Assignment 1: Introduction and Setup

CS 148 Autumn 2015-2016

Due Date: Monday, 28 September 2015 by 7pm

Follow the instructions carefully. If you encounter any problems in the setup, please do not hesitate to reach out to CA's on Piazza or come to office hour.

Background

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.

Requirements Overview

This section lists the requirements for compiling OpenGL4 on different system platforms. In the next sections, we will go over the details for Windows, Mac OS, and Linux.

- C++ Compiler with C++11 Support
- OpenGL 4.1 Support
- CMake 2.8+

Mac OSX You will need to manually compile this library:

- Open Asset Import Library

You will also need to extract the SDL2 framework file (instructions below).

Windows On Windows, we have provided pre-built binaries.

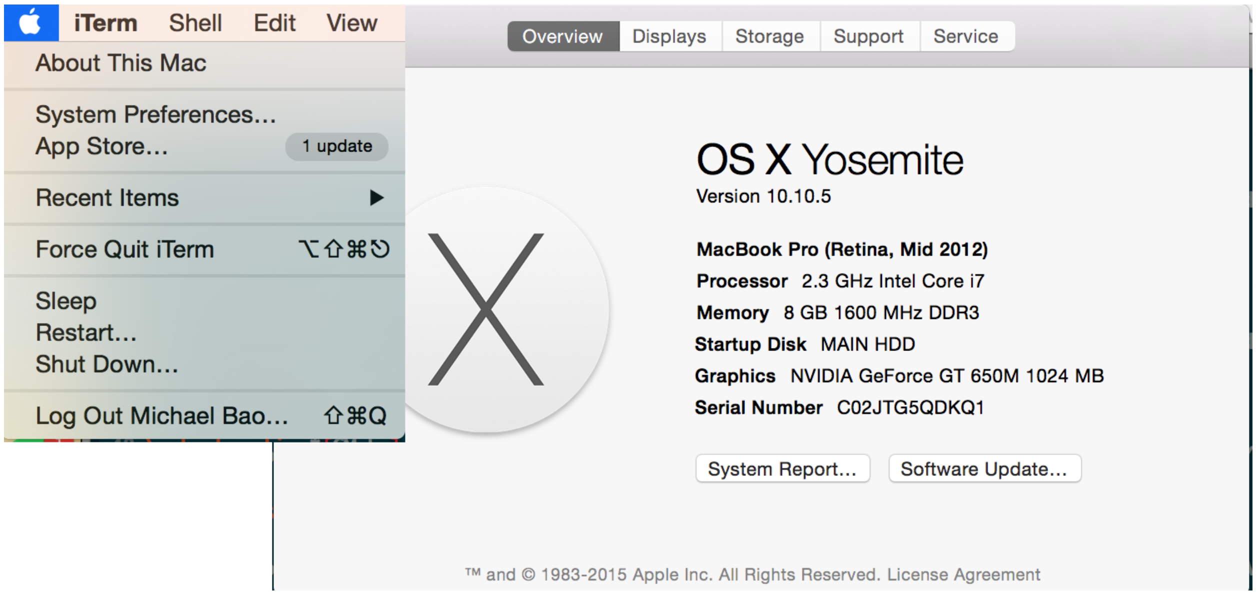
Linux You will need to manually compile/download from your package repository the following libraries:

- SDL 2
- Open Asset Import Library
- FreeImage
- GLEW

Mac OSX Setup

C++ Compiler Make sure you have XCode installed. If you do not have XCode installed, you can go here to download XCode 7 (it will redirect you to the app store). 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).



OS X Yosemite
Version 10.10.5

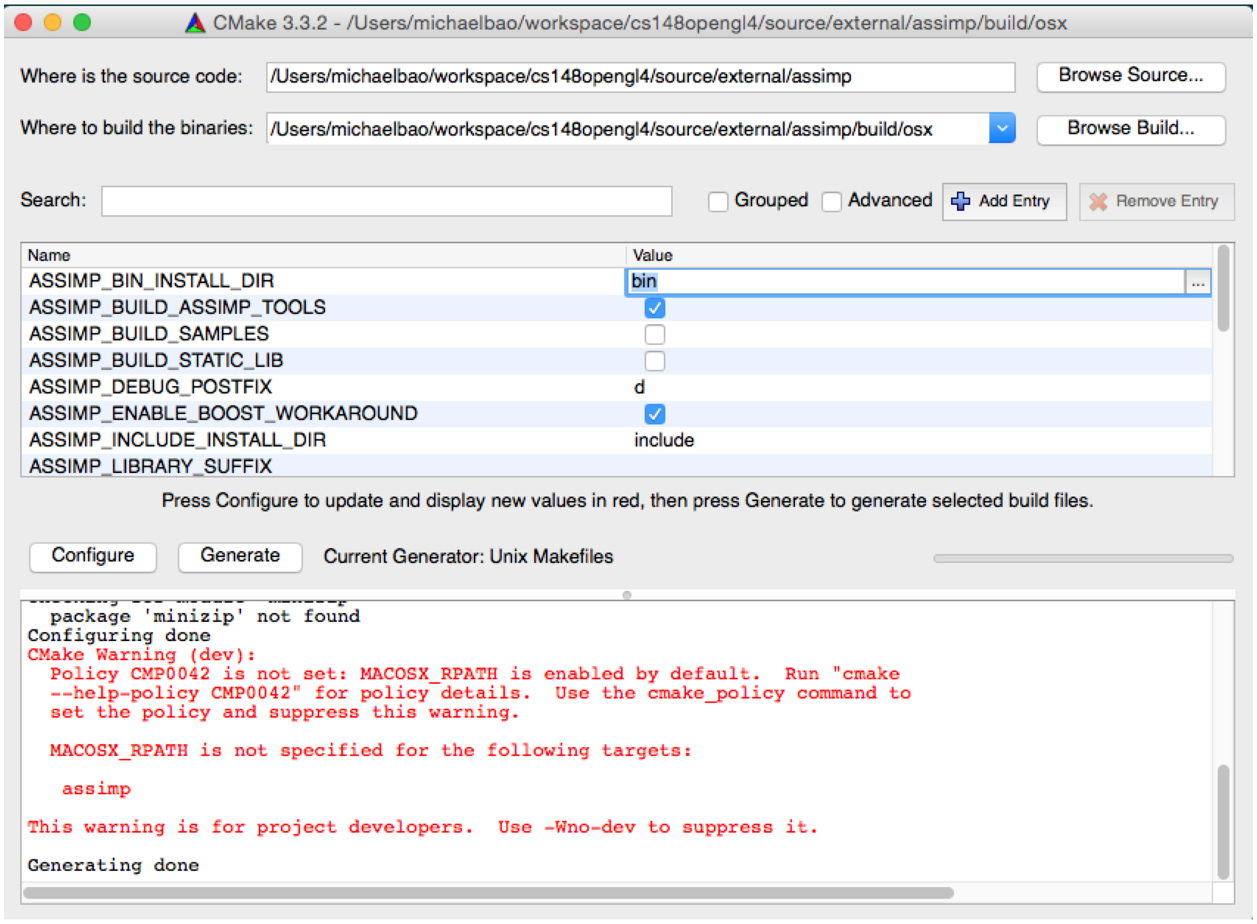
MacBook Pro (Retina, Mid 2012)
Processor 2.3 GHz Intel Core i7
Memory 8 GB 1600 MHz DDR3
Startup Disk MAIN HDD
Graphics NVIDIA GeForce GT 650M 1024 MB
Serial Number C02JTG5QDKQ1

™ and © 1983-2015 Apple Inc. All Rights Reserved. License Agreement

CMake Download the latest CMake binaries from the CMake website: here.

SDL2 In the 'source/external/SDL2/osx' folder, there is an SDL2.dmg file. Double click it to open it and you should see a window pop up. In that window, there should be a SDL2.framework file. You will want to drag and drop this file back into 'source/external/SDL2/osx'.

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'.



CMake 3.3.2 - /Users/michaelbao/workspace/cs148opengl4/source/external/assimp/build/osx

Where is the source code: /Users/michaelbao/workspace/cs148opengl4/source/external/assimp

Where to build the binaries: /Users/michaelbao/workspace/cs148opengl4/source/external/assimp/build/osx

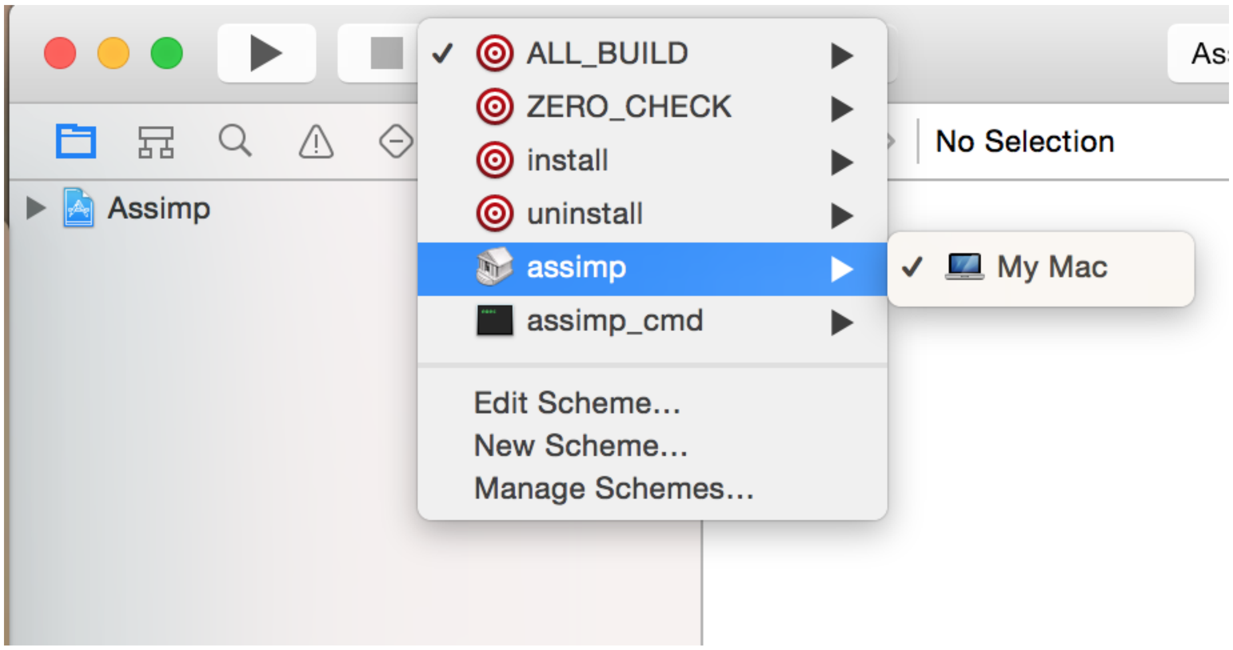
Name	Value
ASSIMP_BIN_INSTALL_DIR	bin
ASSIMP_BUILD_ASSIMP_TOOLS	✓
ASSIMP_BUILD_SAMPLES	☐
ASSIMP_BUILD_STATIC_LIB	☐
ASSIMP_DEBUG_POSTFIX	d
ASSIMP_ENABLE_BOOST_WORKAROUND	✓
ASSIMP_INCLUDE_INSTALL_DIR	include
ASSIMP_LIBRARY_SUFFIX	

Press Configure to update and display new values in red, then press Generate to generate selected build files.

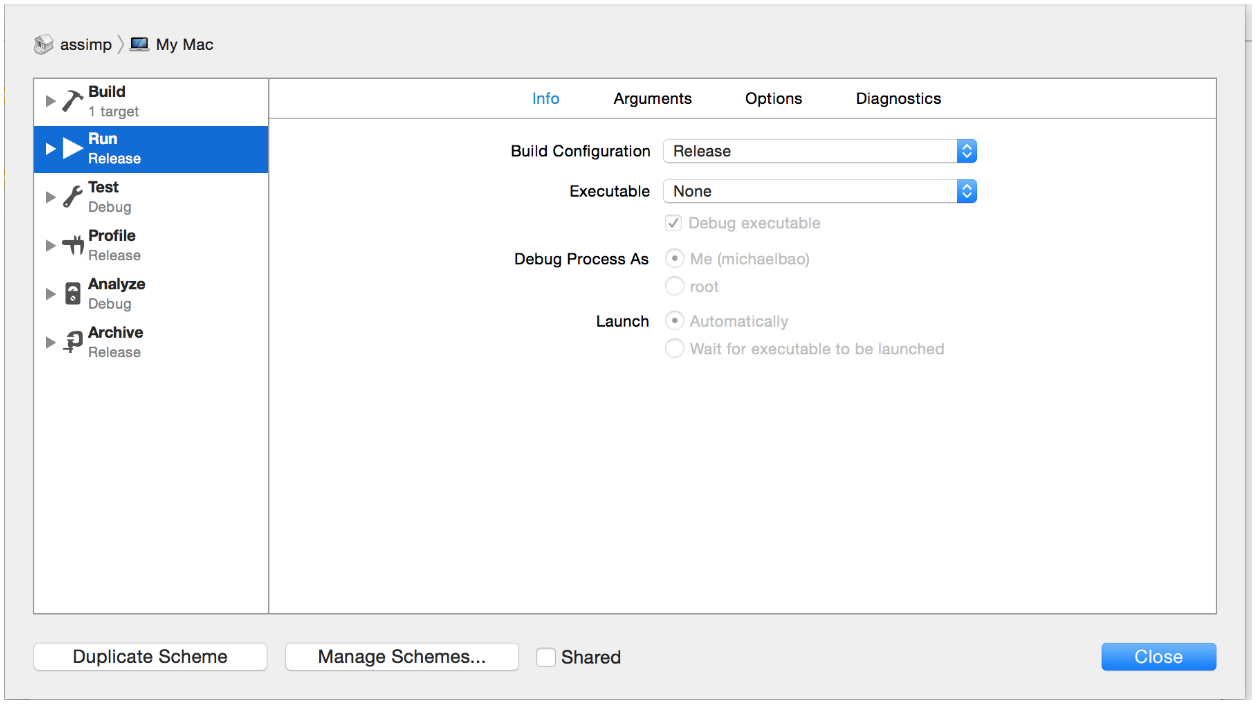
Current Generator: Unix Makefiles

```
  package 'minizip' not found
Configuring done
CMake Warning (dev):
  Policy CMP0042 is not set: MACOSX_RPATH is enabled by default.  Run "cmake
  --help-policy CMP0042" for policy details.  Use the cmake_policy command to
  set the policy and suppress this warning.

  MACOSX_RPATH is not specified for the following targets:

   assimp

This warning is for project developers.  Use -Wno-dev to suppress it.

Generating done
```

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".


assimp 〉 My Mac

Build: 1 target
Run: Release
Test: Debug
Profile: Release
Analyze: Debug
Archive: Release

Info | Arguments | Options | Diagnostics

Build Configuration: Release
Executable: None
[✓] Debug executable
Debug Process As: [● Me (michaelbao)] ○ root
Launch: [● Automatically] ○ Wait for executable to be launched

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".

Windows Setup

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.

Linux Setup

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-essential' package (on Ubuntu).

```
sudo apt-get install build-essential
```

The G++ 4.8 package is available on Ubuntu 14.04 LTS or later! If you have another distribution, check your distribution package mananager 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 the command below to make sure you get a version that's greater than or equal to 4.1!

```
glxinfo | grep "OpenGL version"
```

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

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

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 We 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 shared object library files for assimp. Copy them using the following command:

```
cp source/external/assimp/build/unix/code/libassimp.so* 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.

Compilation

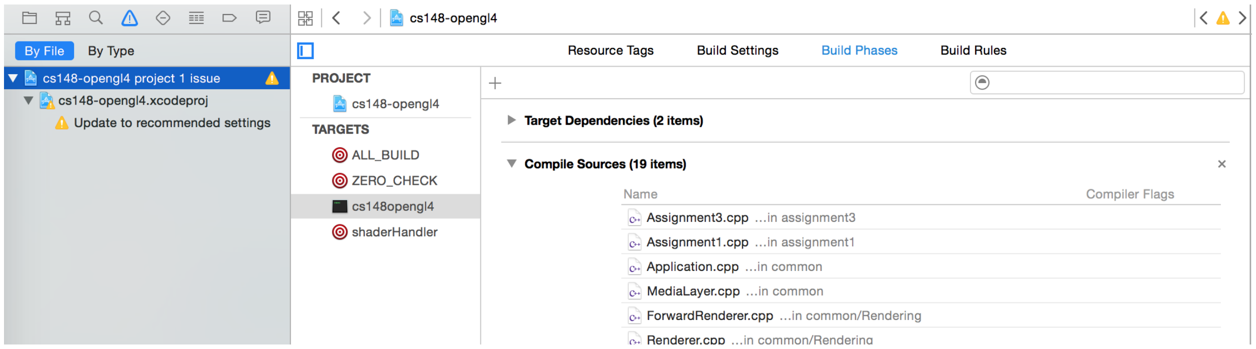
Compiling is a two step process regardless of whether you are using the CMake GUI or the CMake Command-Line Interface (CLI). First you will use CMake to generate the proper build files (Visual Studio, XCode, or Makefiles) and use Visual Studio/Xcode/Makefiles to actually build the application (aptly 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

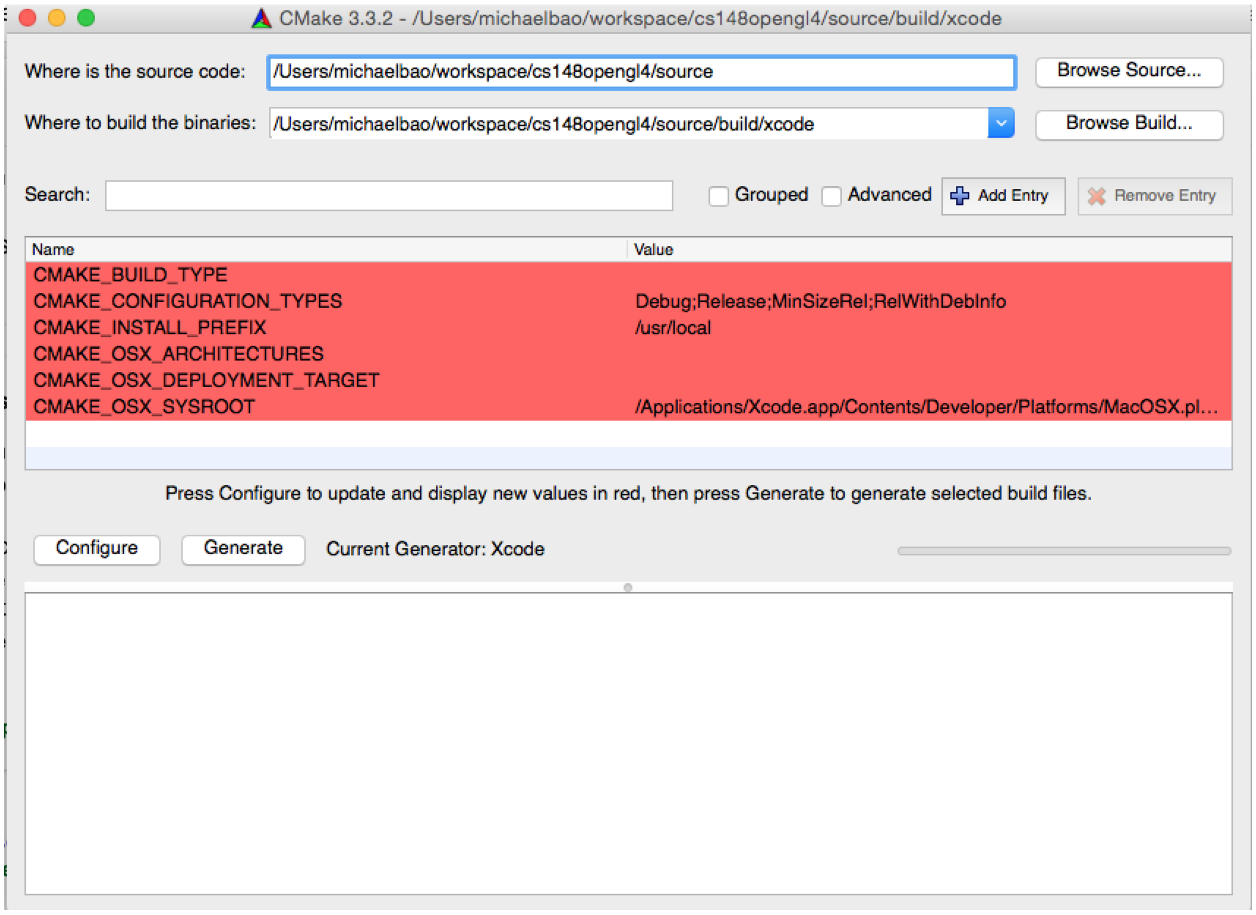
```
#undef ASSIGNMENT
#define ASSIGNMENT N
```

Afterwards, make sure you have added the AssignmentN.h and AssignmentN.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!



Name	Compiler Flags
Assignment3.cpp ...in assignment3	
Assignment1.cpp ...in assignment1	
Application.cpp ...in common	
MediaLayer.cpp ...in common	
ForwardRenderer.cpp ...in common/Rendering	
Renderer.cpp ...in common/Rendering	

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 1.


CMake 3.3.2 - /Users/michaelbao/workspace/cs148opengl4/source/build/xcode

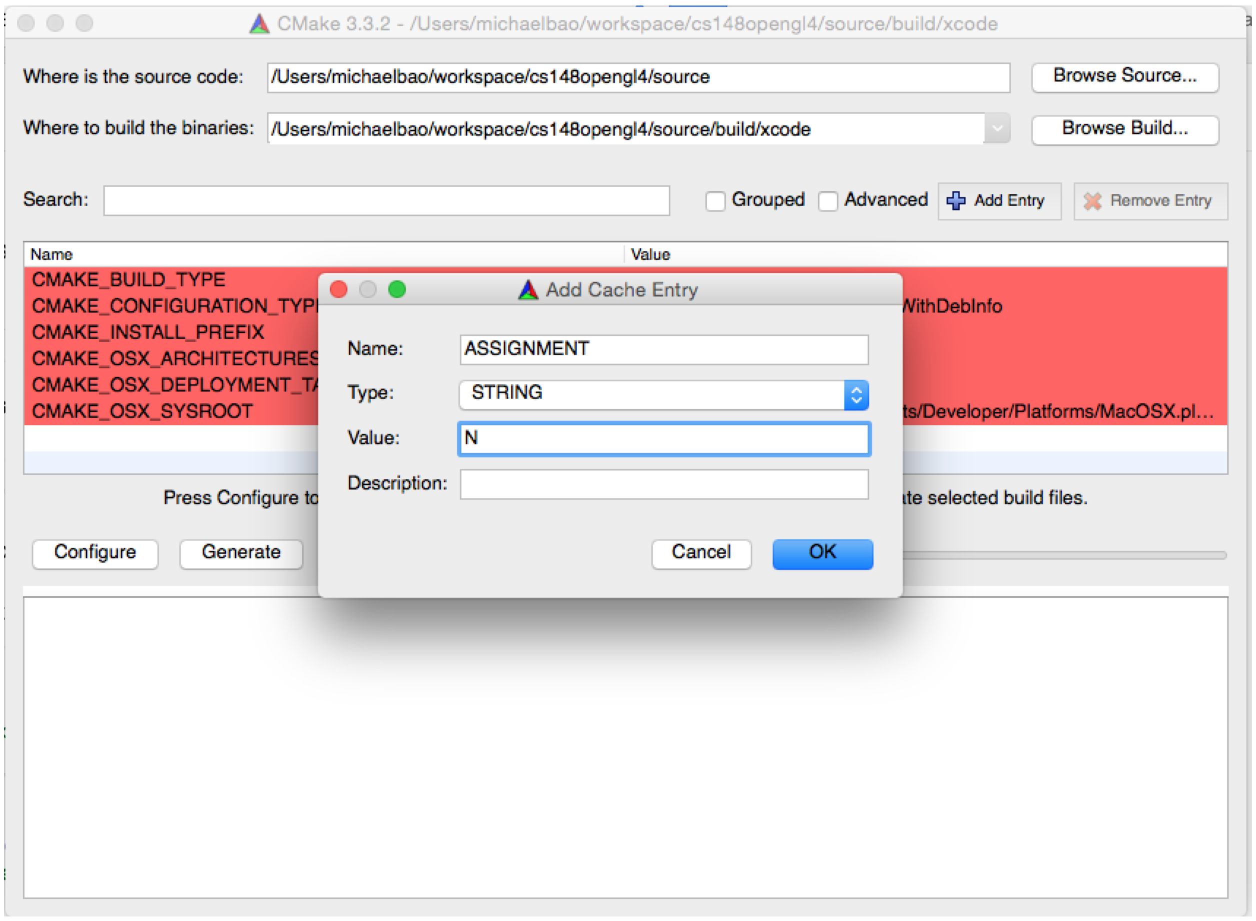
Where is the source code: /Users/michaelbao/workspace/cs148opengl4/source

Where to build the binaries: /Users/michaelbao/workspace/cs148opengl4/source/build/xcode

Name	Value
CMAKE_BUILD_TYPE	
CMAKE_CONFIGURATION_TYPES	Debug;Release;MinSizeRel;RelWithDebInfo
CMAKE_INSTALL_PREFIX	/usr/local
CMAKE_OSX_ARCHITECTURES	
CMAKE_OSX_DEPLOYMENT_TARGET	
CMAKE_OSX_SYSROOT	/Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.pl…

Press Configure to update and display new values in red, then press Generate to generate selected build files.

Current Generator: Xcode



Where is the source code: /Users/michaelbao/workspace/cs148opengl4/source

Where to build the binaries: /Users/michaelbao/workspace/cs148opengl4/source/build/xcode

CMAKE_BUILD_TYPE
CMAKE_CONFIGURATION_TYP
CMAKE_INSTALL_PREFIX
CMAKE_OSX_ARCHITECTURES
CMAKE_OSX_DEPLOYMENT_TA
CMAKE_OSX_SYSROOT

WithDebInfo

ts/Developer/Platforms/MacOSX.pl...

Name: ASSIGNMENT

Type: STRING

Value: N

Description:

Press Configure to ... te selected build files.

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 we 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=1
```

Afterwards, the file "cs148-opengl4.xcodeproj" will exist and you will be able to open it up in XCode.

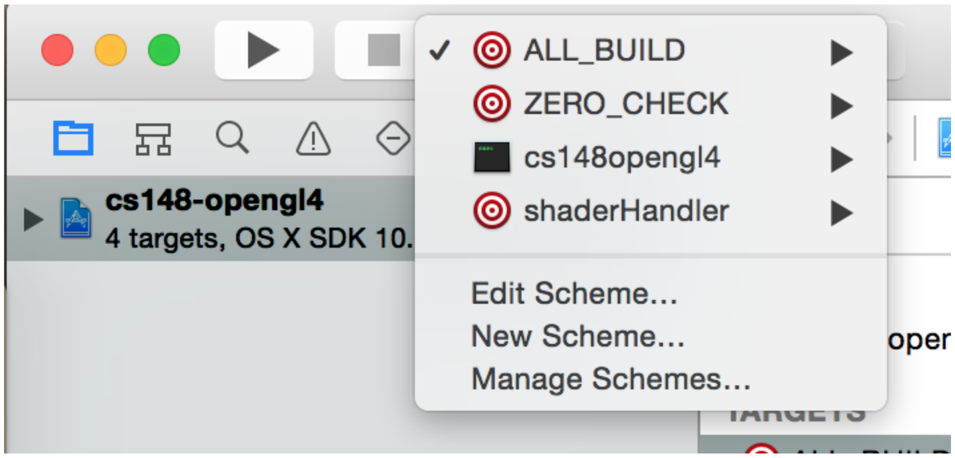
If you want to generate Makefiles, go into "build/unix" and then either go into the Release or Debug folder. Then you will want to run:

```
cmake ../../../ -G "Unix Makefiles" -DASSIGNMENT=1 -DCMAKE_BUILD_TYPE=CONFIG
```

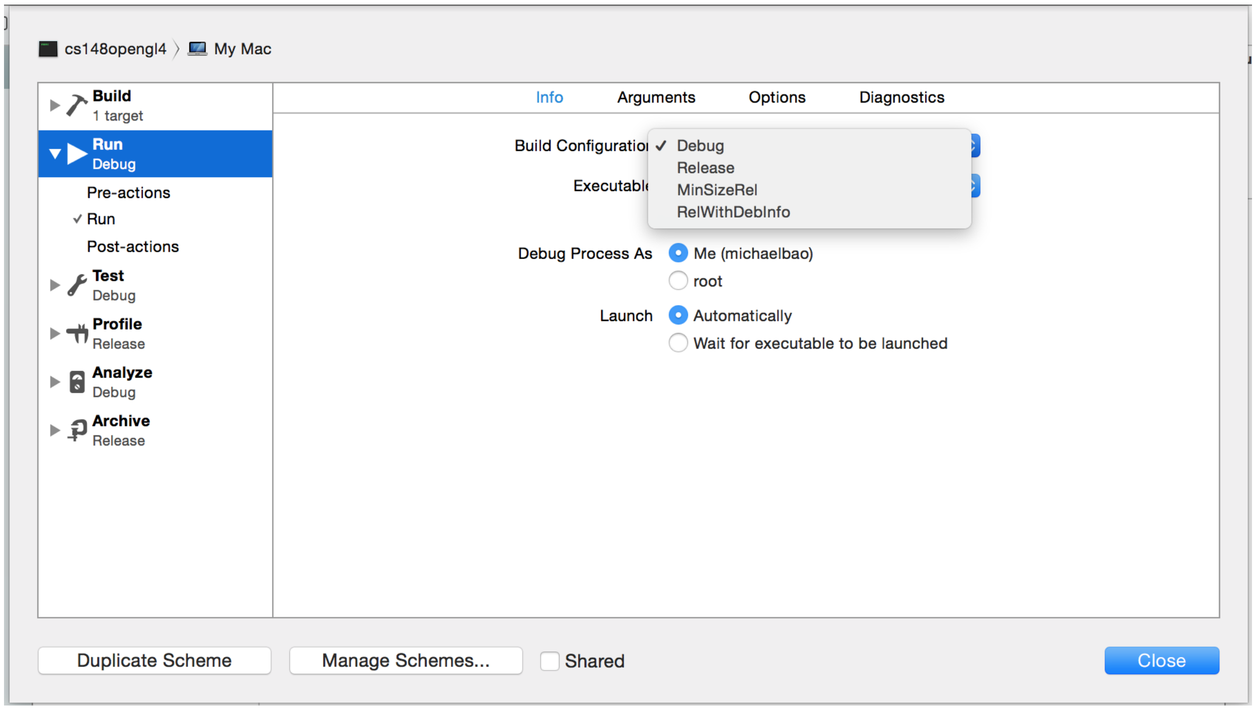
In this case, CONFIG is either Release or Debug depending on which folder you are in.

Compile and Run

Mac OSX - 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 Relase 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.

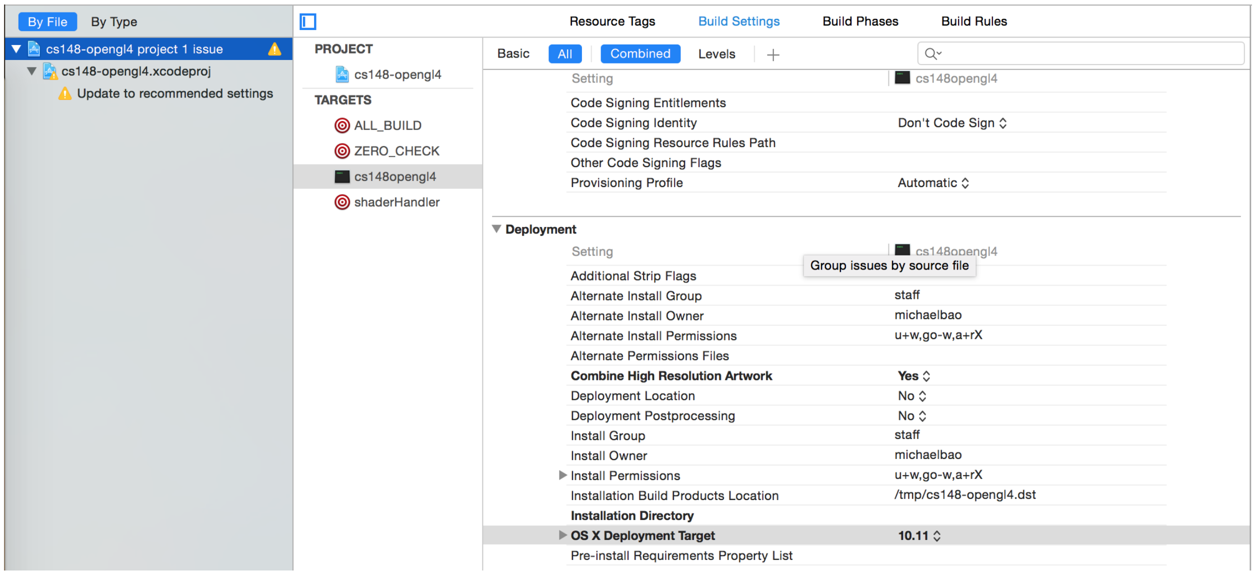


Build Configuratio ✓ Debug

Debug Process As [✓ Me (michaelbao)] root

Launch [✓ Automatically] Wait for executable to be launched

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.



Setting	cs148opengl4
Code Signing Entitlements	
Code Signing Identity	Don't Code Sign ◊
Code Signing Resource Rules Path	
Other Code Signing Flags	
Provisioning Profile	Automatic ◊

Deployment

Setting	cs148opengl4
Additional Strip Flags	
Alternate Install Group	staff
Alternate Install Owner	michaelbao
Alternate Install Permissions	u+w,go-w,a+rX
Alternate Permissions Files	
Combine High Resolution Artwork	**Yes ◊**
Deployment Location	No ◊
Deployment Postprocessing	No ◊
Install Group	staff
Install Owner	michaelbao
Install Permissions	u+w,go-w,a+rX
Installation Build Products Location	/tmp/cs148-opengl4.dst
Installation Directory	
OS X Deployment Target	**10.11 ◊**
Pre-install Requirements Property List	

Windows - Visual Studio Open the solution file (*.sln) inside the ”build/vs” folder, choose the build type (Release/Debug) you want, and then build the entire solution.

Note that, you must build the entire solution instead of just the cs148opengl4 project as it will copy the needed DLL's into the output directory. To reiterate, you must build the SOLUTION when you go to compile for a particular build type for the first time. Afterwards, you can right-click on the ”cs148opengl4” project, choose ”Set as StartUp Project”, and run/debug the program.

Linux - Makefile 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).

Should compilation succeed, an executable named “cs148opengl4” will exist which you can run by typing

```
./cs148opengl4
```

You are expected to see two red, blue, and green triangles.

Grading

This assignment will be graded on the following requirements

- OpenGL compiles on your computer and test program runs successfully.

according to the following rubric.

- ✓ – Meets the requirement.
- 0 – OpenGL does not compile and test program does not run.