The Render Frame

Render Cycle

The render cycle Is a routine that repeats in every frame, The main motive of render cycle Is to :

- Clear the gaming objects from the previous frame of a window.
- Draw the newly or updated objects of game for the current frame on a canvas.
- Display the newly or updated gaming objects on a window.

Note

For simplicity I'm calling buffer as canvas so that you can relate the scenario. One thing to note Is that canvas that we use for rendering the objects are double canvas It's very common way to render the objects and It's known as Double Buffering.

Double Buffering

The canvas that we are using for rendering the objects has two sides :

- The one that Is not shown on the screen.
- The one that Is shown on the screen.

Concept of flipping the canvases

- The concept behind Is that, when we draw all the game objects we are drawing them to a hidden canvas that is getting ready to be displayed and flip with the canvas that Is used as to display the game objects and this flipping process continues In each frame this technique Is also known as **swapping the buffers.**
- The flipping Is done by calling the **window.display( ).**
- This way, the player will never see the drawing process as the canvas has all the sprites added to it. It also guarantees that the scene will be complete before it is flipped. This prevents a graphical glitch known as **tearing.**

display( ) can make thread to sleep

Apart from that, the Window::display( ) method can put the to thread sleep for a calculated amount of time to achieve a target framerate (frames per second). We

can set the desired framerate by calling **Window::setFramerateLimit( )** once at the beginning of the program. The function doesn't guarantee the limiting of the framerate to the exact amount we pass it, but rather it makes a close approximation.

The recommended frame rate limit are 30 and 60 frames per second.

clear( )

It clears the canvas for redrawing. Notice that it takes a sf::Color argument, which is an RGBA representation of a color. We can initialize it manually by calling the constructor and passing each value individually, or we can use one of the preset colors. For example Color::Red, Color::Blue, Color::Magenta, or rgba values (we will see more about Color class and rgba concepts in future tutorials).