24783 Advanced Engineering Computation: Problem Set 6

(*) In the following instruction (and in all of the course materials), substitute your Andrew ID for where you see *yourAndrewId*.

START EARLY!

1 Check Out or Update Base Code and Libraries

Please make sure you have up-to-date libraries and course files before starting an assignment.

If you have not done working-directory set up as described in the first assignment (like in case you need to work from a different computer), please see Problem Set 1 and set up the working directory.

I assume you created the working directory called *24783* under your home directory and you checked out your Git repository in there.

Home directory is typically *C:\Users\username* in Windows, */Users/username* in macOS, and */home/username* in Linux, where *username* is the user name in your local computer.

First, open command-line (Developer PowerShell or Terminal), and move to your working directory by typing:

```
cd ~/24783
```

You need to check out (or clone) Git repositories once. If you have not checked out yet, do the following:

```
git clone https://yourAndrewId@ramennoodle.me.cmu.edu/Bonobo.Git.Server/course_files.git
git clone https://yourAndrewId@ramennoodle.me.cmu.edu/Bonobo.Git.Server/yourAndrewId.git
```

You need to replace "yourAndrewId" with your Andrew ID. You'll be asked to type in credentials.

Also we are going to use two additional repositories:

```
git clone https://github.com/captainys/MMLPlayer.git
git clone https://github.com/captainys/public.git
```

If you are successful, you should have the following directory structure under your home directory.

```
Your User Directory
├── (Other files and directories)
└── 24783
    └── course_files
```

```
├── public
├── MMLPlayer
└── yourAndrewID
```

If you already have checked out these repositories (most likely you did for Problem Set 1), you need to update (or git pull) in those repositories. By change directory to the location where you checked out repositries and then type:

```
git pull
```

To update all four repositories, you can type the following commands in a sequence:

```
cd ~/24783/course_files
git pull
cd ~/24783/yourAndrewID
git pull
cd ~/24783/public
git pull
cd ~/24783/MMLPlayer
git pull
```

2 Copy Base Code and Add to Git's Control

Do the following to make a copy of the base code to your directory.

```
cd ~/24783
cp -r course_files/ps6 yourAndrewID/.
```

Adding /. after the destination directory is a weak defense to prevent files from copied to wrong location in case you misspelled the directory.

And then type the following to add them to the Git's control:

```
git add yourAndrewID/ps6
```

Once you add ps6 sub-directory to Git's control, you can do commit and push as many times as you want to send your files to the server with no penalty before the deadline.

It is recommended to make frequent commits so that you can go back to earlier version in case you mess up.

3 Make a CMake Project

3.1 Top-Level CMakeLists.txt

Write a top-level CMakeLists.txt under ps6 sub-directory, so that:

- it enables C++11 features,
- it includes ps6 sub-directory,
- it includes MMLPlayer library (optional), and
- it includes public libraries.

Since public-library sources are located outside of your source tree, you also need to specify where the build files are written. Therefore, the lines for including the MML player and public libraries should look like:

```
add_subdirectory(../../public/src ${CMAKE_BINARY_DIR}/public)
add_subdirectory(../../MMLPlayer/ym2612 ${CMAKE_BINARY_DIR}/ym2612)
add_subdirectory(../../MMLPlayer/mmlplayer ${CMAKE_BINARY_DIR}/mmlplayer)
```

3.2 CMakeLists.txt Files for Executables

Write a CMakeLists.txt file in ps6_1 sub-directory. It must define an executable called ps6_1, which uses ps6.cpp. Don't forget MACOSX_BUNDLE keyword even if you are doing it in Windows or Linux to make it cross platform.

Once you set up CMakeLists.txt files, test-build ps6_1. First you create a build directory somewhere outside of ps3 directory, and then run cmake, and then type:

```
cmake --build . --target ps6_1 --config Release
```

to build the executable. If you omit –target ps6_1, you will need to wait for libraries unrelated to this assignment.

When you use add_subdirectory, use relative path. Absolute path like C:/Users/soji/24783/soji does not exist in the grading environment. Also use slash instead of backslash. Windows can accept both backslash and slash. Other platforms only accepts slash. I want you to learn cross-platform programming.

3.3 Add to Git's Control

After writing these files, make sure to add the files to Git's control.

4 Make Particle-Based Fluid Simulator Faster by Multi-Threading

The base code implements the following paper:

Mathias Müller, David Charypar, and Markus Gross, *Particle-Based Fluid Simulation for Interactive Applications*, Proceedings of SIGGRAPH 2003, pp. 154-159

It is a very simplified and approximated particle-based fluid simulator. It does not simulate incompressibility, therefore the volume of the water is not preserved. Nonetheless, it creates a good-enough simulation for visual effect.

Typically this type of particle-based simulation can be made faster by the following two methods:

(1) Multi-threading, and

(2) Using a background data structure such as a structured lattice.

In this assignment, you practice (1).

4.1 Print Frames-Per-Second on the Console Window Every Second

Use chrono library to measure time and print frames per Second (how many frames have been drawn in the past second) on the console window.

4.2 Expand the Field and Increase the Particle Count

The base code uses the field from -10 to 10 as defined by minX and maxX member variables of the FluidSimulator class. Change the values to -20 and 20 to double the width of the field.

Particles are generated in FluidSimulator::MakeTestCase function. Increase the range of Y where initial particles are placed from 50.0 times dy to 100.0 times dy to increase the initial particle count.

4.3 Read the Base Code and Identify Functions that can be made Multi-Threaded

You can use the calculation as is. You do not have to modify formulation and overall structure of the code. (If you are interested, go ahead and read the formulation.)

There are some functions where you can increase the performance by multi-threading.

4.4 Make Multi-Threaded

Make functions identified in section 4.3 multi-threaded. Use 8 threads.

As discussed in class, pre-start threads before going into the simulation loop and use condition variable to start a task and wait for the task to complete.

5 Test with Compiler Server

Make sure to test your source code with our compiler server!

6 Submission and Verification

Just as you did for other problem sets, submit your CMake script and the source file to the Git server.

Then clone your repository to a different location and make sure that all of your files have been sent to the Git server.

You can do the following:

```
cd ~
mkdir 24783Verify
cd 24783Verify
git clone https://yourAndrewID@ramennoodle.me.cmu.edu/Bonobo.Git.Server/yourAndrewId.git
```

Once you made sure all the files have been submitted, you can delete files and directories under 24783Verify directory.

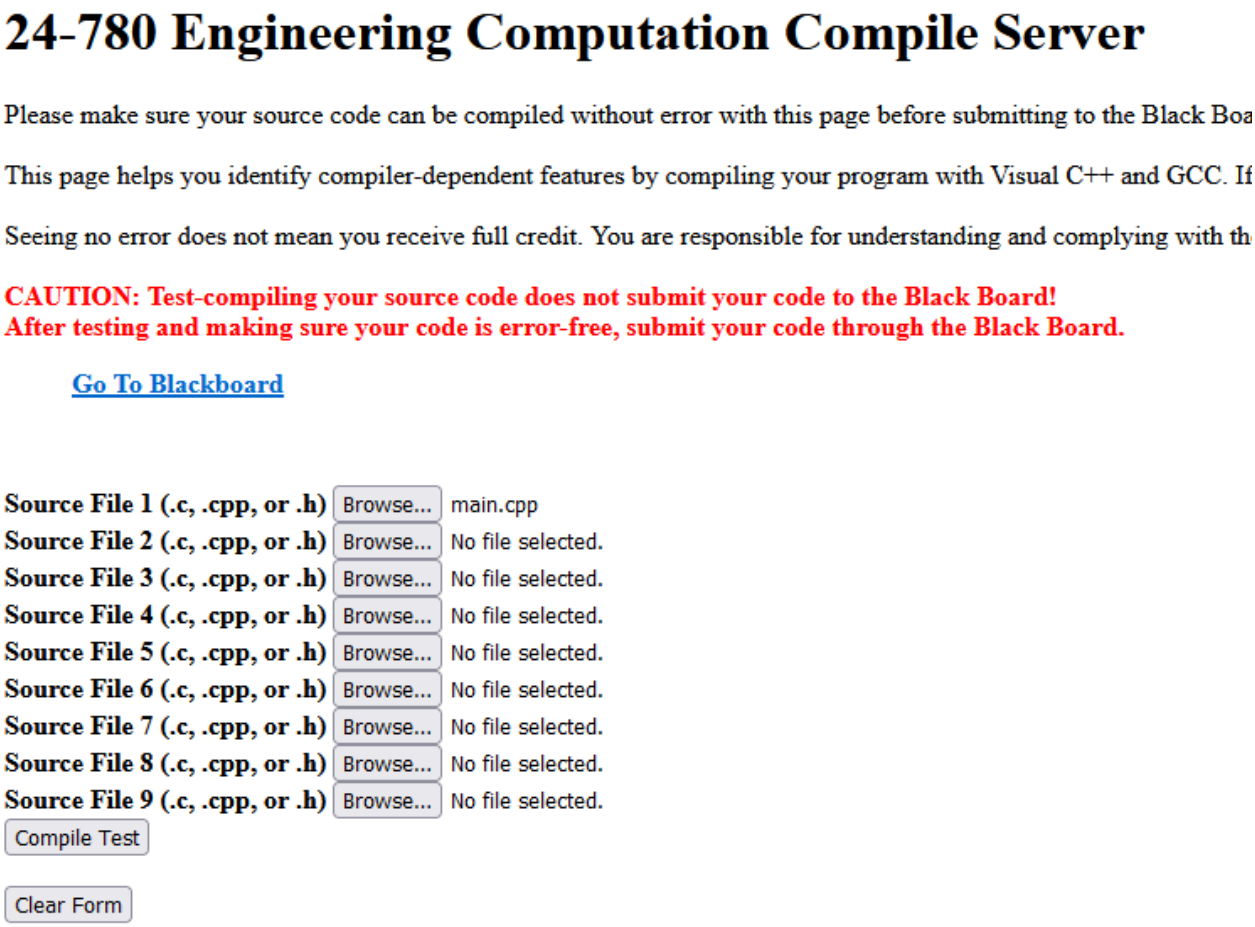

Fig. 1: Make sure to test your source code!