CREATING A WINDOW

The first thing we need to do is create an OpenGL context and an application window.

- This is specific per OS

There are many libraries that provide this functionality, such as:

- GLUT
- SDL
- SFML
- GLFW

I'll be using **Visual Studio 2022** as my IDE, and the notes will reflect such.

GLFW

GLFW is a library, written in C, that provides the bare necessities required for rendering to the screen using OpenGL.
Allows creating an OpenGL context, defining window parameters, and handling user input.

CMake

CMake is a tool that can generate project/solution files for Visual Studio from a collection of source code files using pre-defined CMake scripts.
Allows generating a Visual Studio project file from GLFW's source package, which can be used to compile the GLFW library.

Building GLFW

1. Download the source package and extract it
   a. GLFW Download Link
   b. The important parts are the resulting **library** from compilation and the **include** folder.
   c. The pre-compiled binaries are available, which makes building much easier, but this is for practice.
2. Download and install CMake
   a. CMake Download Link
   b. CMake is used to generate project/solution files for your IDE from a library's source code.
   c. This is used to compile the GLFW library.
3. Open CMake, select the glfw-*VERSION/*glfw-*VERSION/* folder as the location for the source code
4. Create a folder in that same directory named "build" and select that folder as the location to build the binaries.
   a. **NOTE:** This is how the pre-compiled binaries are made. All of this can be skipped if you have the pre-compiled binaries already.
5. Click Configure, then click Generate
   a. Make sure you are building for the version of your IDE you want before clicking generate.
6. Open the project file in the build folder (i.e. GLFW.sln) and build it.
   a. CMake automatically configured the project file, so you just have to build it to get the compiled library file (build/src/Debug/glfw3.lib).

Compilation

After generating the project file with CMake, you can just build it to get the compiled library file.
The IDE still needs to know where to find the library and the include files, so you can either:

1. Add the contents of GLFW's /lib and /include folders to the IDE's /lib and /include folders, respectively.
2. **(RECOMMENDED)** Create a new set of directories at a chosen location that contains the header files/libraries from third party libraries to which you can refer to from your IDE/compiler.

Our First Project

Open Visual Studio and create a new C++ project from the "Empty Project" template.

- Make sure to change from 32-bit (x86) to 64-bit (x64)!

Linking

In order for the project to use GLFW, we must link the library with our project.

- First, set the library and include directories of the project in *Configuration -> VC++ Directories*.
- Then, specify **glfw3.lib** in the linker settings by going to *Configuration -> Linker -> Input* and adding the library to the additional dependencies.

OpenGL Library on Windows

Add **opengl32.lib** to the linker settings.

- **NOTE:** The 64-bit version of the library has the same name as the 32-bit version.

GLAD

Since OpenGL is just a specification, it is up to the driver manufacturer to implement the specification to a driver that the specific graphics card supports.
Because there are many different version of OpenGL drivers, the location of most its functions is unknown at compile-time and need to be queried at run-time, which is pushed off onto the developer.

- The retrieval of these locations is OS-specific, cumbersome, and must be done **for each function you may need** that is not yet declared. This is where GLAD comes in.

GLAD is an open-source library that abstract away the setup shown above so you would only need the last two lines to achieve the same result.
GLAD uses a web service for download.

- GLAD Configuration Link

1. Use these setting for generating the GLAD library files:
   - Language: C/C++
   - Specification: OpenGL
   - API: gl Version 3.3
   - Profile: Core
   - Generate a loader: Yes
2. Copy the folder in /include into your /include directory and add the glad.c file to your project.