Lab 1.2
ImGui, Shaders, and Uniform Variables

1. Look at the keyboard() callback in the C++ template. Specifically, observe that the shader is reloaded when the 'r' key is pressed. Now look at the draw_gui() function. Add an imgui button that reloads the shader when it is pressed.

    - Test the button you just created by changing the fragment shader (template_fs.glsl) to display the mesh in red by changing the body of main() to `fragcolor = vec4(1.0, 0.0, 0.0, 1.0);`
        - I like to add a new filter in my project (right-click -> add filter) named shaders, and add my shader files to it so that I can easily open them in VS 2017.

      Save the shader, then press the button to see the effect of the new shader.

    - Now see if you can change the shader to display the texture coordinates as red and green color components. Changing the fragment color is a useful way of debugging since there is no printf/cout in glsl.

    - Change the shader back to the original functionality of displaying the texture color.

2. Look at the idle() callback in the C++ template. Modify the function to pass time_sec to the shader as a uniform variable. See Example 2.2 on page 48 of the textbook for an example. Add a corresponding variable named 'time' to the fragment shader. Test that the variable is being properly passed by setting the fragcolor.r to sin(time).

3. Change the vertex and fragment shader code (template_vs.glsl and template_fs.glsl) to use *layout qualifiers* for all uniform variables. See page 24 for an example and explanation.

4. After making the change in part 3, it is redundant to call glGetUniformLocation(…) in the C++ code. Eliminate all of these function calls from the code by replacing, for example,

    ```
    int tex_loc = glGetUniformLocation(shader_program, "diffuse_color");
    ```

    with

    ```
    const int tex_loc = ???;
    ```

    where **???** should be replaced with whatever value you specified in the layout qualifier for that variable in the glsl shader code.