24783 Advanced Engineering Computation: Problem Set 3

(*) 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
```

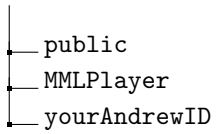
You need to replace "yourAndrewId" with your Andrew ID. You'll be asked to type in credentials.

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
```



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/ps3 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/ps3
```

Once you add ps3 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 ps3 sub-directory, so that:

- it enables C++11 features,
- it includes two sub-directories, flythrough and jump, and
- 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(../../public/MMLPlayer/ym2612 ${CMAKE_BINARY_DIR}/ym2612)
add_subdirectory(../../public/MMLPlayer/mmlplayer ${CMAKE_BINARY_DIR}/mmlplayer)
```

3.2 CMakeLists.txt Files for Executables

Write a CMakeLists.txt file for each flythrough and jump sub-directories, just like you did in Problem Set 1. The executable names must be flythrough and jump. In windows, the file names will become flythrough.exe and jump.exe in the build directories, but do not add .exe in CMakeLists.txt.

This time, simple-window library is taken from the public libraries. The name of the library is fssimplewindow.

When building with cmake, add options "-target flythrough jump" so that it will skip compiling un-related executables and binaries.

3.3 Add to Git's Control

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

4 Refactor Fly-Through and Jump Programs in Event-Driven Programming Style

Let's practice event-driven programming style one more time.

Refactor Fly-Through and Jump programs in event-driven programming style as introduced in class. The main function should look like:

```
int main(void)
{
  FsOpenWindow(0,0,800,600,1);

  ApplicationMain app;
  while(true!=app.MustTerminate())
  {
    app.RunOneStep();
    app.Draw();
  }

  return 0;
}
```

Therefore, you need to write ApplicationMain class, which has at least MustTerminate, RunOneStep, and Draw member functions.

5 Build the CMake Projects and Run

Make a build directory outside of your source tree and use cmake to set up and build the program.

6 Test Your Code on the Compiler Server

Test your source files (.cpp and .h files) on the compiler server. Some assignment may not require .h files. You do not have to test files that you don't make modifications. The files you need to test are the ones you write or modify.

We have four compiler servers:

- http://freefood1.andrew.cmu.edu:24780
- http://freefood2.andrew.cmu.edu:24780
- http://freefood3.andrew.cmu.edu:24780
- http://freefood4.andrew.cmu.edu:24780

Compiler servers are accessible from within CMU network only. To access from the outside network, use VPN to connect to the CMU network.

Make sure you don't see red lines when you select your files and hit "Compile" button on the server.

We have multiple servers to make it less likely that all of them need to shut down for maintenance. If do not have to test on all of the servers. You need to make sure that your code passes on one of the servers.

7 Submit

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.

7.1 Add Files to git's control

In this case, you want to add all the files under ps3 subdirectory. To do so, type:

```
git add ~/24783/yourAndrewID/ps3
```

This command will add ps3 directory and all files under the subdirectories.

7.2 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 2 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.

You can re-submit (commit and push) your solution as many times as you want with no penalty before the submission due.

8 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.

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.

9 Bonus Point

Use MML Player library to add BGM to Jump. +2 points if you copy and paste one of the sample music. +5 points if you write MML on your own.