

## 24-783 Problem Set 2

Due: One-week assignment. Please see Canvas for the exact 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:

- 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
cd ~/24783/src/public
git pull
```

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.

### PS2-1 Set up projects for the Cannon-ball game and Bouncing Ball

You first create projects for the two projects for the event-driven style version of Cannon-ball game and Bouncing Ball.

1. In the command line window change directory to:  
    ~/24783/src/*yourAndrewId*
  2. Create a sub-directory called:  
    ps2
  3. Then, create the following two sub-directories:  
    ps2/cannonball  
    ps2/bounce
- 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 ps2/cannonball, and the other in ps2/bounce)
  5. In ps2/cannonball, write a CMakeLists.txt, which has an executable target called “cannonball” from main.cpp, and the target link fslazywindow library.
  6. In ps2/bounce, write a CMakeLists.txt, which has an executable target called “bounce” from main.cpp, and the target link fslazywindow library.
  7. In the top-level CMakeLists.txt (~/24783/src/CMakeLists.txt), add sub-directories:  
    *yourAndrewId*/ps2/cannonball  
    *yourAndrewId*/ps2/bounce

If your top-level CMakeLists.txt contains a project called bounce (and cannonball as well maybe) from the previous lectures, remove that sub-directory first, and then add the two new sub-directories. **You write your own top-level CMakeLists.txt for compiling and testing the programs, but for grading the script will create a top-level CMakeLists.txt, which assumes this directory structure. Make sure to comply with this directory structure.** 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 “**cannonball**” and “**bounce**” 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*/ps2  
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.

## **PS2-2 Convert Cannon-ball game and Bouncing ball to an Event-Driven style.**

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

~/24783/src/ps2/cannonball/basecode.cpp

~/24783/src/ps2/bounce/basecode.cpp

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

~/24783/src/*yourAndrewId*/ps2/cannonball/main.cpp

~/24783/src/*yourAndrewId*/ps2/bounce/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)?

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 your .cpp and .h files.

The screenshot shows the web interface of the 24-780 Engineering Computation Compile Server. The browser address bar shows the URL: freefood1.andrew.cmu.edu:24780. The page title is "24-780 Engineering Computation Compile Server". Below the title, there is a warning: "Please make sure your source code can be compiled without error with this page before submitting to the Black Board." and "This page helps you identify compiler-dependent features by compiling your program with Visual C++ and GCC. If you are using a compiler-dependent (non-standard) feature, you will see an error." A red warning box states: "CAUTION: Test-compiling your source code does not submit your code to the Black Board! After testing and making sure your code is error-free, submit your code through the Black Board." Below this, there is a link "Go To Blackboard". The main section contains a table with 9 rows, each representing a source file. The first row is "Source File 1 (.c, .cpp, or .h)" with a "Browse..." button and the file name "main.cpp". The other rows are "Source File 2" through "Source File 9", each with a "Browse..." button and the text "No file selected." At the bottom of the table is a "Compile Test" button. Below the table is a "Clear Form" button.

| Source File                     | Browse... | File Name         |
|---------------------------------|-----------|-------------------|
| Source File 1 (.c, .cpp, or .h) | Browse... | main.cpp          |
| Source File 2 (.c, .cpp, or .h) | Browse... | main.cpp          |
| Source File 3 (.c, .cpp, or .h) | Browse... | No file selected. |
| Source File 4 (.c, .cpp, or .h) | Browse... | No file selected. |
| Source File 5 (.c, .cpp, or .h) | Browse... | No file selected. |
| Source File 6 (.c, .cpp, or .h) | Browse... | No file selected. |
| Source File 7 (.c, .cpp, or .h) | Browse... | No file selected. |
| Source File 8 (.c, .cpp, or .h) | Browse... | No file selected. |
| Source File 9 (.c, .cpp, or .h) | Browse... | No file selected. |

[Compile Test](#)

[Clear Form](#)

(3) Click on Compile Test.

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

The screenshot shows the "Compile-Test Result" page. The browser address bar shows the URL: freefood1.andrew.cmu.edu:24780/cgi-bin/receivefile.exe. The page title is "Compile-Test Result". Below the title, there is a message: "CGI Revision:6092" and "Session=1548210833118509109". The "Received Files:" section shows a list of files: "main.cpp" and "main.cpp". Below this, there is a section titled "Preliminary Check for Non-Standard features >>". Under this section, there are two "Checking main.cpp" sections. Each section has an "Included Files" list: "stdio.h", "stdlib.h", "time.h", "math.h", and "float.h".

**Compile-Test Result**  
CGI Revision:6092  
Session=1548210833118509109

**Received Files:**

- main.cpp
- main.cpp

**Preliminary Check for Non-Standard features >>**

**Checking main.cpp**  
**Included Files**

- stdio.h
- stdlib.h
- time.h
- math.h
- float.h

**Checking main.cpp**  
**Included Files**

- stdio.h
- stdlib.h
- time.h
- math.h
- float.h

