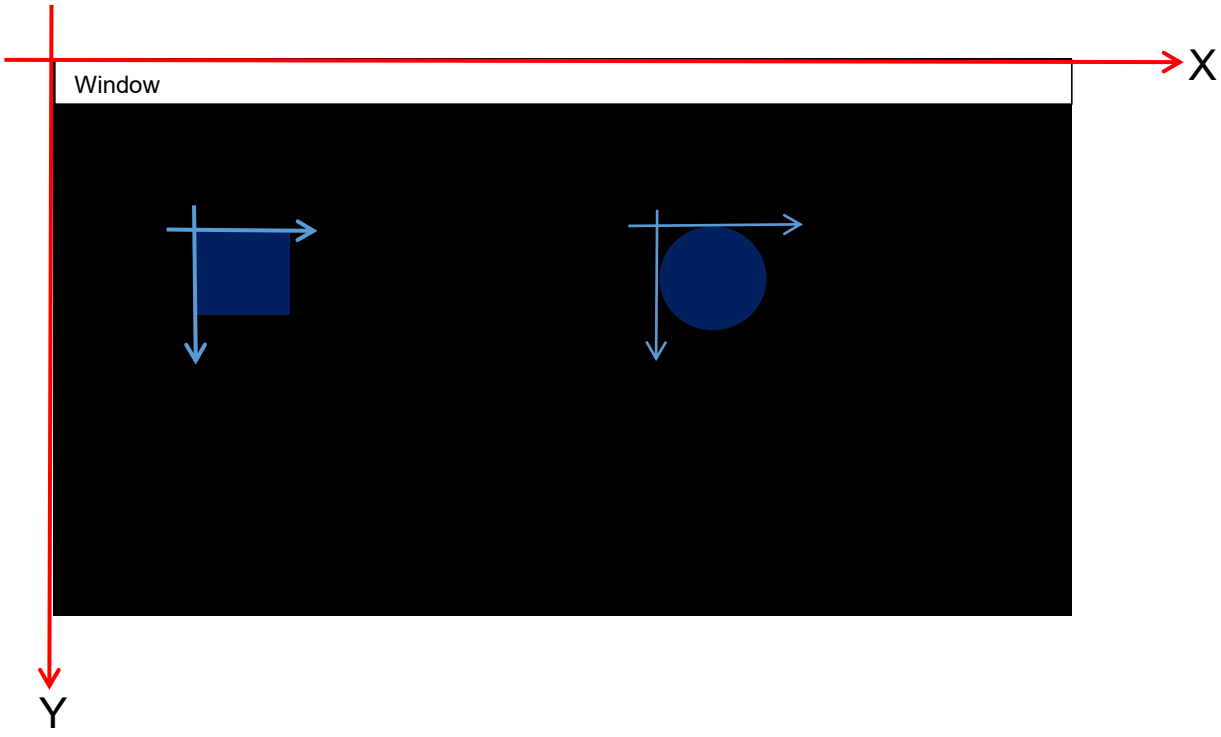
In SFML (and most graphics libraries), the window uses a **screen coordinate system** that's slightly different from the one we study in mathematics. By default, the **origin (0,0)** is at the **top-left corner** of the window.

- The **X-axis** increases to the **right**, just like in math.
- The **Y-axis**, however, increases **downward** toward the bottom of the window, which is the opposite of the usual Cartesian system.

Every shape you draw—whether it's a rectangle, circle, or custom object—also has its own **local coordinate system**, with its own origin, position, and transformations. Understanding how the global window coordinates interact with each shape's local coordinates is key to getting precise control over placement, alignment, and movement.



```
//To Add Rectangle
RectangleShape rectangle({ 120.f , 80.f });  //( 120.f is width and 80.f is height)
rectangle.setFillColor(Color::Blue);
rectangle.setOrigin({ 60.f , 40.f });
rectangle.setPosition({ 600.f , 400.f });
```

This line of code creates a rectangle with a **width of 120** and a **height of 80**. The *setFillColor()* function is then used to fill the rectangle with a chosen color.

To simplify calculations—especially when rotating or moving shapes—we often use the *setOrigin()* function. This lets us redefine the rectangle's origin point. By default, the origin is at the **top-left corner**, but we usually set it to the **center** of the rectangle. That way, transformations like rotation and positioning become much easier to manage.

Finally, the *setPosition()* function allows us to place the rectangle anywhere in the window, relative to its origin.

Ask Yourself

1) Where will the rectangle be in this case?
2) What will change in the display if I rewrite the code as:

```
RectangleShape rectangle({ 80.f , 120.f });
rectangle.setFillColor(Color::Red);
rectangle.setPosition({ 600.f , 400.f });
rectangle.setOrigin({ 60.f , 40.f });
```