**COSC 89.18/189.02 Physical Computing Starter Code Manual**

1. Quick Guide:

If you are already experienced with using CMake to generate a C++ project, please read the following paragraphs for a quick guide. If not, you may read Section 2 and 3 first and then get back to check this section as a summary.

**Code Structure**

The starter codebase is organized as **ext**, **src**, **proj**, and **viewer**. We put all external codes (e.g., the Eigen library) in **ext**. We put the common headers that can be shared among different subprojects in **src** (e.g., the classes of particles, mesh, grid, file IO, etc.). The folder **proj** maintains a number of subprojects we will use for demo or assignments. A subproject is dependent on **src** and **ext**, but is independent from any other subproject in **proj**.

Usually, you are asked to write code in one or multiple files in a subproject (e.g., in proj/a0\_hello\_world or in proj/a1\_mass\_spring). You don’t need to change files in **ext** or **src** (If you do, make sure to submit them for your assignment and let us know to avoid any compiling issue).

**OpenGL Viewer**

The folder **viewer** maintains the code for the OpenGL viewer that is used to visualize your simulation data. You may think of the viewer as a subproject, which is independent from other subprojects. The difference is, the viewer has its own external dependencies, including **freeglut**, **glm**, and **imgui**, which are all put in viewer/ext. The viewer also has its own common headers and source files in viewer/src/.

We provide viewer executables for different platforms. But you can also modify and compile the source code to generate your own customized viewer.

**Bulid and Compile**

We use CMake to separate the source code and the build files. **CMakeLists.txt** is maintained in each subproject. To generate build files (e.g., an .sln file for Windows or a makefile for Linux), you need to 1) create a **build** folder to hold all subprojects; 2) create a subfolder under build for a specific subproject (e.g., **build/a0\_hello\_world**); and 3) use CMake to generate the build files (e.g., in **build/a0\_hello\_world**) based on the source code (e.g., in **proj/a0\_hello\_world**).

**Executable**

The executables are generated in the Release or Debug folder under the subdirectory (e.g., **build/a0\_hello\_world/Release**).

The simulation data is generated in the same folder as the executable. Typically, we specify the argument –o for the folder name of each simulation (e.g., **build/a0\_hello\_world/Release/output**).

1. Compile and run the helloworld project:

**Step 1: Checkout the source code from GitLab and enter the codebase folder**:

**git checkout** [**git@gitlab.com:boolzhu/dartmouth-phys-comp-starter.git**](mailto:git@gitlab.com:boolzhu/dartmouth-phys-comp-starter.git)

**cd dartmouth-phys-comp-starter**

**Step 2: Build the project using CMake:**

Substep 2.0: Install CMake

[Windows]: Visit <https://cmake.org/download/> and download the latest version.

[Linux]: **sudo opt-get install cmake**

Substep 2.1: Create a build folder and a project subfolder in the codebase directory

**mkdir build\a0\_hello\_world**

**cd build\a0\_hello\_world**

Substep 2.2: Generate the project

If you use command line, type the following based on your operating system:

[Win64]:

**cmake -G "Visual Studio 15 2017 Win64" ../../proj/a0\_hello\_world**

[Win32]:

**cmake -G "Visual Studio 15 2017" ../../proj/a0\_hello\_world**

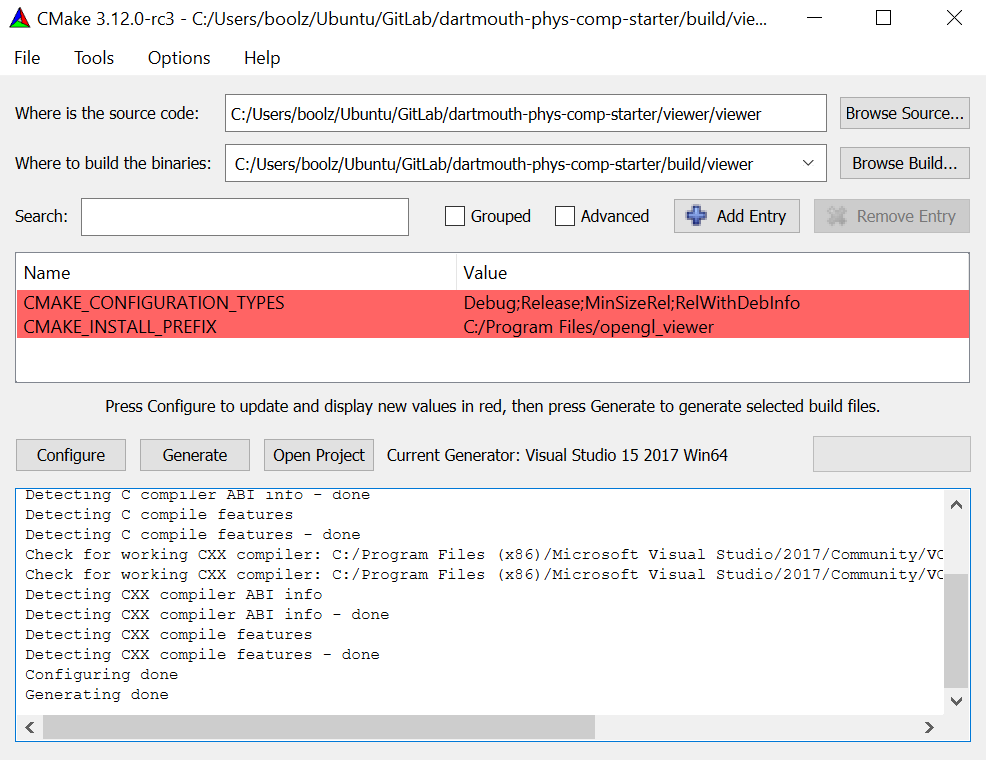
[Linux]:

**cmake ../../proj/a0\_hello\_world**

If you use CMake UI on Windows, input the paths for the source code and the build binaries, then click **Configure**, **Generate**, and **Open Project.**

Attention:

1. Typically, we set the source code path as **[Your path]/dartmouth-phys-comp-starter/proj/a0\_hello\_world** and the build path as **[Your path]/dartmouth-phys-comp-starter/build/a0\_hello\_world** ((see figure below)
2. Select the right generator for the project. For example, select **Visual Studio 15 2017 Win64** for Windows 64 and **Visual Studio 15 2017** for Windows 32.



**Step 3: Compile the code:**

[Windows]: Open the .sln project file, switch to Release mode, and click Build->Build Solution

[Linux]: make

If the code is compiled successfully, you will see an executable generated in the Release folder

**Step 4: Run the code:**

[Command]:

cd Release

[Windows]: hello\_world.exe

[Linux]: ./hello\_world.exe

If the code runs correctly, you should see the following output in the terminal:

--- Test\_Eigen ---

v: 1 2 3, 1 2

vi: 1 2 3, vt: 1 2 3

1, 2, 3, 4, 5,

--- Test\_Particles\_And\_Mesh ---

[Particle] #particles: 4

[Particle] X(0): 1 1 1, V(0): 2 2 2

[Triangle mesh] #vertices: 3, #triangles: 1

Compile and run the OpenGL viewer project:

**We provide executables for the OpenGL viewer in the repository. You may download and use them directly to visualize your simulation data.**

You may also want to compile the source code if you are interested in customizing your own viewer. To do so, you may follow the same steps as you make and compile the hello\_world project. The only difference is: the CMake source code path for the viewer is **[Your path]/dartmouth-phys-comp-starter/viewer/viewer** and the build path is **[Your path]/dartmouth-phys-comp-starter/build/viewer**

Once the viewer is compiled successfully, you will find the executable under the folder of build/viewer/Release. Run it in the command line (by default it will only open up an OpenGL window without reading any data), you will see a window like this:

