

CMSI 370

INTERACTION DESIGN

Fall 2008

Assignment I202

This assignment hopes to give you some exercise on direct manipulation programming, plus revisit some code from past classes.

Not for Submission

If you haven't already done so, read Chapter 6 from Shneiderman/Plaisant.

For Submission

Design and implement a direct manipulation Swing or DHTML user interface for a program that was written in a previous class (185, 186, 281, etc.). Use the interaction design metrics, guidelines, principles, and theories discussed in class so far to inform the design decisions that you make for the application.

Most likely, your original program was written in the command line interaction style. While adapting your code to a direct manipulation interaction style, use the MVC pattern to keep your original logic as intact as possible. Note that if you're doing a web interface, you'll need to port your code from Java to JavaScript as well.

You may use other interaction styles (menus, maybe even commands if applicable) as part of your overall user interface, but make sure that the primary elements use direct manipulation.

Design Report

In addition to your code, turn in a short LaTeX "design report" that discusses the decisions that you made in creating the direct manipulation user interface. Include at least the following sections:

- A description of the original program — what it did, and how the user interacted with it
- A description of your user interface design, stating the guidelines, principles, and theories that influenced your design decisions.
- A description of what changes you made to your original code, both functionally and structurally, in order to adapt it to its new user interface.

How to Turn it In

- Commit the program to CVS under `/homework/cmsi370/directmanipulation`. As usual, include a Maven project file.
- Commit your design report source files under `/homework/cmsi370/directmanipulation/doc` (note how this is a subdirectory of the program's location; plan your local copy accordingly).

Extra Credit

You will get extra credit if you build *two* distinct user interfaces on top of *the same* model/business logic code base (and by "the same," I mean *the same* — the exact source files, not a "tweaked" copy). This shows that you have cleanly separated the domain objects and business logic of the program from its presentation and interaction elements.

Ideally, you can start your program with a menu that allows the user to choose a user interface. The code can then set up the appropriate top-level view for that interface, whether it is some JFrame content or a web page.