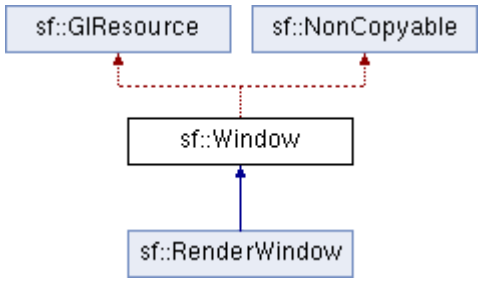
Creating Window

- In this tutorial we will see how to create window by the help of SFML and also see different ways to create a window by **sf::Window and sf::RenderWindow class.**

sf Namespace

Every class in SFML is under this namespace, which separates all the classes in SFML from the classes in other libraries.

Take a look on a Inheritance diagram



sf::GlResource Class

- Base class for classes that require an OpenGL context.
- This class is for internal use only, it must be the base of every class that requires a valid OpenGL context in order to work.

sf::NonCopyable Class

- Utility class that makes any derived class non-copyable.
- This class makes its instances non-copyable, by explicitly disabling its copy constructor and its assignment operator.
- To create a non-copyable class, simply inherit from **sf::NonCopyable**.
- The type of inheritance (public or private) doesn't matter, the copy constructor and assignment operator are declared private in **sf::NonCopyable** so they will end up being inaccessible in both cases. Thus you can use a shorter syntax for inheriting from it (see below).

```
class MyNonCopyableClass : sf::NonCopyable
{
    ...
};

// If any class that inherits the MMyNonCopyableClass will not able to inherit the
// common property of MMyNonCopyableClass
```

sf::Window Class

What is Window Class ?

- Window class is dervied from both **sf::GlResource Class** and **sf::NonCopyable Class**
- Window class Is used to create a window by passing the arrguments into a constructor or by calling create() function, It take 4 arrguments the first two arrguments are necessary and last two are optional.
- It defines under Window.hpp header file.
- It defines an OS window that is able to receive an OpenGL rendering.
- The **sf::Window** class provides a simple interface for manipulating the window: move, resize, show/hide, control mouse cursor, etc. It also provides event handling through its pollEvent()
  and waitEvent() functions.

About Window Class Arrguments

- First arrgument- It specify the dimensions of window, you can get the dimensions or mode of window by the help of VideoMode class (In the next tutorial we will see much more about Videomode class).
- Second arrgument- It is use to add the tittle to your window
- Third arrgument- It is use to provide style to your window like closing, opening, minimizing, resizing window.
- Fourth arrgument- ContexSettings (ConntextSettings is now, off to the topic we will see them in future tutorial).

Create a window by passing arrguments into constructor

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

int main()
{
    // passing the arrguments into a constructor
    sf::Window window(sf::VideoMode(800, 600), "My window", sf::Style::Default);
    sf::sleep(sf::seconds(30));

    return 0;
}
```

Create a window by calling create function

- If Window::create() is called on a window, which is already open, it closes the window and reinitializes it with the new set of parameters.

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

int main()
{

    sf::Window window;
    // calling the create function
    window.create(sf::VideoMode(800, 600), "SFML Window", sf::Style::Default);
    sf::sleep(sf::seconds(30));

    return 0;
}
```

sf::RenderWindow class

- **sf::Window class** doesn't provide an interface to draw SFML shapes.
- **sf::RenderWindow** is the main class of the Graphics module.
- **sf::RenderWindow class** is derived from **sf::Window class**, thus it inherits all its features: events, window management, OpenGL rendering, etc and also adds more functionality on top of it. So we can still create it, poll events, and so on, in the same way we do with the base class Window.

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

int main()
{
    sf::RenderWindow window(sf::VideoMode(800, 600), "SFML Window", sf::Style::Default);
    sf::sleep(sf::seconds(30));
```

```
    return 0;
}
```

- Below Image will show you the window(800x600) that's created with the help of above code.

Style arrgument of sf::Window class

- SFML provides some styles that you can implement to your window.

`sf::Style::None`	No decoration at all (useful for splash screens, for example); this style cannot be combined with others
`sf::Style::Titlebar`	The window has a titlebar
`sf::Style::Resize`	The window can be resized and has a maximize button
`sf::Style::Close`	The window has a close button
`sf::Style::Fullscreen`	The window is shown in fullscreen mode; this style cannot be combined with others, and requires a valid video mode
`sf::Style::Default`	The default style, which is a shortcut for `Titlebar | Resize | Close`

- You can create you own style by combining different style with the help of bitwise OR operator

```
// combining the Close and Resize style
sf::Uint32 style = sf::Style::Close | sf::Style::Resize;
// Now you have to only use small style as third arrgument
```

What's the purpose sleep( ) function

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

int main()
{
    sf::RenderWindow window(sf::VideoMode(800, 600), "SFML Window", sf::Style::Default);

    return 0;
}
```

In this code I haven't use this statement **sf::sleep(sf::seconds(30))**, Now when you run this code you won't see anything. offcourse you are creating the window but the window is not fully functional till now, As a result the program naturally exists.

Now to block the main function from existing the main function, we have to delay the window thread. SFML provides a simple interface for that; just add the **sf::sleep(sf::seconds(30))** line after the line which creates the window. Now, the window is clearly visible for the duration 30 sec. of sleep.