HELLO WINDOW

*Documentation of the GLFW functions will be slightly lackluster because it is **VERY HEAVILY** documented in code comments. If you want to learn more about what a GLFW function does, look in the code.*

Include the GLAD and GLFW header files.

Initialize the GLFW window.
- This involves initializing GLFW, and then setting the major and minor version of OpenGL used and the profile used, using window hints.
- Initializing GLFW Window

Create the window object, which holds all the windowing data, and change the current context to that newly created window object.
- Creating GLFW Window

Initialize GLAD so we can use OpenGL functions.
- Initializing GLAD

Lastly, set up the viewport.
- The viewport is used to tell OpenGL how we want to display the data and coordinates with respect to the window.
- Viewport
- **NOTE:** Processed coordinates in OpenGL are between -1 and 1, so if the viewport size is 800x600, a processed point at location (-0.5, 0.5) would be mapped to (200, 450) in screen coordinates.

We Have a Window! What Now?

Currently, resizing the window does not change the size of the OpenGL viewport.

Create a callback function on the window that gets called each time the window is resized.
Then, tell GLFW to call the function every time the window is resized by registering the callback function.
- Window Resize Callback

The render loop is where the whole application comes together to output something.
- A render loop runs until the window or application is told to terminate.
- An iteration of the render loop is commonly called a frame.

Drawing to the screen is accomplished by swapping between the **front** and **back** buffers in the render loop.
- The **front buffer** contains the pixels that are currently being displayed on the screen while the **back buffer** draws the pixels for the next frame.
- This prevents artifacting that would otherwise occur if there were only one buffer, since that buffer would be in charge of drawing the next frame while also being displayed on the screen.

To keep the window from closing instantly, create a render loop that swaps the buffers and processes polled window events.
- Basic Render Loop

Clean/delete all of GLFW's resources that were allocated once the render loop ceases.
- Cleanup

Input

Add some input control in GLFW.
- Set up an input processing function that closes the GLFW window when the escape key is pressed.
- Close Window Input Processing

Rendering

All of the rendering commands go in the render loop (obviously), since rendering occurs every frame.

Change the color of the window.
- Coloring the Window

GLAD and GLFW Includes

Wednesday, March 16, 2022 12:38 AM

NOTE: GLAD must be included before GLFW, since the include file for GLAD includes the required OpenGL headers behind the scenes (li ke GL/gl.h).

```
#include <glad/glad.h>
#include <GLFW/glfw3.h>
```

Initializing GLFW Window

Tuesday, March 15, 2022 9:05 PM

Since we're using OpenGL 3.3, we want to tell GLFW that so it can make the proper arrangements when creating the OpenGL context.

- This prevents GLFW from running if a user doesn't have OpenGL 3.3 or higher.

We also want to tell GLFW that we're using the core-profile mode, meaning we don't need the backwards-compatible features from legacy OpenGL.

```
int main()
{
    // Initializes GLFW
    glfwInit();
    // Configures the major and minor versions of OpenGL used (3.3)
    glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, 3);
    glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, 3);
    // Configures the type of OpenGL profile used (Core)
    glfwWindowHint(GLFW_OPENGL_PROFILE, GLFW_OPENGL_CORE_PROFILE);
    // If using Mac OS X, you need to also include this to the initialization
    //glfwWindowHint(GLFW_OPENGL_FORWARD_COMPAT, GL_TRUE);

    return 0;
}
```

List of all possible options and its corresponding values: GLFW Window Handling Documentation

Creating GLFW Window

Tuesday, March 15, 2022 10:25 PM

We want to create a window of size 800x600, with the title "My OpenGL App". We want it to be windowed, not fullscreen. We don't need it to share its context with another window. We want to make the context of the current thread this new window.

```
GLFWwindow* window = glfwCreateWindow(800, 600, "My OpenGL App", NULL, NULL);
if (window == NULL)
{
    std::cout << "Failed to create GLFW window." << std::endl;
    glfwTerminate();
    return -1;
}
glfwMakeContextCurrent(window);
```

Initializing GLAD

Tuesday, March 15, 2022 10:38 PM

We pass GLAD the function to load the address of the OpenGL function pointers which is OS-specific. glfwGetProcAddress defines the correct function based on which OS we're compiling for.

```
// Initializes GLAD
if (!gladLoadGLLoader((GLADloadproc)glfwGetProcAddress))
{
    std::cout << "Failed to initialize GLAD" << std::endl;
    return -1;
}
```

gladLoadGLLoader(load)

Viewport

Tuesday, March 15, 2022 10:58 PM

```
// Allows OpenGL rendering in the whole window
glViewport(0, 0, 800, 600);
```

If we set the viewport dimensions to be smaller than GLFW's dimensions, then all OpenGL rendering would be displayed in a smaller window with extra space for other elements outside the OpenGL viewport.

Window Resize Callback

Tuesday, March 15, 2022 11:16 PM

```
// Changes the size of the viewport
void framebufferSizeCallback(GLFWwindow* window, int width, int height) {
    // Defines the region OpenGL rendering occurs in the window.
    glViewport(0, 0, 800, 600);
}
```

```
// Calls the framebufferSizeCallback function whenever window is resized
glfwSetFramebufferSizeCallback(window, framebufferSizeCallback);
```

Basic Render Loop

Tuesday, March 15, 2022 11:48 PM

```
// Render Loop
while(!glfwWindowShouldClose(window))
{
    // Swaps the front and back buffers
    glfwSwapBuffers(window);
    // Processes events that have occurred since the last loop
    glfwPollEvents();
}
```

Cleanup

Wednesday, March 16, 2022 12:02 AM

```
// Deletes all of GLFW's resources
glfwTerminate();
return 0;
```

Close Window Input Processing

Wednesday, March 16, 2022 12:14 AM

```
// Controls input processing
void processInput(GLFWwindow *window)
{
    if(glfwGetKey(window, GLFW_KEY_ESCAPE) == GLFW_PRESS)
        glfwSetWindowShouldClose(window, true);
}
```

```
// Render Loop
while (!glfwWindowShouldClose(window))
{
    processInput(window);

    glfwSwapBuffers(window);
    glfwPollEvents();
}
```

Coloring the Window

Wednesday, March 16, 2022 12:31 AM

We clear the color buffer (otherwise, the results from the previous frame would still be shown), and then change the color stored by the color buffer.
NOTE: glClearColor holds the color that is applied after glClear is called.

```
//
glClearColor(0.2f, 0.3f, 0.3f, 1.0f);
// Clears the color buffer of the screen
glClear(GL_COLOR_BUFFER_BIT);
```

glClearColor is a state-setting function.
glClear is a state-using function.