CMSI 370-01

Interaction Design

Fall 2015

Assignment 1211

OK, back to the code gang. This time themed around the direct manipulation interaction style, of course.

Outcomes

This assignment will affect your proficiency measures for outcomes *2b*, *3a*, *3b*, and *4a–4f*.

Background Reading

Textbook reading is centered on the direct manipulation interaction style, which would be Shneiderman/Plaisant Chapter 5.

For the programming assignments, the most helpful additional material outside of the web and the *bazaar* code will be the case studies in Chapter 9 of the JavaScript textbook. These case studies demonstrate lower-level event handling with some direct manipulation elements.

For Submission

Part 1: Direct Manipulation Application

For this part, you will need a multitouch-capable and accelerometer-aware web browser. If you have your own device, then great; if not, arrange to check one out from the Keck lab.

- Modify the *boxes-touch* bazaar code so that it supports multitouch *"flicking"* (for lack of a better term). Implement guard or clamping code so that boxes don't just fly into oblivion. *Recommendation:* Bounce the boxes off the walls a la "pong."
- In addition, make the boxes respond to *gravity*—i.e., when the user tilts the device, read the current acceleration and move the boxes accordingly. Flicking and gravity should complement each other so that, if the user "flicks" a box upward while the device is upright, gravity concurrently pulls it down.

You will probably want to expand the "box area" and change the default boxes for this application. Also, feel free to calibrate the way your code interprets reported acceleration values to actual movement within the containing element. *Best practice:* Make that calibration easy to change in case a different device reports acceleration differently.

How to Turn it In

Commit a *copy* of your modified code under *boxes-with-physics* in your private GitHub repository (i.e., *don't* make the changes in place within *bazaar* and issue a pull request).

Part 2: Direct Manipulation Widget

We end by going back to the basics: design and implement a reusable direct manipulation widget for use in web browsers in general, and for your custom front end in particular. To emphasize reusability, implement your widget as a *jQuery plug-in*.

The point here is to see how low-level event handling (e.g., mouse/keyboard activity) translate into higher-level ones (e.g., selection or change events). If the first programming assignment involved direct manipulation "in the large," this one exercises direct manipulation "in the small." Some ideas:

- A selection knob or slider
- A rolling or scrolling item selector
- An entry field that accepts text/numbers with drag-and-drop character tiles
- A "here-to-there" drag-and-drop area
- A directional pad ("d-pad") control

You may use jQuery but Bootstrap use may be *CSS only*—no Bootstrap JavaScript components allowed, whether in code or triggered by *data* attributes. If you have a widget idea that is not in this list, check with me to see if it will work.

How to Turn it In

Commit your code in two places. Under *widget-from-scratch/*, provide these distinct pieces:

1. The reusable code for the widget itself (typically CSS and JavaScript)
2. A "demonstration page" that shows a standalone instance of your widget in action
3. "Eat your own dog food:" Under *front-end/*, integrate your widget into the user interface that you have already built.