Drawing 2D Stuff

SFML provides a graphics module which will help you draw 2D entities in a much simpler way

The Drawing Window

To draw entities for example- shapes provided by the graphic module you must use a specialized window class `sf::RenderWindow`. This class is derived from `sf::Window`, and inherits all its functions. Everything that you've learnt about `sf::Window` (creation, event handling, controlling the framerate, mixing with OpenGL, etc.) is applicable to `sf::RenderWindow` as well.

On top of that, `sf::RenderWindow` adds high-level functions to help you draw things easily. In this tutorial we'll focus on two of these functions: `clear` and `draw`. They are as simple as their name implies: `clear` clears the whole window with the chosen color, and `draw` draws whatever object you pass to it.

Here is what a typical main loop looks like with a render window :

```
#include <SFML/Graphics.hpp>

int main()
{
    // create the window
    sf::RenderWindow window(sf::VideoMode(800, 600), "My window");

    // run the program as long as the window is open
    while (window.isOpen())
    {
        // check all the window's events that were triggered since the last iteration of the loop
        sf::Event event;
        while (window.pollEvent(event))
        {
            // "close requested" event: we close the window
            if (event.type == sf::Event::Closed)
                window.close();
        }

        // clear the window with black color
        window.clear(sf::Color::Black);

        // draw everything here...
        // window.draw(...);

        // end the current frame
        window.display();
    }

    return 0;
}
```

- Calling `clear` before drawing anything is mandatory, otherwise the contents from previous frames will be present behind anything you draw.
- Calling `display` is also mandatory, it takes what was drawn since the last call to `display` and displays it on the window.

What can I draw now ?

Now that you have a main loop which is ready to draw, let's see what, and how, you can actually draw there.

SFML provides four kinds of drawable entities: three of them are ready to be used (**sprites, text and shapes**), the last one is the building block that will help you create your own drawable entities **(vertex arrays**).

Although they share some common properties, each of these entities come with their own nuances and are therefore explained in dedicated tutorials :

- Sprite tutorial
- Text tutorial
- Shape tutorial
- Vertex array tutorial

Drawing From Threads

SFML suports multi-threaded drawing and you don’t have to do anything to make it work. The only point is to remember is to deactivate a window using it in another thread. That's because a window (more precisely its OpenGL context) cannot be active in more than one thread at the same time.

```
void renderingThread(sf::RenderWindow* window)
{
    // activate the window's context
    window->setActive(true);

    // the rendering loop
    while (window->isOpen())
    {
        // draw...

        // end the current frame
        window->display();
    }
}

int main()
```

```
{
    // create the window (remember: it's safer to create it in the main thread due to OS limitations)
    sf::RenderWindow window(sf::VideoMode(800, 600), "OpenGL");

    // deactivate its OpenGL context
    window.setActive(false);

    // launch the rendering thread
    sf::Thread thread(&renderingThread, &window);
    thread.launch();

    // the event/logic/whatever loop
    while (window.isOpen())
    {
        ...
    }

    return 0;
}
```

As you can see, you don't even need to bother with the activation of the window in the rendering thread, SFML does it automatically for you whenever it needs to be done.

Remember to always create the window and handle its events in the main thread for maximum portability.