OpenGL, GLFW, GLEW Installation Guideline

To write graphics programs for this course, you will need three things:

OpenGL: OpenGL is the graphics library specification we will use to create our graphics programs. Note that OpenGL is simply a specification, which means it just defines the behavior of all the different OpenGL implementations. There is no official implementation of the library to download and use. Instead, operating system and GPU vendors generally provide users with their implementations.

Window manager (GLFW): OpenGL only deals with purely graphical tasks such as rendering, shading, etc. It does nothing to create and manage application windows or take input from the user. We will use the **GLFW** window management library during this course, but there are other viable alternatives such as freeglut or SDL.

GLEW: Open**GL** **E**xtension **W**rangler Library is a library that loads the supported OpenGL extensions for your platform. You will need it to load the supported OpenGL functions. There are other extension loaders available, and manually loading the extensions is also an option, but we chose GLEW as the simplest solution for this course. Note that you do not need GLEW on MacOS.

This document will assist you as you install OpenGL, GLFW, and GLEW for Ubuntu, Windows, and MacOS systems.

Linux (Ubuntu)

These instructions are for Ubuntu 20.04 64 bit. You might need to change the commands if you are using a different distro.

OpenGL

If you have a GPU on your machine and you have sufficiently updated drivers, your system should already have OpenGL version 4.x available to you. You can check this by using these commands:

```
sudo apt install mesa-utils
glxinfo | grep "OpenGL version"
```

If you do not have OpenGL 4.1 or later available on your device but you have a GPU, you should install the latest drivers from your GPU vendor. If your Linux distro has a driver manager, you can use it to install the GPU drivers. If not, you can use the following links to download the drivers from a web browser:

- AMD: http://support.amd.com/en-us/download/linux
- Nvidia: http://www.nvidia.com/object/unix.html
- Intel: https://01.org/linuxgraphics

After installing the latest drivers, please check again to make sure you have OpenGL version 4.1 or later available to you.

GLFW
To install GLFW you can use the command:

sudo apt install libglfw3

Glew
To install glew you can use the command:

sudo apt install libglew-dev

Compilation
Once you have all three libraries available on your machine, you can compile your graphics programs as follows:

g++ example.cpp -o example -lGL -lglfw -lGLEW

Windows

OpenGL
If you have a GPU on your machine and you have updated drivers, your system should already have OpenGL version 4.x available to you. You can check this by using this free application: http://realtech-vr.com/home/glview

If you do not have OpenGL 4.1 or later available on your device but you have a GPU, you should attempt to install the latest drivers from your GPU vendor. Here are some links to let you do that:

- AMD: https://www.amd.com/en/support
- Nvidia: http://www.nvidia.com/Download/index.aspx
- Intel: http://www.nvidia.com/Download/index.aspx

After installing the latest drivers, please check again to make sure you have OpenGL version 4.1 or later available to you.

GLFW & GLEW using NuGet
Nuget is a package manager included with the Visual Studio IDE. It allows a very easy setup of GLFW and GLEW. To install GLFW and GLEW with Nuget, in Visual Studio select **Tools -> NuGet Package Manager -> Package Manager Console** and then type “Install-Package nupengl.core”. Both libraries should then be available in your project environment.

Troubleshooting for Windows
- To disable precompiled headers for your project do; Select your project, use the "Project -> Properties" menu and go to the "Configuration Properties -> C/C++ -> Precompiled Headers" section, then change the "Precompiled Header" setting to "Not Using Precompiled Headers" option.
- When running in debug mode, If you see “cannot find or open the pdb file” error, Go to Tools -> Options -> Debugging -> Symbols and set the checkmark. Prefer running without debugging (Ctrl + F5) to avoid this.
- If you get an error regarding “_CRT_SECURE_NO_WARNINGS,” Go to Project -> Properties and then All Configurations -> C++ -> Preprocessor. Select the first field and add “_CRT_SECURE_NO_WARNINGS” without quotes to the first field. Make sure to add a semicolon (;) after your addition.

Alternative GLFW installation for Windows (Advanced)
If you do not wish to use NuGet, you can download the glfw library from the following link: https://www.glfw.org/ . You will have to then configure your project environment to include the library. This step will vary based on your IDE/compiler.
Alternative GLEW installation for Windows (Advanced)
If you do not wish to use NuGet, you can download the glew library from the following link: https://sourceforge.net/projects/glew/files/glew/2.1.0/glew-2.1.0-win32.zip/download . Like you did with GLFW, you need to configure your compiler/IDE so that it can access the necessary .dll, .lib and .h files.

MacOS

* These instructions were made for latest versions of MacOS (Ventura, Big Sur, etc). If you use old versions, you might run into issues. We suggest updating your system before attempting the installation. Note also that you do not need GLEW on Mac OS.

GLFW
Install GLFW using Terminal and Homebrew. You can get homebrew from https://brew.sh/ if you do not have it installed. Then you must run the command:

```
brew install glfw
```

OpenGL
Install Xcode from App Store.
Launch Xcode, from the File menu -> New -> Project.
Select macOS then Command Line Tool.
Select C or C++ as language.
Enter name, click Next, choose folder location and click Create.
Click on Project on the left pane, Select Build Phases, under Link Binary with Libraries click + button.
Use search, find and add **OpenGL** framework.
Add **libglfw** by clicking **Add Other.**
Default path for brew installs is /opt/homebrew/Cellar.
Select **libglfw.{version}.dylib (e.g. libglfw.3.3.dylib)** under: /opt/homebrew/Cellar/glfw/{version}/lib/ (e.g./opt/ homebrew/Cellar/glfw/3.3.8/lib/)

Some hints on Xcode project settings:
- Make sure that shader files are in the project working folder (you can set your working folder through Product->Scheme->Edit Scheme->Info
- Set your header search paths through Build Settings (so that you include all the header files provided)
- Disable “Metal API validation” through Product->Scheme->Edit Scheme->Diagnostic
- Set “Inhibit all warnings” to “yes” through Build Settings
- Include the file InitShader.cpp into your project