CSCI 437 - Homework: Pong

Out: September 4th
Due: September 17th, noon

Goal

Reimplement the classic arcade game 'pong' using SFML. More information on the game 'pong' can be found at http://en.wikipedia.org/wiki/Pong

Core Requirements

The following core requirements should be implemented (the weight of each requirement is provided for your convenience):

- The game should be played in an 800x600 window. Changing the size of the window should be disabled **(10%)**.
- A well-defined game loop that coordinates game timing and event management **(20%).**
- A single player version should be implemented where the user controls a single paddle **(15%)**, and the computer the other paddle **(15%)**. The paddle can be moved by keyboard or mouse (one method of control is sufficient). The paddle of both the player and the AI should move at a constant speed.
- A ball should bounce of the top and bottom edge of the playfield and the direction of the ball should change to the reflected direction **(15%)**. To add an element of uncertainty, a small random perturbation could be added/subtracted from the reflected direction **(5%)**.
- A player scores when the ball hits the edge of the competitors' side **(5%)**.
- The game starts with both players having zero points, and ends when one player reaches 11 points. A message should be shown to indicate whether the player won or lost. The player should have the option to restart the game or quit **(10%)**.
- The current score should be shown in the window **(5%)**.

Bonus Features

In addition to the above core requirements, additional bonus points can be obtained by implementing one or more of the bonus features listed below.

- Additional support for correct resizing and maximization of the window **(10%)**. As well as pausing of the game when the window is minimized **(5%)**.
- Add sounds when hitting a wall, pad, or when the ball is leaving the screen **(15%)**.
- Make the pad move slower when just starting to move, and accelerate the longer the pad is moving in the same direction up to some maximum speed **(10%)**.

- Take the speed and direction of the pad motion in account when computing the reflected direction **(15%)**.
- Add obstacles **(10%).**
- User selectable difficulty by setting the game speed (i.e., more difficult == faster game) **(10%)**.
- Add a title screen with selectable options (e.g., which pad to play with, difficulty level, etc...) **(15%)**.
- Add randomly placed power-ups that can be enabled to hitting it with a ball (e.g., hold ball on pad, safety net behind pad, double size pad, etc..) **(10%).**
- Something else creative **(?%)**

Practical

This homework should be solved using the SFML library. You can either installed a precompiled version (if you OS is supported) or compile the library from source code from http://www.sfml-dev.org SFML is preinstalled on the lab machines. This homework also relies on CMake (http://www.cmake.org) to provide cross platform Makefile support (CMake supports both Linux and Windows). Note that depending on how much you have used of your disk-quota on the departmental lab machines, you might not be able to build and compile pong in your home directory. In that case, you can use the temporary (and local) 'scratch' directory. However, you will need to back up your code regularly (using a Git or SVN repository is recommended).

A skeleton-code that provides a basic SFML-based 'Hello World' program can be found on Blackboard. It is highly recommended to start from this skeleton code; you must use the CMake configuration. To compile the skeleton code (see the included README.TXT), create a subdirectory 'Debug', and from within this directory run 'cmake ..' to generate the Makefile. SFML might not be found if it is not installed in a standard location. To aid cmake in finding the library you will need to update the LIB and INCLUDE variables (the following example assumes SFML is installed in a directory SFML in your home; change accordingly):

```
export LIB=$LIB:~/SFML/lib
export INCLUDE=$INCLUDE:~/SFML/include
```

You will need to set these environment variables each time you run 'cmake'. Normally, you do not need to rerun cmake, unless in the situation described below. The CMake files in this skeleton have been set up to simplify adding files without the need to alter the CMake files:

- If you want to add a new executable, just add the corresponding '.cpp' file in the 'bin' subdirectory, and **rerun** 'cmake ..' to update the Makefiles.
- If you want to add a source file, just add the corresponding '.cpp' file in the 'src' subdirectory, and **rerun** 'cmake ..' to update the Makefile.
- If you want to add a header file, just add the corresponding '.h' file to the 'include' subdirectory. You do not need to update the Makefile by rerunning cmake.

- Adding an external library requires more work – you will need to add a FindPackage entry in the CMakeLists.txt file. Please refer to how SFML is added in the included CMakeLists file as a template.

In case you want to remove all compiled files, just remove the 'Debug' directory. Based on the directory name, cmake will generate a makefile to compile your code in either Debug or Release mode. In Debug mode, debug information is included in your executable, and certain checks are included. Release mode, runs faster, but it is advised to on use this when you know your code works. To compile in release mode, just create a directory 'Release', and run 'cmake ..' from within this directory.

Submission Details

The following should be emailed to "ppeers@cs.wm.edu" by **September 17th, noon:**

- A report (max 1 page **PDF**) named <firstname>-<lastname>.pdf that contains the following information:
  - **Time required to finish the core requirements this homework**
  - Time spend on the bonus features (if any).
  - A list of any external libraries (besides STL or Boost) that you used and why.
  - A list of core features that do not work correctly.
  - A list of implemented bonus features, indicating whether they work or not.
- An archive of your code/data. To prepare, run the following commands in the root of your source (pong):
  - rm –rf Debug
  - rm –rf Release
  - cd ..
  - tar czvf <firstname>-<lastname>.tgz pong

Remarks

- The estimated time required for the basic requirements is 16 hours, excluding time to master C++.
- This homework accounts for **2/9th** of the total for this course. Additional bonus credits (**max 1/9th**) can be obtained by implementing the bonus features.
- The grading for this homework will be as follows:
  - 10% whether or not your code compiles (without excessive changes to the compile process).
  - 50% on correctness and playability. Does the ball move correctly? Can the pad be moved correctly? Are scores being tracked? Does the game end? The percentages listed in the 'core requirements' and 'bonus features' pertain to this component of the grade.
  - 30% on code quality. Is the code well documented? No memory leaks? Clean OO design?

    - 10% on the report.
- This is not a group project; this homework should be solved alone.
- **You are not allowed to use existing SFML pong solutions found online – your code will be compared to available online solutions, and if a match is found, you will receive 0 credits.**
- You are allowed to use external libraries, but you will need to justify their use in your report. In general, external libraries should be used sparingly. If in doubt, contact the lecturer.
- Late policy: score -= ceil(hours_late / 24) * 10. Negative scores are possible. An extension needs to be requested at least 48 hours before the deadline. Each extension request needs to clearly justify why the extension is needed.