GLFWwindow* window; *#GIVING THE NAME OF A WINDOW THAT WE ARE GOING TO USE*

/ Initialize the library */*

```
if (!glfwInit())
      return -1;
```
➔ **CHECK THAT EVERYTHING IS FINE**

// Open a window and attach an OpenGL context to the window surface

```
        window = glfwCreateWindow(800, 600, "OpenGL 101", NULL, NULL);
        if (!window)
        {
                std::cerr << "Failed to open a window! I'm out!" << '\n';
                glfwTerminate();
                exit(-1);
        }
```
➔ **CHECK THAT EVERYTHING IS FINE**

// Set the window context current

```
        glfwMakeContextCurrent(window);
```
➔ **WE USE THIS WINDOW**

// Set the swap interval, 1 will use your screen refresh rate (vsync)

```
        glfwSwapInterval(1);
```
➔ **IT'S JUST TO TELL THE LIBRARY TO REFRESH AFTER 1 V-BLANK (NOT ESENTIAL)**

RENDERING:

Rendering or image synthesis is the automatic process of generating a photorealistic or non-photorealistic image **from a 2D or 3D model** (or models in what collectively could be called a scene file) by means of computer programs. Also, the results of displaying such a model can be called a **render**. A scene file contains objects in a strictly defined language or data structure; it would contain geometry, viewpoint, texture, lighting, and shading information as a description of the virtual scene. The data contained in the scene file is then passed to a rendering program to be processed and output to a digital image or raster graphics image file. The term "rendering" may be by analogy with an "artist's rendering" of a scene.

SWAP INTERVAL:

Swap Interval is a means of synchronizing the swapping of the front and back frame buffers with vertical blanks (v-blank): the hardware event where the screen image is updated with data from the front framebuffer. It is a very common means of preventing frame "tearing," (seeing half of one frame and half of another) as often seen in high-motion-content graphics. (*VBLANK is the time between the end of the final line of a frame or field and the beginning of the first line of the next frame*)

glfwSwapInterval(1); ➔ **PREVENT FRAME TEARING**

// Use red to clear the screen

```
        glClearColor(1, 0, 0, 1);
```

// Create a rendering loop that runs until the window is closed **VERY IMPORTANT**

```
        while (!glfwWindowShouldClose(window)) {
                // Clear the screen (window background)
                glClear(GL_COLOR_BUFFER_BIT);

                // Draw
                // ...

                // Swap front and back buffers for the current window
                glfwSwapBuffers(window);

                // Poll for events
                glfwPollEvents();
        }
```

```
// Destroy the window and its context
        glfwDestroyWindow(window);

        // Terminate GLFW
        glfwTerminate();
        return 0;
}
```