp>

sf::Sprite loadSprite(std::string filename);

void whiteSquareProblem()
{
    // create window
    sf::RenderWindow window(sf::VideoMode(800, 600), "window");
    window.setFramerateLimit(60);

    // load the sprite
    sf::Sprite sprite;
    sprite = loadSprite("../asset\\images\\car.png"); // pass the path as std::string

    while (window.isOpen())
    {
        sf::Event event;
        // handle the events
        while (window.pollEvent(event))
        {
            if (event.type == sf::Event::Closed)
                window.close();
        }

        // update the frame

        // render the frame
        window.clear();
        window.draw(sprite);
        window.display();
    }
}

// function that will create the problem
sf::Sprite loadSprite(std::string filename)
{
    sf::Texture texture;
    texture.loadFromFile(filename);

    return sf::Sprite(texture);
}
```

```
// main.cpp

#include "white_square_prob.hpp"

int main () {
    whiteSquareProblem();
    return 0;
}
```



## The importance of using as few textures as possible


Using as few textures as possible is a good strategy, and the reason is simple: Changing the current texture is an expensive operation for the graphics card. Drawing many sprites that use the same texture will yield the best performance.

you can only use one texture per `draw` call), which will be much faster to draw than a set of many entities.

**To know more about functions related to sprite then follow the below link : )**

**sf::Sprite Class Reference (SFML / Learn / 2.5.1 Documentation)**

Drawable representation of a texture, with its own transformations, color, etc. More... Drawable representation of a texture, with its own transformations, color, etc. sf::Sprite is a drawable class that allows to

 [https://www.sfml-dev.org/documentation/2.5.1/classsf\\_1\\_1Sprite.php#details](https://www.sfml-dev.org/documentation/2.5.1/classsf_1_1Sprite.php#details)



