Steph Greene
Professor Miller
6.813 HW2
25 April 2012

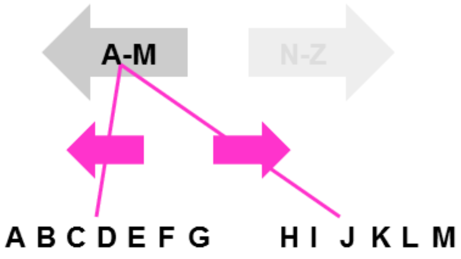
Heuristic Evaluation of Keyboard

<u>Heuristic:</u> Nielson #4: User Control and Freedom
<u>Aspect:</u> Cannot Delete Characters
<u>Description:</u> As far as I can tell, there is no way to delete characters after they've been selected. Users can override them by positioning the "cursor" (the bold underline) under the letter and selected a new one, but they can't make a space blank after a letter has been put there.
<u>Severity:</u> Major
<u>Potential Mitigations:</u>

1) My first instinct was to look for a blank space among the special characters. Allowing users to override a mistyped character with a blank space would be a relatively simple fix.
2) Currently using the up arrow when in the top box doesn't do anything, so you could give it delete functionality when the focus is on the top box.
3) It looks like you're trying to create a keyboard using only the arrow keys. This is admirable, but considering the technologies you mentioned in your problem statement, virtually all of these will also have other keys. You could give one of them delete functionality.

<u>Heuristic:</u> Nielson #4: User Control and Freedom
<u>Aspect:</u> Cannot Undo Selections
<u>Description:</u> There should be a way to go back and correct errors when entering the username/password/movie. Right now, if I enter my username, start entering my password, and want to go back and change the username, I can't do it without restarting the application.
<u>Severity:</u> Major
<u>Potential Mitigations:</u> Add either a back button or a combination of arrows that allows the user to go back a screen.

<u>Heuristic:</u> Nielson #6: Flexibility and Efficiency
<u>Aspect:</u> Choice of Default Letters
<u>Description:</u> When the user chooses to go left or right from the menu state on the right, the current defaults "E" and "J" do not minimize either the maximum distance the user might need to reach any letter or optimize the default location based on letter frequency. If it did, this might greatly increase efficiency for seasoned users.



<u>Severity:</u> Moderate
<u>Potential Mitigations:</u>

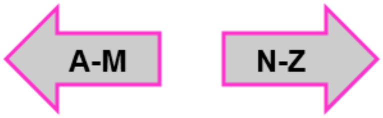
1) Change the default to be the middle character in each half menu. For example, it should be 'D' and either 'J' or 'K' in the left hand uppercase letters menu.

2) Try to minimize distance from frequently used letters. For the first half of the alphabet, that would be “A” and “E.” In this case, E-G should probably be moved to the right side so as not to confuse users.

Heuristic: Nielson #9: Error Prevention
Aspect: Viewing Password
Description: This one is more of a tradeoff. Presumably after the user enters their password, the system will ensure the username password combination is valid. Currently, users can’t tell whether the password they’ve typed is what they expect. This interface isn’t like a keyboard where users can feel whether they’ve hit the right letter. As users become more familiar with the interface and learn to enter letters faster, they might make more mistakes. Letting the user see the most recent letter they’ve typed would help mitigate this, but it also might be a security concern since anyone around would be able to see their password if this were used in something like Netflix and the interface was displayed on the television.
Severity: Moderate
Potential Mitigations: Use the touchscreen model, where the most recent letter is displayed for either a few seconds or until the next letter is selected.

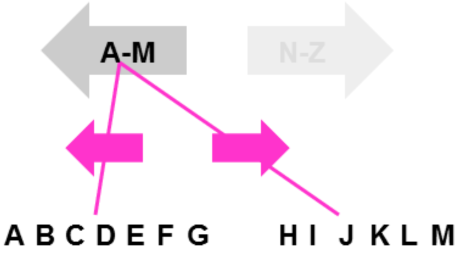
Heuristic: Hielson #4: User Control and Freedom/Hielson #6: Flexibility and Efficiency
Aspect: Undoing Immediate Actions
Description: One of the most difficult aspects of this interface for me was the fact that pressing the opposite arrow did not always undo the last move. Even after I had figured out the pattern, I still had to work to remember not to press the opposite arrow automatically which slowed me down a bit.
Severity: Moderate
Potential Mitigations: This one is a bit harder because the opposite arrow is usually needed for another function. The inclination to undo a move by pressing the opposite arrow was powerful enough that it would be worth putting a bit of time in to try to mitigate the effect.

Heuristic: Nielson #6: Flexibility and Efficiency
Aspect: Interface Reset
Description: After I press enter, the interface resets to the state displayed on the right, with the big arrows selected. After a fair amount of use, I figured out this was likely to allow people to efficiently select a letter from the other side (i.e. choose “A” and then “N”) very rapidly. Initially, I was entirely confused about the purpose and I wasn’t expecting it, which cost me a bit of time trying to reorient myself. Even once I knew the reset would occur, I rarely remembered it when selecting enter, I was just able to reorient faster. I also tried using this interface as remote (i.e. using only a thumb) as would be consistent with the Netflix use case described in the problem statement. In this case the reset cost me a bit more time, because I would keep clicking in the direction I wanted to go and I would have to correct it by pressing a new arrow. This was more of a problem in this case because it took noticeably more time and effort to move just my thumb around rather than simply using a different finger that might already be positioned over the needed button.



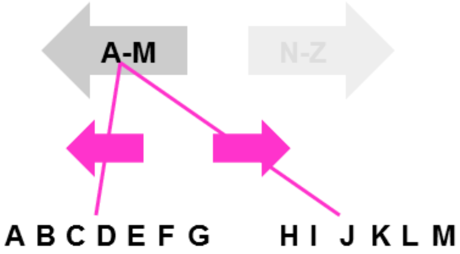
Severity: Moderate
Potential Mitigations: Perform A/B testing and decide whether resetting is more efficient.

Heuristic: Nielson #1: Match the Real World
Aspect: Arrow Directions
Description: After using the interface for over an hour, it still takes me a bit of conscious thought not to press the down arrow to get the menu on the right to display. I think since the menu appears underneath the prior display.
Severity: Moderate
Potential Mitigations: None at this time.



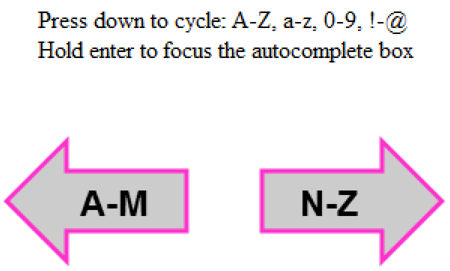
Heuristic: Nielson #2: Consistency and Standards
Aspect: Continuous Scrolling
Description: Holding down the arrow key does not allow continuous scrolling. To scroll, the user needs to repeatedly press the arrow keys.
Severity: Moderate
Potential Mitigations: Add handling for continuous scrolling.

Heuristic: Nielson #6: Flexibility and Efficiency
Aspect: Potentially Unnecessary Step
Description: The step pictured to the right seems somewhat unnecessary and therefore its inclusion seems inefficient. However, it does allow the list of letters to have two defaults (in this case “E” and “J”) which might be more efficient for seasoned users.
Severity: Moderate/Minor
Potential Mitigations: Perform A/B testing and decide whether to remove this step.

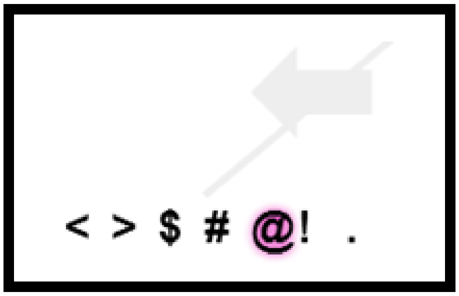


Heuristic: Nielson #6: Flexibility and Efficiency
Aspect: Character Order
Description: The current character order is great for learnability since virtually everyone needing to use this interface will know the alphabet, but it is not particularly effective from an efficiency standpoint. I’m not sure the efficiency gained in reordering the characters would outweigh the cost in learnability, but it’s something to consider.
Severity: Minor
Potential Mitigations: Reevaluate character order and run A/B testing.

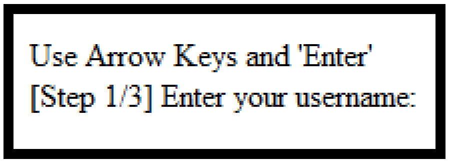
Heuristic: **Nielson #3: Help and Documentation**
Aspect: “Hold enter to focus autocomplete box”
Description: Some users may be able to figure out what this means, but “focus” is more of a computer science term that an average user might not know. Just to be sure, I ran this by a friend who is Course 7/15 and she was not sure what the phrase meant. The use of “hold enter” is also a bit misleading. You seem to be trying to get across the fact that the enter button needs to be held down for it to work, but nowhere does it indicate that the button must be released for anything to happen. When I showed this to the same friend, she held down the button for about 10 seconds wondering why nothing was happening.
Severity: Minor
Potential Mitigations: Make language more “average” user friendly.


Press down to cycle: A-Z, a-z, 0-9, !-@
Hold enter to focus the autocomplete box

Heuristic: **Nielson #2: Consistency and Standards**
Aspect: Missing Space
Description: There is no space between the “@” and the “!” in the right hand side of the extra character select menu. Obviously, this isn’t a huge problem, but every other character is nicely spaced.
Severity: Extremely Minor
Potential Mitigations: Add a space.



Heuristic: **Nielson #5: Visibility of System Status**
Aspect: Step Number Display
Description: Displaying the step the user is currently on and how many are left is helpful. This gives the user a rough estimate of the amount of time remaining until they’ve completed their task
Severity: N/A
Potential Mitigations: N/A


Use Arrow Keys and 'Enter'
[Step 1/3] Enter your username:

Heuristic: **Nielson #1: Match the Real World**
Aspect: Down Arrow Yields Lowercase Characters
Description: When looking at uppercase characters, the down arrow yields lowercase versions of the same characters. This is helpful as users would naturally expect this. Lowercase characters tend to be smaller and the down arrow is used in many cases to make things smaller. Several other interfaces I’ve seen also use the down arrow to display lowercase arrows in text selection, so there is some additional external consistency there.
Severity: N/A
Potential Mitigations: N/A

Heuristic: Tog's First Principles #3: Color Blindness
Aspect: Color Choice
Description: The high contrast in color choices will greatly assist the color blind and those with limited visibility use your interface.
Severity: N/A
Potential Mitigations: N/A

Heuristic: Norman Principles #4: Feedback
Aspect: Highlighting Focused Element
Description: The pink highlighting gives instant feedback about the most recent move. This will help users learn how to use the interface more quickly.
Severity: N/A
Potential Mitigations: N/A

Heuristic: Nielson #2: Consistency and Standards
Aspect: Color Consistency
Description: The fact that the entire interface is black and white except the pink highlighting trains the eye to look for pink in every menu.
Severity: N/A
Potential Mitigations: N/A

Other Considerations:

Scalability: This isn't really a usability heuristic, but I'm not sure this interface would work especially well if the target application required more special characters. You could add an extra row with your current system, but this might be hard for the user since there is no inherent ordering. The best option I can think of is to have a double row in the special characters menu (shown on right).