Due: 11/30 (11:59PM)

Requirements:

- Develop a simple 3D shooting game using OpenGL. Name your source code `hw3.cpp`. The program should meet the following requirements:
  - This program builds on your flying teapot program (hw2). Therefore, make sure your hw2 program meets *all* the requirements of hw2 before you begin.
  - Create and add a 3D cannon such that it aims forward from the viewer towards the far plane (Fig. 1). It is okay to show only the tip (firing end) of the cannon.
  - The cannon can turn left and right, and up and down (yaw and pitch movements) like a real cannon. The user controls left-right rotation with right-left arrow keys, and up-down rotation with up-down arrow keys. The rotation in either direction should be clamped to a certain range (say, 120 degrees) so the cannonball would stay within the view volume.
  - The cannon should fire a cannonball (sphere) every time the user presses the space bar. The ball should fly in the direction of the current orientation of the cannon. Once the ball goes through any wall and thus no longer visible, delete it from the scene to avoid clogging the memory. Therefore, the system will store no more than a few (fired) balls at any given time.
  - During the game, display text at the top of the screen that shows how many teapots are currently left (Fig. 1).
  - When the cannonball hits one of the teapots (*collision detection* needed here), the hit teapot must immediately disappear from the scene (and so must this cannonball). The text must also be updated accordingly with a new number (one less teapots left).
  - When the last teapot is hit, the game is over and the text should be replaced with “Continue? (Y/N)”. If the user presses ‘y’, start the game over. If ‘n’, exit program.
  - The background walls (all 5 of them) must be texture mapped, although Fig. 1 doesn’t show it. You can use any *tileable* texture image of your choice, but make sure the image file size is less then 1MB (with no bigger than 512×512 pixel resolution). Do not use multiple texture images.
  - If the user presses `'q'` at any time (even during the game), exit program.
  - If the animation is too fast on your computer, slow it down to a level where each teapot’s movement and the cannonball’s movement are clearly recognizable.
  - Make sure there is no sudden jump or discontinuity in the animation. The whole sequence of transformations must be smooth and continuous.

If you are a graduate student:

- The program should meet the following requirements as well:
  - Every time the cannonball fires, rapidly push the cannon backwards a bit to simulate the action-reaction effect. After this quick push back, the cannon should immediately come back to its original position before it can fire a new ball (see `https://youtu.be/2NnUve9xS10`). Try to make this movement as realistic as possible.
  - Whenever a teapot is hit, the hit teapot must fly out of the scene in a random direction towards far plane (so the depth should increase) while randomly spinning. Once the teapot goes past the wall, remove it from the scene.

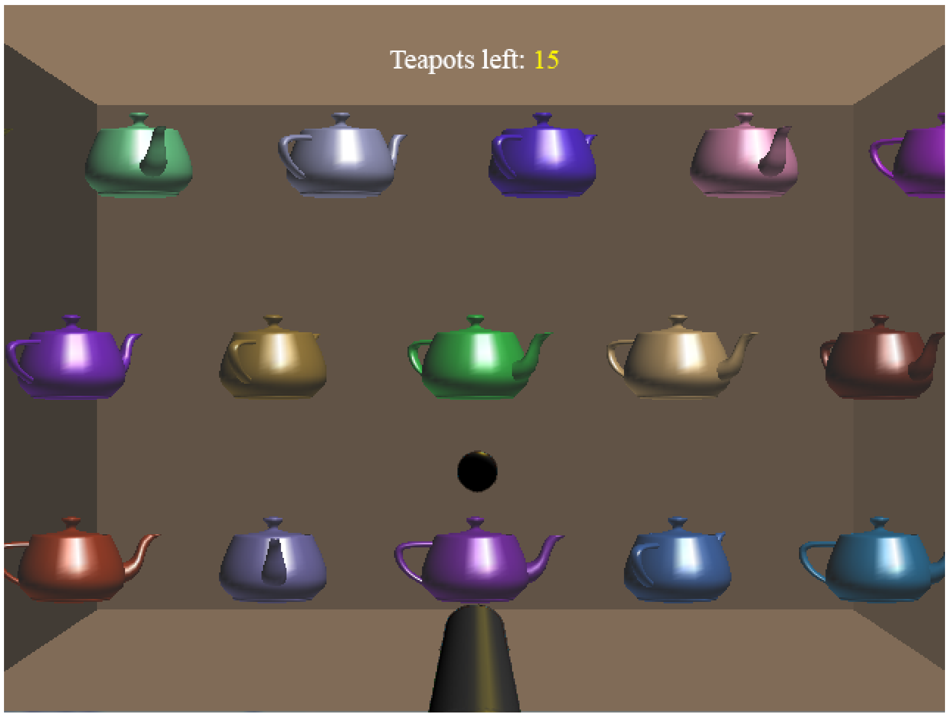

Figure 1: Cannonball shooting game

What to submit:

- Submit only your **source files (.cpp, .h)** and **texture file (.bmp)** that are needed for compilation.

How to submit:

- Use Canvas Assignment Submission system to submit your source files.

Policy

- Do all the assignments on `Visual C++` using C++ and OpenGL.
- Source code (`.cpp` files) should contain enough comment lines so that other people could easily follow your code. Insufficient comments could lead to loss of points.
- Each source code must be **complete** so it can 'run' without error when compiled as it is.
- The contents of each source code file (`.cpp` files) should begin with the following comments:

```
// your name (e.g., John Smith)
// cs4410 hw# (e.g., cs4410 hw3)
// date dd/mm/yyyy (e.g., 09/04/2017)
```

- Non-compilable program will get almost no credit (e.g., executable code is not produced due to compiler errors).
- Non-working program will get almost no credit (e.g., the executable is terminated immaturely due to run-time errors).
- Copying someone else's program is strictly prohibited. If identical (or nearly identical) programs are found among students, every student involved will get automatic 0 for the assignment. The same goes for copying source code acquired from the internet.