

Assignment No. 8

Implementing a Bit of Scrabble with Drag-and-Drop (with Extra Credits)

Date Due: **Aug 13th, 2020**

What This Assignment Is About

The purposes of this assignment are to give you additional experience working with the jQuery UI and to pull together much of what we've been doing throughout the semester. You are to implement **a bit** of the game of Scrabble using drag-and-drop. The idea is to display one line of the [Scrabble board \(one line sample\)](#) to the user along with seven [letter tiles](#) on a [tile rack](#). The user then drags tiles to the board to make a word, and you are to report his or her score, taking the letter values and bonus squares into consideration.

There are several pieces that you need to pull together to create this program:

- a data structure containing the [actual letter tile distribution](#) of the game and the value of each tile
 - there are 100 tiles in all, in the distribution shown at the right
 - Consider what data structure/format you want to use to store this distribution.
 - One possible JSON structure is attached and can be found [here](#). (from students: Ramon Meza and Jason Downing)
 - Another associative array version of the data structure is also attached and can be found [here](#) (from Prof. Heines):
 - [A test program here](#)
- [graphics of the tiles themselves](#)
 - the linked file shows all the letter tiles in a single graphic, which someone will have to separate out into individual files
 - note that the linked file does **not** show all 100 letter tiles
 - graphics of all of the individual tiles can be found on [this page](#)
- graphics of the individual board spaces

LETTER DISTRIBUTION		
A-9	J-1	S-4
B-2	K-1	T-6
C-2	L-4	U-4
D-4	M-2	V-2
E-12	N-6	W-2
F-2	O-8	X-1
G-3	P-2	Y-2
H-2	Q-1	Z-1
I-9	R-6	
BLANK-2		

There are many features that you can add to this program. One of the most obvious is a button that "deals" another seven random letter tiles so that the user doesn't get "stuck" if he or she can't make a word out of the letter tiles that were dealt. I'm sure that you can think of others that will improve the user experience. Be creative. This really should be fun as well as educational.

If you're completely unfamiliar with Scrabble, see <https://en.wikipedia.org/wiki/Scrabble> and <http://scrabble.hasbro.com/en-us/rules>.

Please note that we expect to see very different levels of sophistication and completeness of this assignment. That's fine. No matter how far you get, you will definitely learn.

We also expect people to help each other by sharing data structures, graphics, and even algorithms. That's fine, too, but **remember that when you use others' work or share yours with another student, you must always document where and from whom your various components came from and with whom you shared yours.**

What You Are To Do

1. Refer to the jQuery UI documentation and examples on the [jQuery UI website](#).
2. Begin your development by programming and testing dragging-and-dropping one graphic (the source) to another (the target) to ensure that you know how to do that.

3. Next, add multiple graphic sources and multiple graphic targets and write code that allows you to identify which source was dropped on which target. The use of `this` and `$(this)` can be tricky here, so you'll need to explore the examples on the jQuery UI website and do some experimentation with your own graphics.



4. Implement the "rack" (which can just be an area of the screen) to hold the user's letter tiles. Each tile must be a single draggable image.



5. Implement a single row of the Scrabble board as a series of images, each of which is a drop target.



6. Implement the algorithm to identify which letter tile was dragged to which block.
7. Implement the algorithm to tally the score of the user's word after he or she somehow indicates that all the tiles in the word have been completely played.
8. Add other features to enhance your program.
9. Test your page thoroughly. Try to anticipate all the errors that a user might make, whether intentional or unintentional. Make sure that you handle each possibility.
10. Have a friend run your application and try to "break" it. Plug any "holes" that your friend finds in your application and/or its validation scheme.
11. Create a write-up document describing what features are currently working, partially working, fully working.
12. Please see possible extra credit in "How you will be graded" section.
13. Finally, here are some helpful links for various topics related to this homework from student Jason Downing. We will discuss and post more on Piazza:
1. [How to get current position of an image in jQuery?](#)
 2. [jQuery - add image at specific co-ordinates](#)
 3. [Dictionary Lookups in JavaScript \(extra credit part\)](#)
14. Below are two examples from students' submission. **Please note that borrowing any code from these students are strictly prohibited and would be carefully checked for plagiarism.**
- An example of one-line Scrabble board implementation can be found here: <http://yongcho.github.io/GUI-Programming-1/assignment9.html>
 - An example of the Full Scrabble board lines implementation can be found here: https://downing.io/GUI/assignment9_v2.html
15. If you implement any "extra credit" features, please explain them on your "readme" file.

Submitting Your Assignment for Grading

As always, submit your github link for this assignment using blackboard. Also submit the "readme" write-up document describing the implemented features and their status.

How You Will Be Graded

This assignment will be graded on a 46-point system with points awarded as follows. Please note that the lists of features provided below are not meant to be exhaustive. They are merely representative of the types of things we are looking for in each grading category.

Criteria (numbers in parentheses are the points that can be earned for that bullet item)	Possible Points
Program Integrity / Design <i>Basic Functionality</i> <ul style="list-style-type: none">• (4) letter tiles in the player's "hand" are selected randomly from a data structure with the proper distribution of the letters• (4) letter tiles can be dragged-and-dropped onto target Scrabble squares• (4) program identifies which letter tile is dropped onto which Scrabble square• (4) board includes bonus squares• (4) score is tallied correctly, including consideration of bonus square multipliers <i>Additional Functionality</i> <ul style="list-style-type: none">• (3) any number of words can be played until the player wishes to quit or depletes all tiles• (3) the board is cleared after each round so that a new word can be played• (3) after playing a word, only the number of letter tiles needed to bring the player's "hand" back to 7 tiles are selected• (3) score is kept for multiple words• (2) Tiles can only be dragged from the "rack" to Scrabble board. If the user drop them anywhere else, they will be bounced back to the "rack".• (2) Once the tile is placed on the Scrabble board, it can not be removed.• (2) Except for the first letter, all sub-subsequent letters must be placed directly next to or above/below another letter with no space. Else, they will bounce back to the "rack".	38
Source Code Documentation and Formatting <ul style="list-style-type: none">• user name and pertinent contact information appear in all source files• all files contain adequate explanatory documentation that is meaningful and does not merely echo code• all files are properly indented and formatted with adequate white space for readability• any sources used are cited in comments embedded within code• <u>Complete write-up</u> about implemented features	8
Extra Credit <ul style="list-style-type: none">• (3 point) Full Scrabble board lines are implemented• (2 points) Validating to see if a word that the user enters is valid (from /usr/share/dict/words)	3

* This assignment is based on 91.461 by Prof. Heines.