24783 Advanced Engineering Computation: Problem Set 1

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

In this problem set, you set up programming tools called Git and CMake. You also practice using the command line.

1 Install Git and CMake

Follow the instruction of the course note and install a Git client and CMake on your computer. Read the software instruction and make it available from the command line (Terminal in macOS and Linux. PowerShell or CommandPrompt in Windows.)

2 Change Your Git Password

Using a web browser (Chrome, Firefox, Edge, Safari, or any modern browser should work), log on to: https://ramennoodle.me.cmu.edu/Bonobo.Git.Server Your Git account name is your Andrew ID, and the initial password is also your Andrew ID. Once log on, click on your username on the top-right corner of the browser window, and change the password.

3 Set Up Working Directory

Set up your working directory.

You can create your working directory anywhere in your computer. Let's assume you create a directory called *24783* under your home directory. Home directory is typically *C:\Users\username* in Windows, */Users/username* in macOS, and */home/username* in Linux, where *username* is a local user name. A local user name can be (obviously) different from your Andrew ID and Git account.

3.1 Create Your Working Directory

First open Terminal or Power Shell. In Windows, make sure to choose "Developer Power Shell for VS20??", or you won't be able to use C++ compiler from there.

Then type:

```
cd ~
```

This command moves the current working directory to your home directory.

Next type:

```
mkdir 24783
```

This command creates a directory called 24783 under the current working directory. This directory will be your working directory. If you want to name it differently, you can choose any name.

Then type:

```
cd 24783
```

to move the current working directory to the directory you created.

3.2 Check Out Git Repositories

Then you check out two Git repositories. One is *course_files* where you find problem sets documents, base code, and samples shown in class. The other is the repository for submitting your assignment solutions, which has the same name as your Andrew ID.

Type the following commands:

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

You will be asked to enter your Git account and password. If the above command fails, try:

```
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
```

Adding yourAndrewID@ before the server name will force git to use specified Git account. If you have already using Git with a different Git server such as GitHub, git may try to use cached Git account, and may give an authentication failure, in which case specifying Git account name should solve the problem.

If you still get an error message, make sure you are doing it from the CMU network. If you are not on campus, you need to use VPN. See CMU computing services web site for how to set up your VPN. https://www.cmu.edu/computing/services/endpoint/network-access/vpn/how-to/

If you still get an error, try the following commands instead (make sure to replace *YourAndrewID* part with your Andrew ID.):

```
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
```

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
    └── yourAndrewID
```

4 Make a CMake Project

Copy source files for the Cannon-Ball game from 24-780 and fssimplewindow library to your working directory by typing the following command:

```
cp -r ~/24783/course_files/ps1 ~/24783/yourAndrewID/.
```

Adding "/." at the end of the destination location is a technique for avoiding a copy to a wrong location in case you misspell the destination directory.

Your task is to write three CMakeLists.txt files:

- /24783/yourAndrewID/ps1/CMakeLists.txt
- /24783/yourAndrewID/ps1/simplewindow/CMakeLists.txt
- /24783/yourAndrewID/ps1/cannonball/CMakeLists.txt

The first CMakeLists.txt must include *add_subdirectory* commands for simplewindow and cannonball. Also make sure to enable C++11. File and directory names are case sensitive. It is no longer Windows world. To keep CMake happier, you may also have *cmake_minimum_required* and *project* commands, but not required.

The second CMakeLists.txt is for the simple-window library. The library name must be "simplewindow", no hyphen between simple and window, all small letters. See the lecture note and cut & paste accordingly. Not complying with the naming rule will be the basis for automatic 30% penalty.

The last CMakeLists.txt is for an executable for the Cannon-Ball game. Make sure to add MACOSX_BUNDLE keyword to make it available for macOS environment. The executable name must be "cannonball" with no hyphen between cannon and ball, all small letters. Not complying with the naming rule will be the basis for automatic 30% penalty.

Rememeber, out TAs will grade using a grading script. Not complying with the naming rule will stop the script and will force TAs to manually look into your submission, which will clearly make TAs unhappy. You won't want to be graded by an unhappy TA!

5 Building the CMake Projects

Make sure to use *out-of-source* build, which means that the build directory is completely outside of the source tree. Make sure you won't submit any files created during the build process, which will be very difficult if you end up doing *in-source* build.

You can follow the steps below:

```
cd ~/24783
mkdir build
```

```
cd build
cmake ../yourAndrewID/ps1
cmake --build . --config Release
```

Make sure you are not getting errors.

To run from there, type the command:

```
cannonball/Release/cannonball.exe
```

in Windows, or

```
cannonball/cannonball
```

in Linux, other

```
cannonball/cannonball.app/Contents/MacOS/cannonball
```

Don't type all the letters. Make use of completion by pressing Tab key.

6 Submission

Lastly, you need to submit using git. What you need to do are two things: (1) add files to git's control, and then (2) send to the git server.

6.1 Add Files to git's control

In this case, you want to add all the files under ps1 subdirectory. To do so, type:

```
git add ~/24783/yourAndrewID/ps1
```

This command will add ps1 directory and all files under the subdirectories.

6.2 Send to the Git Server

In Git, sending files to the server is a two-step process. The first step is local commit. You can do it by:

```
git commit -m "Problem Set 1 solution"
```

The message can be anything, but it is recommended to type something meaningful, at least you can see what changes you made to your repository.

Local commit is just local. Git server does not know about any local commit unless the commit is sent (or pushed) to the server. To do so, type:

```
git push
```

Make sure to do it in the CMU network. If you are working from home (probably most likely), use VPN to connect to the CMU network.

7 Verification

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

For example, 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.