

Modern OpenGL Assignments Introduction and Setup

CS148 Fall 2015-2016

WARNING: DO NOT COPY AND PASTE COMMANDS FROM THE PDF INTO THE TERMINAL. IT IS SAFER TO MANUALLY TYPE OUT THE COMMANDS YOURSELF.

If you find any of the instructions below to be unclear, confusing, or otherwise incorrect please contact a CA or post on Piazza!

Requirements

Note that the OpenGL assignments have been modernized to use OpenGL 4.1 and (the newest version supported by Mac OSX) as such it will be necessary to have updated compilers/operating systems/drivers. If for whatever reason you can not run the OpenGL 4.1 assignments, contact the CA's and we will get you setup with the old assignments (OpenGL 2).

You will need to make sure you have the following:

1. C++ Compiler with C++11 Support
2. OpenGL 4.1 Support
3. CMake 2.8+
4. Python 2.7.x

If you are on Linux, you will also need to get the following dependencies on your system:

1. SDL 2
2. Open Asset Import Library
3. FreeImage
4. GLEW

If you are on Mac OSX, you will need to manually compile the following libraries:

1. Open Asset Import Library

On the other operating systems/libraries, we have provided pre-built binaries.

If any of the instructions below are unclear, ask a CA or post on Piazza.

Windows

C++ Compiler

Download Visual Studio 2015 (Community Version) for FREE: [here](#) to make sure you have Microsoft's latest compiler. Visual Studio 2013 should also work should you have that.

OpenGL 4.1 Support

Your system should have the OpenGL headers already. Make sure your drivers are updated from Nvidia/AMD/Intel.

CMake

Download the latest CMake binaries from the CMake website: [here](#).

Python 2.7

Download Python 2.7 from [here](#). Afterwards, make sure your PATH is set such that the Python 2.7 binary directory is included. If you have Python 3+ installed, make sure Python 3 is either not in the path or that running "python -version" in the command prompt runs Python 2.7 instead of Python 3.

TODO: Insert pictures for setting the system path, Check to see what would happen if Python3 and Python2.7 both installed.

Mac OSX

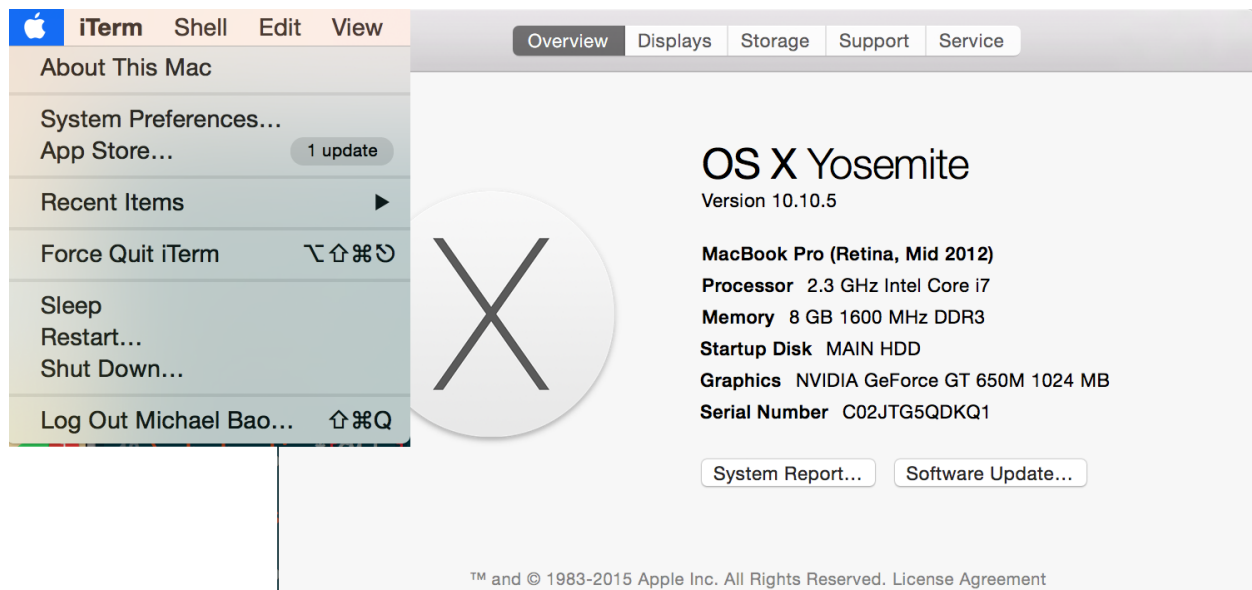
C++ Compiler

Make sure you have XCode installed. If you do not have XCode installed, you can go [here](#) to download it (it will redirect you to the app store). Make sure you download XCode 7 and not the XCode 7.1 beta. If you wish, you can click on 'Additional Tools' ([or here](#)) to find XCode 6.4.

Note that if you want to use the command line instead on Mac OSX, you will have to also install the Command Line Tools for your Mac OSX version and your Xcode version (this is also found under 'Additional Tools').

OpenGL 4.1 Support

Your system should have the OpenGL headers already once you install XCode. To see if your system supports OpenGL 4.1 look at Apple's OpenGL capabilities table: [here](#). To see what graphics card you have, you can click on the 'Apple' button and click on 'About this Mac' and in the window that pops up, your graphics card is displayed next to **Graphics**. Additionally, you'll want to make sure your Mac OS X is updated to at least 10.9 if not 10.10 (upgrade! It's free).



CMake

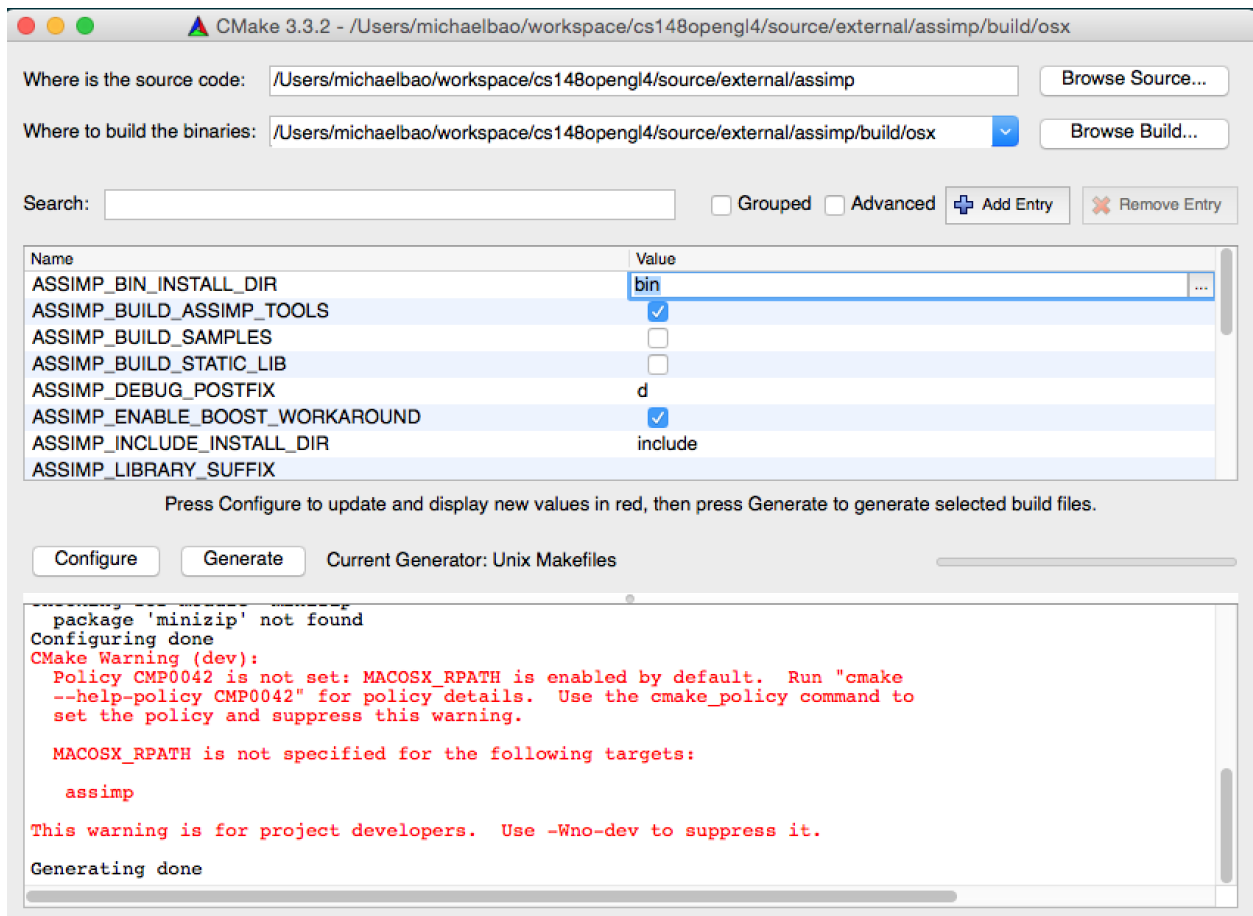
Download the latest CMake binaries from the CMake website: [here](#).

Python 2.7

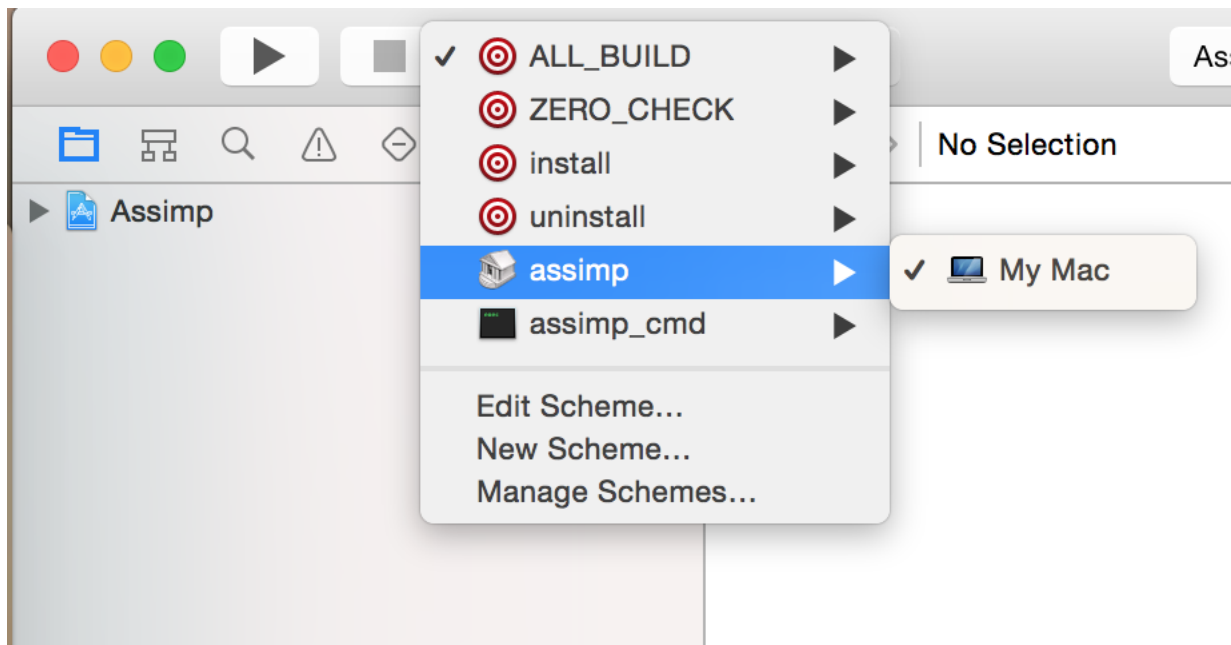
Download Python 2.7 from [here](#) and install it. Running "python -version" in the terminal should display 2.7.x.

Other Dependencies

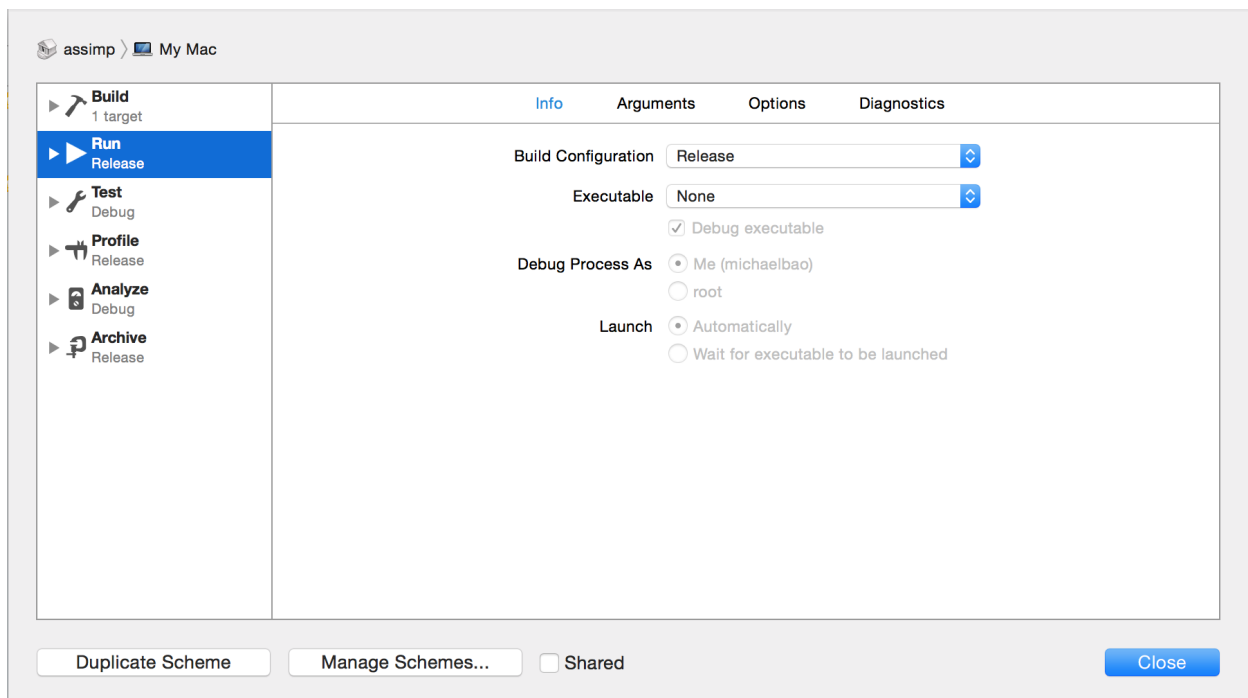
Open Asset Import Library Open up CMake (which you should have already installed) and point 'Where is the source code' to 'source/external/assimp' and 'Where to build the binaries' to 'source/external/assimp/build/osx'.



Then, hit the "Configure" button. At the screen that follows, choose "XCode" when it asks you to specify the generator for this project (feel free to choose Unix Makefiles if that is what you are comfortable with). Click "Generate" and then go to 'source/external/assimp/build/osx' using the Finder and open "Assimp.xcodeproj." In the top left, change the build target from ALL_BUILD (click on ALL_BUILD) and switch to "assimp".



Now you want to change the Debug build to be a Release build. You can find this option by clicking on "assimp" in the top left of XCode and then clicking on "Edit Scheme...". Under "Run" you will want to click on the "Info" tab and change the "Build Configuration" to "Release".



Now you can go to Product -> Build to compile the library. Then, go into "source/external/assimp/build/osx/-code/Release" and copy libassimp.dylib to "source/external/assimp/distrib/osx".

Linux

Note that the instructions here are tailored to Ubuntu. If you have a different distribution, the steps should be similar as the only major difference should be the package names. Talk to a CA if you need help with this.

C++ Compiler

On Linux, you'll want to be using G++ 4.6 or later. If possible, you'll want to be using G++ 4.8.1+. On Ubuntu you can run:

```
sudo apt-get install g++-4.8
```

You will also be needing to run the 'make' command so generally it is recommended that you install the 'build-essentials' package (on Ubuntu).

```
sudo apt-get install build-essentials
```

The G++ 4.8 package is available on Ubuntu 14.04 LTS or later! If you have another distribution, check your distribution package manager to see what version of G++ they have. If you have any problems with another distribution, contact a CA!

OpenGL 4.1 Support

Make sure you have the latest drivers for your GPU. After you installed the drivers (be it open source or closed source), you can double check to make sure you have OpenGL 4.1 by installing the "mesa-utils" package on Ubuntu (may be named "mesa-demos" on other distributions). Then in the terminal run, 'glxinfo — grep "OpenGL version"' and make sure you get a version that's greater than or equal to 4.1!

CMake

You can download the latest CMake using your package manager. On Ubuntu, you can run:

```
sudo apt-get install cmake
```

Python 2.7

In your command line, type "python -version" to see what version of Python you have/if it is installed. If it is not installed, or if you have the wrong version, you will want to run (on Ubuntu):

```
sudo apt-get install python
```

Other Dependencies

SDL 2 SDL 2 should be available in your distribution's package repository. On Ubuntu:

```
sudo apt-get install libsdl2-2.0-0  
sudo apt-get install libsdl2-dev
```

If it is not in the repository, you will have to compile manually. See a CA if you need assistance.

Open Asset Import Library I have provided the Open Asset Import library source code with the assignment framework under "source/external/assimp" as well as a convenient build folder "source/external/assimp/build/unix". In your terminal, go to the build folder and run:

```
cmake -G "Unix Makefiles" ../../..
make -jN
```

Where N is the number of cores you want to use for compilation. Afterwards, in the 'source/external/assimp/build/unix/code' folder, there should be a 'libassimp.a' file, copy this into 'source/external/assimp/distrib/unix'.

FreeImage FreeImage should be available in your distribution's package repository. On Ubuntu:

```
sudo apt-get install libfreeimage3
sudo apt-get install libfreeimage-dev
```

If it is not in the repository, you will have to compile manually. See a CA if you need assistance.

GLEW GLEW should be available in your distribution's package repository. On Ubuntu:

```
sudo apt-get install libglew1.10
sudo apt-get install libglew-dev
```

Other versions of GLEW should be okay. People using Windows/OSX will be using GLEW 1.13.

If it is not in the repository, you will have to compile manually. See a CA if you need assistance.

Compiling

Compiling is a two step process regardless of whether you are using the CMake GUI or the CMake Command-Line Interface (CLI). First you will use CMake to generate the proper build files (Visual Studio, XCode, or Makefiles) and use Visual Studio/Xcode/Makefiles to actually build the application (aptly named cs148opengl4). This section assumes that you have the assignment code unzipped somewhere.

CMake Configuration and Generation

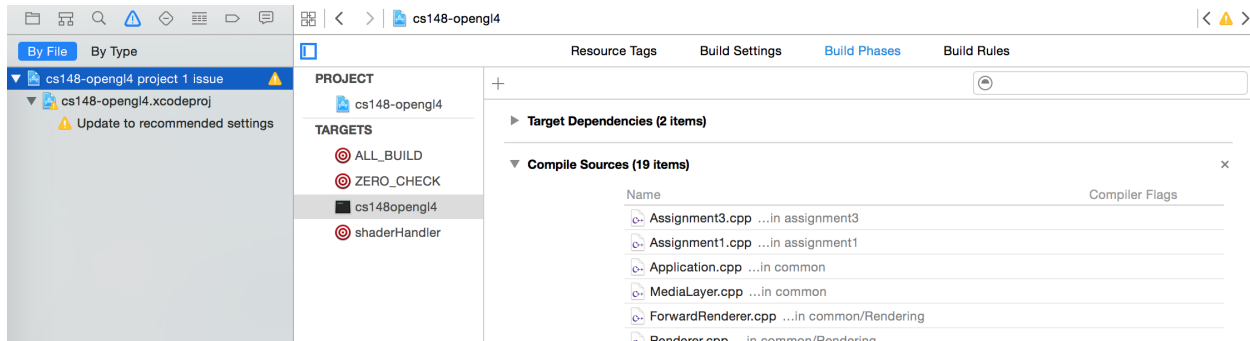
Special Notes for Visual Studio Solutions and XCode Projects

If you are using CMake to generate for Visual Studio or XCode, you can run CMake once (the very first time) and then use Visual Studio or Xcode to manage any new files you might add (instead of using CMake to do that for you). However, should you ever run CMake again, you will overwrite your Visual Studio solution/XCode project and lose any changes you made. To avoid having to run CMake to compile a new assignment, you can instead go into "main.cpp" and after the include statements, add a line that says

```
#undef ASSIGNMENT
#define ASSIGNMENT N
```

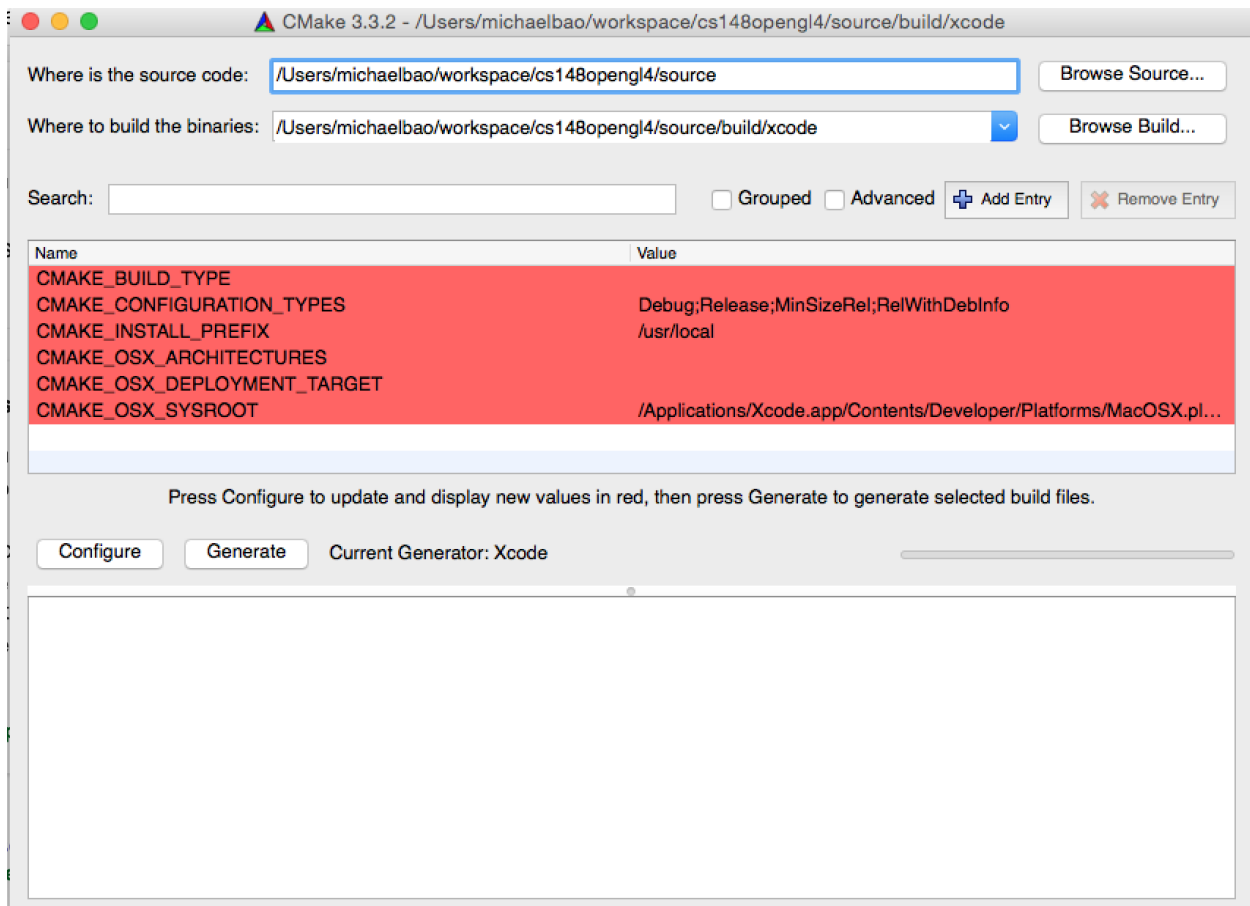
where N is the assignment you wish to compile for (1, 2, 3). Afterwards, make sure you have added the AssignmentN.h and AssignmentH.cpp file to the project. For XCode, make sure that the AssignmentN.cpp is under 'Compile Sources' in your 'Build Phases.' To get to your 'Build Phases', click on the 'cs148-opengl

project', then click on the 'cs148opengl4' target. If the AssignmentN.cpp file is NOT there, click on the plus button at the bottom of the 'Compile Sources' section and add it!

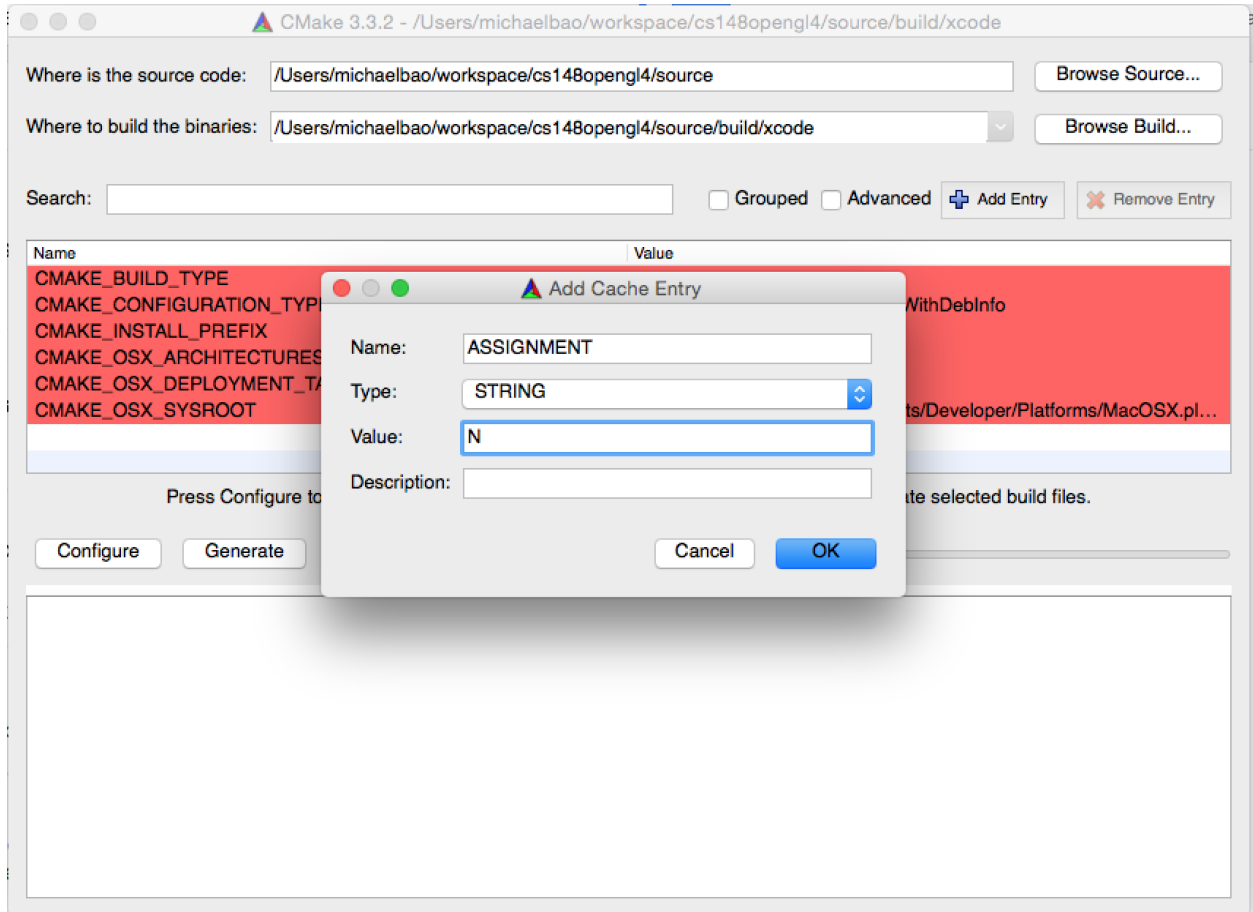


GUI (Windows, Mac OSX)

The images below are for CMake 3.3 on Mac OS X but they should be relevant for Windows as well. Open CMake and point 'Where is the source code' to the 'source' folder located wherever you extracted the assignment framework and point 'Where to build the binaries' to 'build/xcode' if you are on Mac OSX or 'build/vs' if you are on Windows (though it doesn't really matter, you can make your own folder if you really wanted to...).



Afterwards, click the 'Add Entry' button with 'Name' set to "ASSIGNMENT", 'Type' set to 'STRING' and Value set to N where N is the assignment number that you wish to compile (1, 2, 3).



At this point, you can click the Configure button. You should be greeted by a screen to choose the generator for the project. On Mac OSX you will want to choose "Xcode" (or Unix Makefiles, only do this if you know what you are doing!) and on Windows you will want to choose whatever version of Visual Studio you have. Now press the Generate button. Now your Visual Studio solution or XCode project should exist within your previously selected build folder.

CLI (Linux, Mac OS X)

Usually, you'll only be using the CLI on Mac or Linux so I will be ignoring Windows here. If you want to generate the build files for XCode, first go into the "build/xcode" directory and then run:

```
cmake ../.. -G "Xcode" -DASSIGNMENT=N
```

where N is the assignment you are trying to compile (1, 2, 3). Afterwards, the file "cs148-opengl4.xcodeproj" will exist and you will be able to open it up in XCode.

If you want to generate Makefiles, go into "build/unix" and then either go into the Release or Debug folder. Then you will want to run:

```
cmake ../../.. -G "Unix Makefiles" -DASSIGNMENT=N -DCMAKE_BUILD_TYPE=CONFIG
```

In this case, N is the assignment you are trying to compile (1, 2, 3) and CONFIG is either Release or Debug depending on which folder you are in.

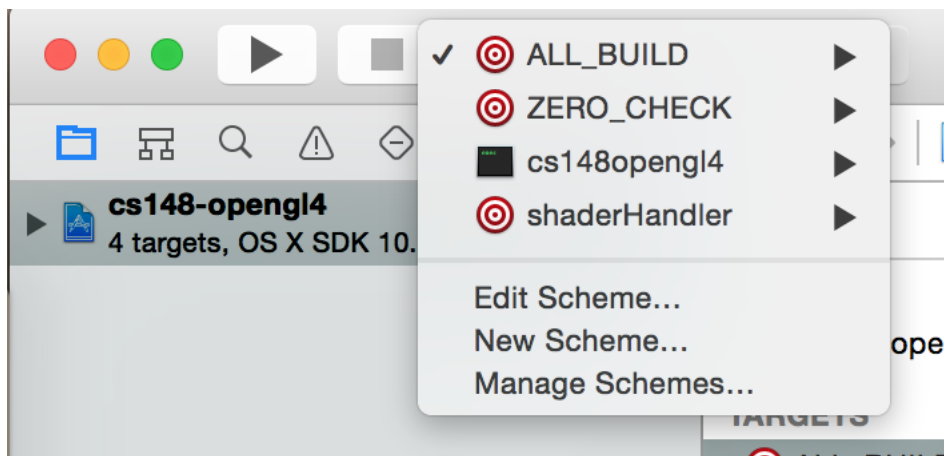
Compilation

Visual Studio

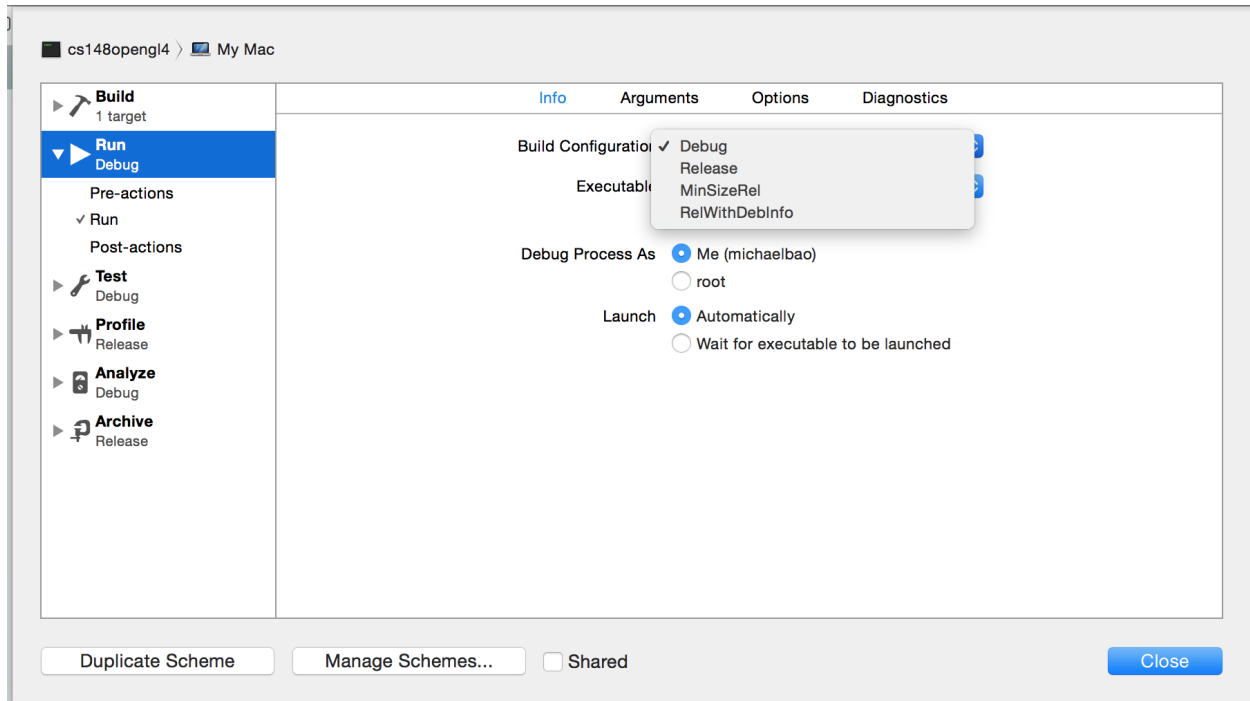
TODO: WRITE INSTRUCTIONS HERE

XCode

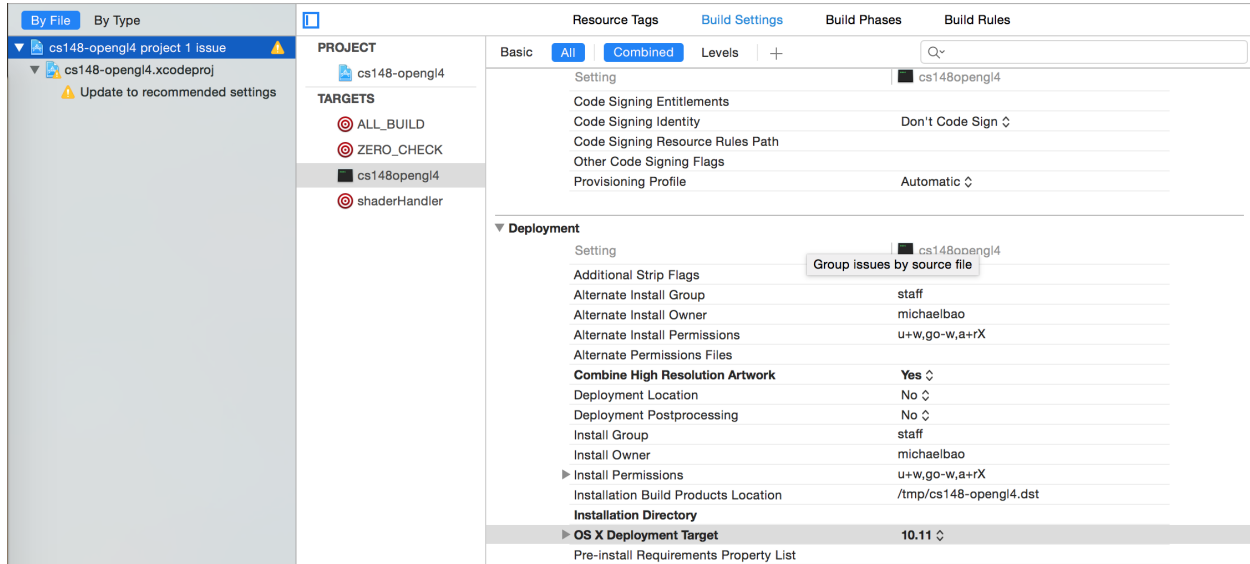
In the upper left corner, your build target by default should be set to ALL_BUILD, change it to cs148opengl4 (click on ALL_BUILD).



At this point, you can hit the 'Play' button and it will compile and launch the application! However, if you ever want to change from a Debug build to a Release build or vice versa, you will need to go same screen as before (click on cs148opengl4) and click on 'Edit Scheme...'. In the screen that pops up, make sure on the left, you have 'Run' selected and on the right, you are in the 'Info' tab. You should see a build configuration dropdown from which you can select the build type that you want.



If you are running XCode 7, you may run into an issue where it tells you that you can not run cs148opengl4 because the deployment target is too high. In that case, click on the cs148-opengl4 project, click on the cs148opengl4 target, click on 'Build Settings', under 'Deployment' find 'OS X Deployment Target' and change it to whatever version of OSX you have.



Makefiles

Inside the build/unix/Debug or build/unix/Release folder, type
make

or

```
make -jN
```

to use N cores (change N to 2 or 4, or however many cores you may have).

Afterwards, should compilation succeed, an executable named "cs148opengl4" will exist which you can run by typing

```
./cs148opengl4
```