CS 117: Introduction to Computer Science	**Winter 2006**
Handout 10: Final Project (Due: Wednesday, 15 March 2006, 5:00p)	*27 February 2006*

It's never just a game when you're winning.
— George Carlin (b. 1937)

The Assignment

The project can be done individually or in a partnership of your choice. Unlike with pair-programming assignments from the course so far, you are not both required to work on all aspects of the project, but you *are* all expected to do your fair share. I will expect projects done in a partnership to be twice as ambitious and impressive as projects done invidually!

Your mission: create a computer game that incorporates some level of computer intelligence. Some notes:

1. Your game can be entirely text based, or you can do some graphics with Canvas or EzImage if you like. No graphics are required, though; I'm much more interested in seeing how much you've learned about how to implement the algorithms that you want to use.
2. Some possible games to implement: 3D Tic-Tac-Toe, Dots and Boxes, Battleship, Go Fish, etc. Another possibility is *Bagels*, which is described at the bottom of the assignment.
3. The requirement of "some level of computer intelligence" means this: the computer should play against you or manage your game in some kind of intelligent way. You don't need to the computer to be a world class player at your game (or even close!)—that's what CS 327 (Artificial Intelligence) is for—but you should make an effort on this front.
4. Let me know if you have questions about the rules or suitability of a particular game. I recommend that you send an email or talk to me about your choice of project to make sure that you haven't picked something too easy or too hard. The biggest trap on this assignment that students often fall into is picking something too complex and not having the time to finish it. Consult with me.

Grading

The criteria on which your projects will be graded are the following, in decreasing order of importance:

1. **Compilation.** If the program doesn't compile, I won't grade it.
2. **Correctness.** How cleanly does the program run? Is it bug free? Does it crash?
3. **Complexity.** How complicated is your game? How complex is the computer intelligence?
4. **Style.** Is the code well organized, readable, and well documented?

Note again: if your program won't compile, then you get a zero. But beyond that, correctness is the most important factor. A program that runs correctly, error-free, and achieves the above specifications will get a better grade than an incredibly smart chess program that crashes and produces unreadable output. Of course, if I'm comparing side by side two programs that run perfectly, the one that accomplishes more will receive the better grade.

To reiterate: It is most important to submit something that works. Set modest goals and achieve them first, then enhance your program later. Only work on getting computer intelligence working after you have the basic game working with human player(s).

On academic dishonesty: You must submit a program that you write yourselves. You may obtain help from other students in getting ideas and in debugging your code, but you must write your own program. You may not use Java code for your game from the Internet or from other sources. You must acknowledge any sources of assistance.

Submission

Your submission is due at 5:00pm on the last day of finals—Wednesday, 15 March 2006.

In addition to submitting your Java files, you should submit the following items of documentation in file `readme.txt` that is in your directory. You may be brief.

- A description of your program and its features.
- A brief description and justification of how it is constructed (class organization, how data are stored, etc.)
- A discussion of the current status of your program—what works and what doesn't, etc.
- Instructions for running your program.

Bagels

Bagels is a two person paper-and-pencil game that is similar to but simpler than Mastermind. One person thinks of a 3-digit number, and the other person tries to guess it. The 3-digit number is not allowed to have repeated digits, but it may begin with a zero (so 012, 987, and 361 are legal, but 112 and 303 are not).

The Guesser makes a 3-digit guess. The Responder compares the guess to the actual mystery number, and responds to the guess by some combination of the words "Pico," "Fermi," and "Bagels." The Guesser keeps guessing until the guess is the mystery number. Here are the response rules:

- If the guess has no digits in common with the mystery number, then the answer is "Bagels."
- If the guess has a digit in common with the mystery number, and the common digit is in the same position in both numbers, then the responder says "Fermi" (once per match).
- If the guess has a digit in common with the mystery number, but that common digit is not in the same position in both numbers, then the responder says "Pico" (once per match).
- If there are multiple matches, all the Picos should be reported before all Fermis. (So you'd never say "Pico, Fermi, Pico"; if there are two Picos and one Fermi, you'd say "Pico, Pico, Fermi" regardless of which digit was the Fermi.)

For example, suppose the mystery number is 395. Here are a few guesses and responses:

```
246    Bagels
037    Pico
105    Fermi
309    Pico, Fermi
395    Fermi, Fermi, Fermi.
```

If you choose this project, you should write a program that will play Bagels with you, both as the Guesser and the Responder. Having the computer act as Responder is fairly straightforward. Having it act as Guesser is trickier, but fun.

Have fun!!