24-783 Problem Set 3

One-week assignment. Please see Canvas for the deadline.

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

In this problem set, you will practice (one more time):

- compiling your CMake projects using the libraries included in the public repository, and
- converting a polling-based program (FsSimpleWindow) into an event-driven program (FsLazyWindow).

Before starting, update the public repository and course_files by typing:

```
svn update ~/24783/src/course_files
svn update ~/24783/src/public
```

I am assuming that you have the same directory structure as we used in the first assignment.

START EARLY!

CMake and SubVersion both must be relatively new concepts for you. Try the problems early, and consult with a TA or the instructor if you have difficulties in following the instructions.

TEST YOUR PROGRAM WITH THE COMPILER SERVER:

We have four compiler servers for checking your source files for errors:

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

Make sure your final-version source files pass the compiler server. Select your .cpp files (and .h files if appricable) and press “Compile Test” button. If you see any red lines in the output, your program needs to be corrected.

PS3-1 Set up projects for the Fly-Through and Orbit.

You first create projects for the two projects for the event-driven style version of fly-through and orbit.

1. In the command line window change directory to:
   ~/24783/src/*yourAndrewId*
2. Create a sub-directory called:
   ps3
3. Then, create the following two sub-directories:
   ps3/flythrough
   ps3/orbit
   File/Directory names are case sensitive. These two directories are for the two projects you are going to create. Use mkdir command to do this.
4. Copy the application template file:
   ~/24783/src/public/src/fslazywindow/template/main.cpp
   to the two directories you created. (Make two copies, one in ps3/flythrough, and the other in ps3/orbit)
5. In ps2/flythrough, write a CMakeLists.txt, which has an executable target called "flythrough" from main.cpp, and the target link fslazywindow library.
6. In ps2/orbit, write a CMakeLists.txt, which has an executable target called "orbit" from main.cpp, and the target link fslazywindow library.
7. In the top-level CMakeLists.txt (~/24783/src/CMakeLists.txt), add sub-directories:
   yourAndrewId/ps3/flythrough
   yourAndrewId/ps3/orbit
   If you haven't created your top-level CMakeLists.txt yet, make sure you have the following lines upfront:
   ```
   set(CMAKE_CXX_STANDARD 11)
   set(CMAKE_CXX_STANDARD_REQUIRED ON)
   add_subdirectory(public/src)
   ```
8. Run cmake and then build "**flythrough**" and "**orbit**" projects. Project names are case sensitive, do not add hyphen or any other characters. Run the programs and make sure you see a blank window from each project.
9. Add files under:
   ~/24783/src/*yourAndrewId*/ps3
   to the SubVersion repository. If you have used "svn mkdir" command in Step 2, you need to add two main.cpp and CMakeLists.txt files (four files in total) in this step. If you have used "mkdir" command in Step 2, you need to add ps2 directory.
10. Commit your files to the SubVersion server.
11. Verify that your files are correctly committed.

PS3-3 Convert Fly-Through and Orbit to an Event-Driven style.

The base codes for the Cannon-ball game and Bouncing-ball programs are in:

~/24783/src/ps2/flythrough/basecode.cpp
~/24783/src/ps2/orbit/basecode.cpp

Convert them to the event-driven style and write them in:

~/24783/src/*yourAndrewId*/ps3/flythrough/main.cpp
~/24783/src/*yourAndrewId*/ps3/orbit/main.cpp

Run CMake and make sure your program works as expected. It is recommended to commit your files after each step of conversion so that you can go back to the previous version if something does not go well.

Commit your files to the server.

Also, check out your submitted files in a different location and make sure all files are submitted, and the file contents are the ones you wanted to be graded.

Since you are able to re-checkout and verify your submission, we do not accept an excuse that you submitted a wrong file(s).

Did you add **MACOSX_BUNDLE** keyword in your add_executable (for graphical programs)?

TEST YOUR PROGRAM WITH THE COMPILER SERVER:

We have four compiler servers for checking your source files for errors:

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

Make sure your final-version source files pass the compiler server. Select your .cpp files (and .h files if appricable) and press "Compile Test" button. If you see any red lines in the output, your program needs to be corrected.

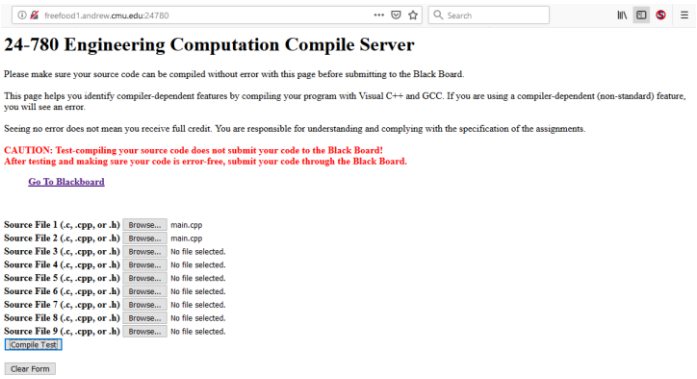
Also, check out your submitted files in a different location and make sure all files are submitted, and the file contents are the ones you wanted to be graded.

Since you are able to re-checkout and verify your submission, we do not accept an excuse that you submitted a wrong file(s).

Also test the C++ files (.cpp and .h files) with the compiler server and make sure you do not see any red lines in the test-results page. Read the instructions in the following page.

(1) Open one of freefood1 to freefood4.andrew.cmu.edu:24780

(2) Browse and select <u>your</u> .cpp and .h files.



(3) Click on Compile Test.

(4) Scroll down and make sure you don't see any red lines.