6.813/6.831 • User Interface Design and Implementation

Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science
Spring Semester, 2011

PS3: Input

Due Session 23

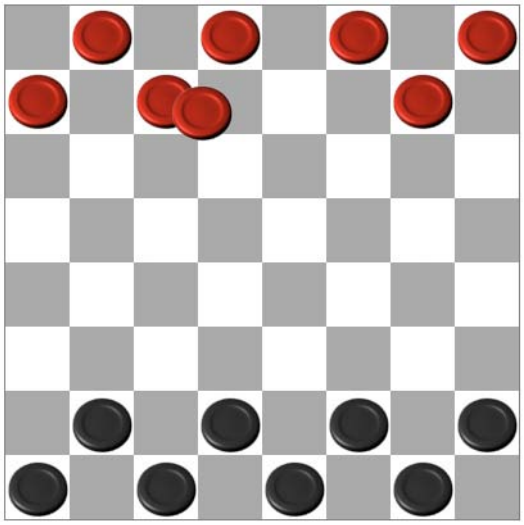
This assignment explores the following topics related to GUI input:

- event handling
- hit detection
- dragging
- selection
- enabling/disabling commands based on view state

In this problem set, you will implement input handling for the checkerboard you created in PS2, so that the user can pick up checkers with the mouse and drag them around, and remove jumped checkers and replace checkers with kings. After this problem set, you should have a basic but functional checkers game.

This assignment builds on top of the code you wrote for PS2.

Problem 1: Dragging Checkers (40%)



Add input handling to your Checkerboard, so that the user can drag checkers around.

Pressing and dragging in a square containing a checker should pick up the checker from the board. The checker should not make any abrupt jumps — neither when the user presses the mouse button, nor when the user starts moving the mouse. The checker should move smoothly with the mouse pointer, hovering over the other checkers on the board.

If the user releases the mouse button when the mouse pointer
is over an empty square, the checker should be moved to that square using Board.move(). The board model should not be changed until then. If the mouse pointer is over a filled square, then the dropped checker should be put back where it was taken from.

If the mouse pointer leaves the checkerboard during the drag operation, the dragged checker may either follow it or stop wherever it is on checkerboard — the behavior is up to you. But if the mouse pointer moves back into the checkerboard, the checker should resume following the mouse. If the mouse button is released when the mouse pointer is off the checkerboard, the checker should be put back where it was taken from.

Hints:

- Be aware of z-order, since you want the dragged component to hover on top.
- You will need to do mouse capture correctly to handle cases where the mouse pointer leaves the whole browser window. Attaching mouse listeners to the root of the view tree (document or window object) is the only way to get this behavior in HTML5.

Problem 2: Checker Selection (40%)

To make the checkers game actually playable, we need two more abilities: removing checkers after they've been jumped, and changing a checker into a king when it reaches the back rank. Although it would probably be better to do this by adding logic to the backend model (for efficiency and error prevention), for the purpose of this problem set we will instead put the user in control of these features, by adding commands that remove a selected checker and make a checker into a king.

Add selection behavior. Clicking on a square with a checker in it should select that checker, and indicate the selection by coloring the square yellow. Clicking on an empty square should clear the selection. At most one checker can be selected at a time, and the highlighted square should always correspond to the checker actually selected. After the user drags and drops a checker, the dropped checker should be selected. Only checkers can be selected, so it should never be the case that an empty square is highlighted in yellow.

Hints:

- Remember that z-order changes may change event targeting, which in turn may affect event translation. If the mousedown event is received by one target (like a canvas), but the mouseup event is received by a different target (like a checker pulled to the top of the z-order), then the browser may not generate a click event.

Problem 3: King and Remove Functionality (20%)

Add two buttons to the button bar, directly under the New Game button. The Remove Checker button should remove the selected checker. The Make King button should replace the selected checker with a king checker of the same color. Both buttons should be disabled when there is no selected checker, and the Make King button should be disabled when the selected checker is already a king.

Going Further

If you found this assignment easy and you're inclined to go further, here are some ideas for optional improvements:

- Do hit testing for the true area of a checker, so that clicking in the corner of a square doesn't pick up the checker.
- Support unlimited undo of checker moves, removes, and conversion to kings.

What to Hand In

Package your completed assignment as a zip file that contains all of your files.

List your collaborators in the comment at the top of index.html. Collaborators are any people you discussed this assignment with. This is an individual assignment, so be aware of the course's collaboration policy .

Here's a checklist of things you should confirm before you hand in:

1. Make a fresh folder and unpack your zip file into it
2. Make sure your collaborators are named in index.html
3. Make sure all assets used by your code are found in the fresh folder and load successfully
4. Make sure that the page renders correctly in at least two modern, standards-compliant web browsers (Firefox, Chrome, Safari)

6.831 / 6.813 User Interface Design and Implementation
Spring 2011