Creating a functioning window

In the previous section we create a window but the window in not a functional window, Now to create a fully functional window we need to add some more functionality on top of the previous code.

Now, to add more functionality we need to first create a proper structure of the code and this is important because without proper structure you won’t be able to create a functioning window and this will be your standard structure to any application that you develop with the help of SFML.

Creating a Standard Structure

Constructing a structure for a functioning window will give you an idea of your code flow like- setup, game loop, and cleaning frame.

The three main parts of the standard structure

- Setup
- Game Loop (Application loop)
- Clearup

Demonstration of standard structure

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

// The entry point into our program
int main()
{

// *Setup

//  # Creating a window(800x600) because we are create a graphical program
    sf::RenderWindow window(sf::VideoMode(800, 600), "SFML Window", sf::Style::Default);

// *Game Loop

// # Our main game loop is here and we wanted to render the window on a dekstop until the user close
//    the window.
// # This loop is a infinite loop and it gets terminate when user close the window explicitly.

    while (window.isOpen())
    {
        // Game loop stages
        // 1. handle the events
        // 2. update the frame
        // 3. render the frame

        // 1. handle the events
        sf:: Event event;
        // so if any events happened, they are pushed into a queue and then we can handle them by
        // sf::Event::pollEvent(sf::Event& event) function.
        while (window.pollEvent(event))
        {
            // here is the sample event that can be tracked
            if (event.type == sf::Event::EventType::Closed) // terminating condition
                window.close();
        }

        // 2. update frame
            /*
            .
            .   update the scenes, objects etc of your game.
                functions that comes under render frame sections are- setPostion(), move() etc.
            .
            .
            */

//  *Clear

//   #  In a window typically we want it to refresh and draw a new 'frame' every time this loop
//      executes. To do so we first clear the content of the window and this can be done by clear() function.

        // 3. render frame
            /*
            .
            .   display the scenes, objects etc of your game into a window.
            .   functions that comes under render frame sections are- draw(), display()
            .
            */
    }
//
    return 0;
}
```

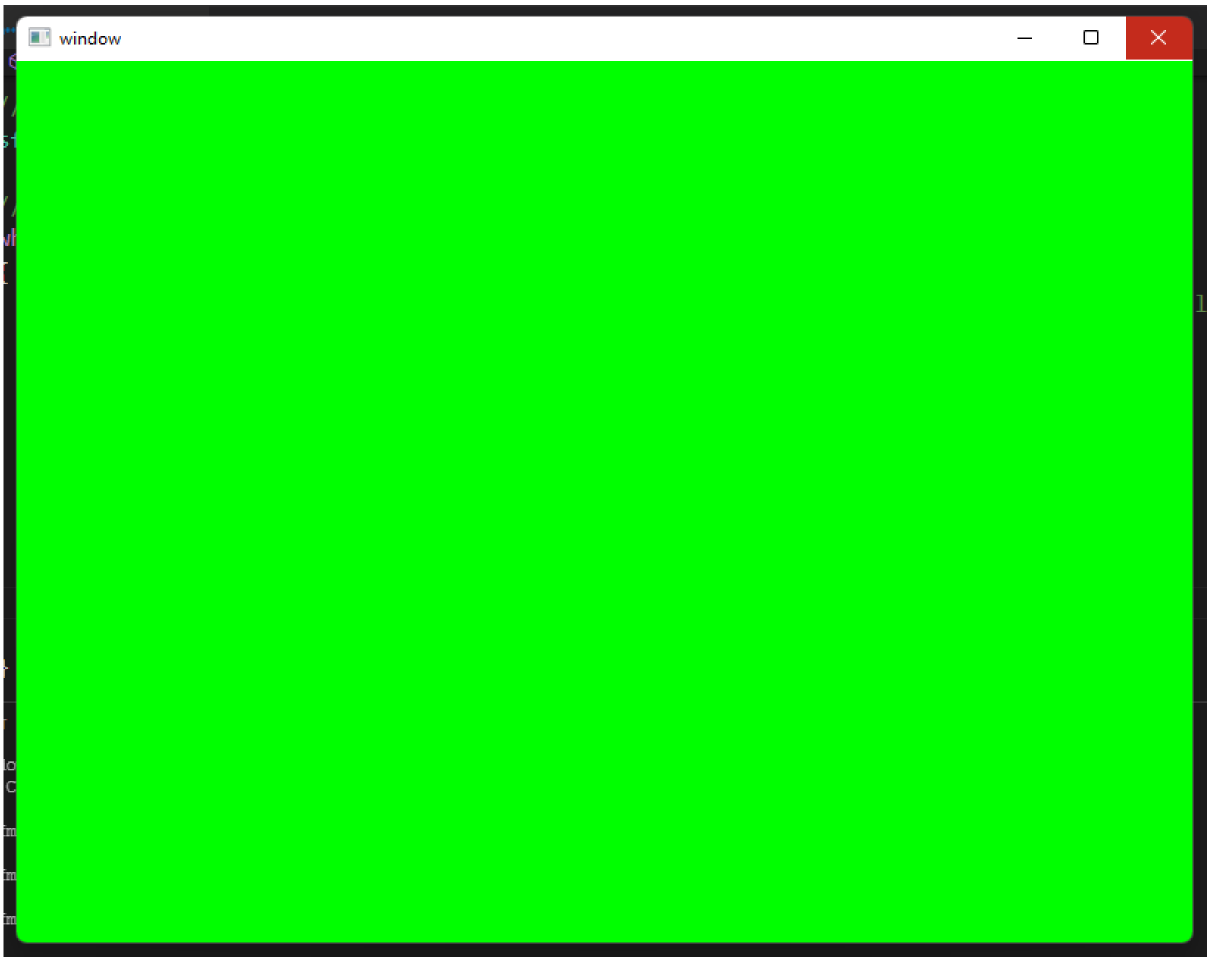
Final code of creating a functioning window

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

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

    // game loop
    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- set the window color to green in each frame
        window.clear(sf::Color::Green);
        // display the color green on a window
        window.display();
    }

    return 0;
}
```



Now after creating this window you will be able to resize, close, move the window and much more.

sf::Event::EventType::Closed what does it mean ?

- Here, **sf::Event** is class and it defines a enumeration, EventType and closed is the member of enumeration EventType.
- You can write **sf::Event::EventType::Closed** to this also **sf::Event::Closed** In this format you directly acessing the member of enumeration.