Using CMake

We will use CMake, a cross-platform, open-source make system. The provided homework code will be developed on MaxOS using CMake/clang, but has been tested and should also work on Linux using CMake/g++, and Windows using CMake/Visual Studio. Note that fighting through the installation & compilation of the various graphics libraries on a specific OS/hardware can be frustrating. Use the Google and please ask for help if you are stuck.

When you start a new assignment/project, you should first run cmake on the provided code following the directions below. These steps allow you to setup the build environment, verify installation of all libraries, ensure the project correctly compiles and links, and allows you to test the initial executable. These steps create the Makefile or Visual Studio project file for your system. Once completed you can launch your favorite source code editor or Integrated Development Environment (IDE) and start working on the assignment.

Please let the instructor know (by email or through the Discussion Forum) if there are errors/updates to any of the information below. Also, please share any additional installation/compilation instructions you have for your specific environment. And if you catch any portability bugs in the code while working on the homework let us know so we can update the provided files.

Common Instructions for all Operating Systems

- Install CMake. You'll need version 3.1.1 or later.
  Note for the Windows OS: Windows CMake should be on the path ahead of Cygwin CMake. This is a common mistake!
- Install GLFW, GLEW, and GLM (for Linux/Windows) or Metal (for Mac) following the Graphics library installation notes.
- Checkout the repository from Submitty with the provided files into a new directory, e.g., `AdvancedGraphics/hw0/src`. The CMakeLists.txt file, all the .cpp and .h files, the shaders, the data files, as well as your README.txt file will be in this directory.

- Create a separate `build` directory for building this project. We suggest that you put this directory next to your `src` directory for this project -- e.g.,
  - `AdvancedGraphics/hw0/src`
    contains the provided code files (`CMakeLists.txt`, `.cpp`, `.h`, etc.).
  - `AdvancedGraphics/hw0/build`
    will be initially empty, but will be used to store all of the cmake / compilation process files and the executable.

  Note: Separating the source code from the compilation/build directory is a good software development strategy and works well with version control (we'll be using git version control with Submitty!). You should "check in" all edits to files (and any new files you may create) in the `src` directory. It also makes it simple to submit the source code w/o submitting any platform-specific compilation files. To do a clean build from scratch, you can simply delete and recreate the `build` subdirectory.
- It may not be necessary, but we will include these files in your src/ directory to help CMake find the libraries you've just installed:
  - FindGLFW.cmake
  - FindGLM.cmake

Now on to the OS specific CMake instructions...

On Linux or MacOS

- Open up a terminal/shell and `cd` into the `build` directory you created above. Then run CMake giving it the relative path to the source code directory. For example:

  ```
  cmake ../src
  ```
- Now build the program:

  ```
  make
  ```

  Your executable will be in the current directory (or on a Mac, in a subdirectory).
- *On Linux*, to run the program, and refer to a data file that is in the `src` directory, you'll type something like this:

  ```
  ./ifs --input ../src/fern.txt
  ```

  On Mac (with Metal), it will be something like:

```
./ifs.app/Contents/MacOS/ifs --input ../src/fern.txt
```

- If you add new .cpp files (or .m Objective-C files), be sure to add them to the CMakeLists.txt file.
- To rebuild the program after changing the source files, just type `make`.

On Windows with Visual Studio

Note: Cygwin does not support the more recent versions of OpenGL (yet?). So on Windows, you will use the Visual Studio compiler.

Note: Windows Subsystem for Linux (WSL) is great for CSCI 1200 Data Structures and some other courses... But does it work for graphics/GPU programs? Unlikely. Why try to do things the really hard way? Either use native Windows or dual boot Linux.

- Install Visual Studio
  Any recent version of Visual Studio should work fine for the assignments in the course. Adjust the instructions below to insert your specific version of Visual Studio.
  Note: You will want the full "Visual Studio" IDE. Not just Visual Studio Code / VS Code.
- Launch the Visual Studio command shell (not just the cmd shell and not a Cygwin shell). From the Start menu, under All Programs, find your Visual Studio version and expand it. Then expand Visual Studio Tools. Select the "Visual Studio 20XX Command Prompt". Or, on Windows 8.... hold down the windows key and hit 'q', then search for "Visual Studio Tools" or "VS20XX" or "Command Prompt", and double click on the "Developer Command Prompt for VS20XX". Make sure to use the command prompt *without "x64"* in the name if you're using 32-bit libraries, and use a command prompt *with "x64"* in the name for 64-bit libraries. You don't need "Cross Tools".
- Within the Visual Studio command shell, navigate (using cd) into the `build` directory you created above. Then run CMake giving it a generator corresponding to your version of Visual Studio and the source code directory. Some examples are below...

  To list the available generators, run:

  ```
  cmake --help
  ```

  For Visual Studio 2019, using the 64-bit graphics libraries:

  ```
  cmake -G"Visual Studio 16" -A x64 ../src
  ```

Note: The '-A x64' argument may be the default for Visual Studio 2019, so you may be able to just type:

```
cmake -G"Visual Studio 16" ../src
```

For Visual Studio 2015 using the 64-bit graphics libraries:

```
cmake -G"Visual Studio 14 Win64" ../src
```

For Visual Studio 2015 using the 32-bit graphics libraries:

```
cmake -G"Visual Studio 14" ../src
```

It should say something like: "Building for: Visual Studio 2019" or "Building for: Visual Studio 2015".
Hopefully it finds all the necessary libraries and completes successfully!

If it complains about one of the graphics libraries not being found, make sure you have installed them correctly. (NOTE: You need to re-launch the Visual Studio command shell after installing new libraries).

- Now actually run the compiler:

```
cmake --build .
```

And hopefully the build is successful and you have an executable will be in Debug\your_program.exe or debug\your_program.exe

- To run your executable with a data file in the source directory, type something like this:

```
debug\ifs.exe --input ../src/fern.txt
```

- Once your build environment is setup, you can launch the Visual Studio IDE by opening the solution file (named XXX.sln). E.g.,

```
start hw0.sln
```