# 24-780 Engineering Computation Problem Set 06

You need to create a ZIP file (It may appear as a compressed folder in Windows) and submit the ZIP file via the 24-780 Canvas. The file name of the ZIP file must be:

PS06-YourAndrewID.zip

For example, if your Andrew account is *hummingbird@andrew.cmu.edu*, the file name must be:

PS06-hummingbird.zip

If your ZIP file does not comply with this naming rule, you will automatically lose 5% credit from this assignment. If we are not able to identify who submitted the file, you will lose another 5% credit. If we finally are not able to connect you and the submitted ZIP file, you will receive 0 point for this assignment. Therefore, please make sure you strictly adhere to this naming rule before submitting a file.

The ZIP file needs to be submitted to the 24-780 Canvas. If you find a mistake in the previous submission, you can re-submit the ZIP file with no penalty as long as it is before the submission deadline.

Notice that the grade will be given to the final submission only. If you submit multiple files, the earlier version will be discarded. Therefore, if you re-submit a ZIP file, the ZIP file MUST include all the required files. Also, if your final version is submitted after the submission deadline, late-submission policy will be applied no matter how early your earlier version was submitted.

Make sure you upload your Zip file to the correct location. If you did not upload your assignment to the correct location, you will lose 5%.

The ZIP file needs to include:

- C++ source file of your program (ps6-1.cpp and ps6-2.cpp)

Submission Due: 10/10 (Tue) 23:59

# START EARLY!

Unless you are already a good programmer, there is no way to finish the assignment overnight.

**PS6-1 Word search (30 points)**

Write a C++ program (**ps6-1.cpp**) that takes a file name and a keyword as input from the console window and finds how many times the word appears in the file. You can only consider exact match, you don't have to consider case sensitivity. I.e., "Word" and "word" are considered different words.

You can assume that the file is an ASCII text file and each line is shorter than 256 letters. Also, assume that words are separated only with space, tab, period, or comma. In the parsing program shown in class assumed words are separated by space, tab, and comma only. For this assignment, your program also needs to recognize period as a separator. The comparison must be done word by word. Therefore, if you search for "clear", your program must not count "clearance" as one occurrence although "clear" matches the part of "clearance." The result needs to be printed on the console window. (Hint: You can start from the word-numbering program that we used in class.)

(Sample running image)

```
Enter Input File Name>sampleTextFile1.txt
Enter Keyword>solution
The keyword appears 9 times.
```

Remember that the text file must be placed in the working directory. (Or, you can specify full-path name to the file in the console window.)

**Make sure your program can be compiled with no error in one of the compiler servers. Don't wait until the last minute. Compiler servers may get very busy minutes before the submission deadline!**

**PS6-2 Break-down game (70 points)**

In PS6-2, you write a break-down game. Your program opens an 800x600 window. The user controls a racket by mouse. The racket coordinate follows the mouse coordinate (racketX=mouseX and rackety=mouseY). However, if the Y coordinate of the mouse cursor is less than 500 or greater than 600, the racket coordinate will be forced to 500 or 600 respectively. (The racket will not move higher than y=500 or below y=600). The racket will span 50 pixels to the left and right of racketX, and from racketY to racketY+20 pixels vertically.

A ball is initially located at (100,200), and the initial velocity is (vx,vy)=(8,8). The ball moves by (vx,vy) pixels in each step. The ball bounces on top, left, and right wall. Also, it bounces when it is hit by the racket.

At the beginning, you have 16 blocks. Each block is 100 pixels wide and 20 pixels high. The top left corner of the most top-left block is (0,80). And blocks are arranged eight blocks in each row and two blocks each column. There is no void space between blocks initially.

In each step, your program must sleep 25 milliseconds.

When the ball hits a block, the block is destroyed and the ball velocity in Y axis is inverted. (To make it simple, we invert ballVy even when the ball hits the side of the block.)

If all the blocks are destroyed, your program must print “You win!” and the program must terminate.

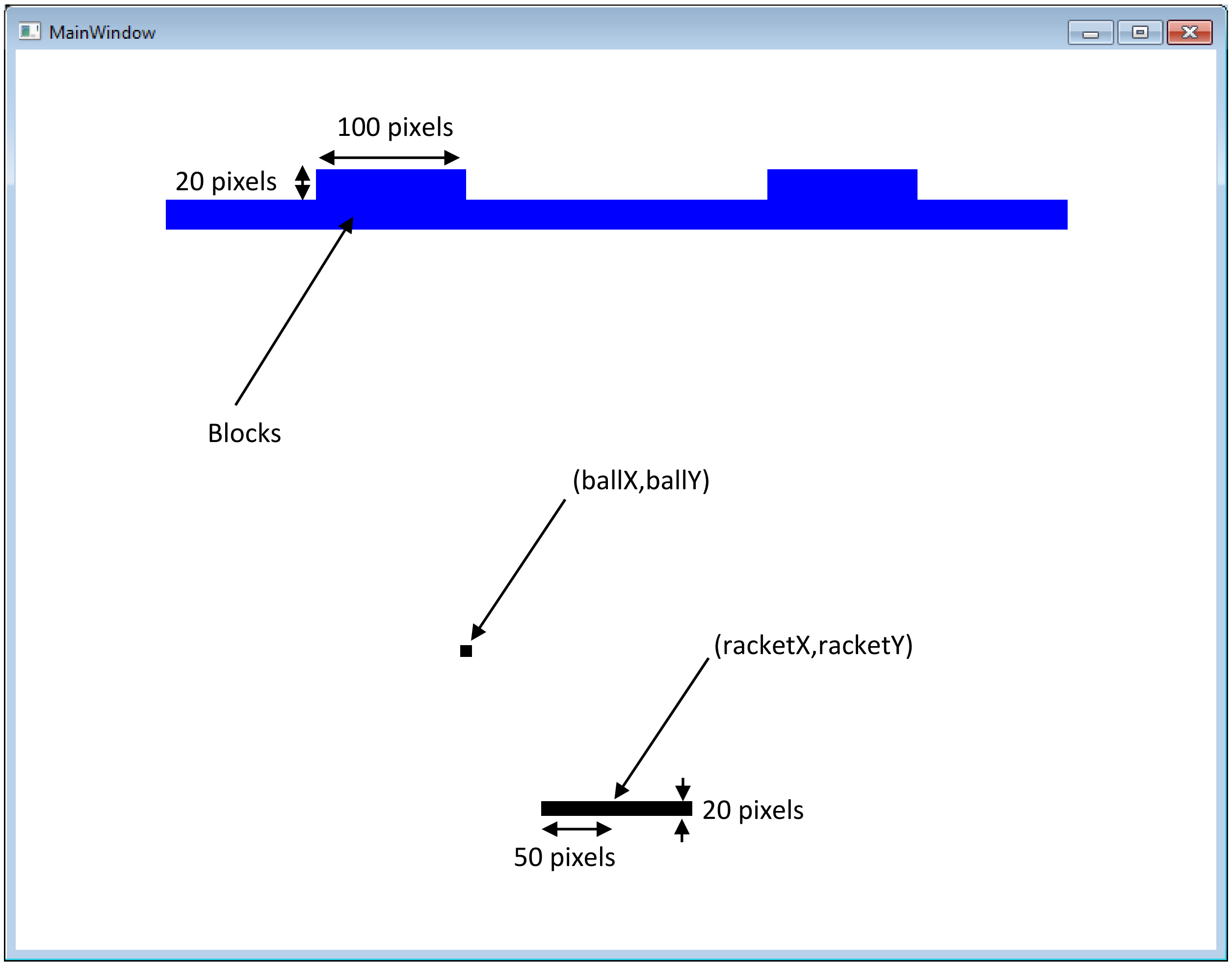
If the ball goes below the bottom edge of the window, your program must print “Miss!” and then the program must terminate.

The background of the window must be white, and the racket and the ball must be black. You can choose colors of the blocks as long as the blocks are clearly visible.

Don’t get scared. You can do it step by step. It is NOT much more difficult than the shooting game. Recommended steps are:

1. Write racket control and show it on the window.
2. Write ball state variables and move it.
3. Write ball bouncing.
4. Add data structure for blocks, and draw them.
5. Check ball-block collision.
6. Check for the winning condition.

<u>**Make sure your program can be compiled with no error in one of the compiler servers. Don’t wait until the last minute. Compiler servers may get very busy minutes before the submission deadline!**</u>


MainWindow
100 pixels
20 pixels
Blocks
(ballX,ballY)
(racketX,racketY)
20 pixels
50 pixels