Journal of Universal Computer Science, vol. 20, no. 7 (2014), 986-1005
submitted: 31/1/14, accepted: 30/6/14, appeared: 1/7/14 © J.UCS

Keyboard-Card Menus: A New Presentation of Non-Standard Shortcuts

Benjamin Berman
(The University of Iowa, Iowa City, Iowa, USA
benjamin-berman@uiowa.edu)

Juan Pablo Hourcade
(The University of Iowa, Iowa City, Iowa, USA
juanpablo-hourcade@uiowa.edu)

Abstract: "Keyboard-card menus" are a new type of menu system in which potentially hundreds of menu items are arranged in sets of keyboard patterns that are designed to be navigated using only a computer keyboard's character keys, for fast access. In selecting items from these menus, novice users physically rehearse the same actions that an expert would use. We describe these menus and their potential applications in further detail, along with a study comparing keyboard-card menus' presentation of what are effectively shortcuts with a presentation of these same shortcuts that uses dropdown menus. The data from our study shows that keyboard-card menus have significant advantages over dropdown menus in making the transition to expert use faster.

Key Words: Keyboard shortcuts, accuracy, efficiency, visualization, learnability, computer chording
Category: H.5.m

1 Introduction

Keyboard-card menus are a new type of menu we have developed which present menu items in computer-keyboard shaped arrangements. They are designed to be navigated using the keyboard, and in doing so users end up learning shortcuts. keys themselves as modifier keys (and which we discuss in more detail below). Our testing shows significant advantages over a presentation of these same shortcuts that uses dropdown menus. Our ultimate goal in working on keyboard-card menus is to develop a system that avoids making tradeoffs between three properties: how easy the system is to learn, how efficiently experts are able to select menu items, and how many menu items are easily accessible.

While widely-used interaction techniques often support two of these three properties, they do so at the expense of the third, which can be considerably limiting for certain tasks and people. Consider an example we had in mind while developing the menu: undergraduate college students writing and manipulating mathematics. Many of these students do not have the time to learn an unfamiliar, non-WYSIWYG (what-you-see-is-what-you-get) system like *LaTeX*, but also need an efficient system since they are asked to write many equations in homework assignments (especially if they are asked to "show their work"), *and* they need to be able to access many mathematical symbols. Handwriting recognition systems might be much easier to learn, but even if the system's recognition accuracy is 100%, in our opinion handwriting speed is a low bar for efficiency (14 to 15 year old students' handwriting, when copying, has been measured as averaging only 118 characters per minute or, at 5 characters per word, 24 words per minute [Graham et al. 1998]). WYSIWYG systems like *Mathematica* (http://www.wolfram.com/mathematica/) and Microsoft *Word*'s equation editor (http://www.microsoft.com/mac/downloads) allow use of either the mouse or keyboard shortcuts to select symbols from menus and toolbars. Although this might be seen as supporting both novice and expert users, since keyboard shortcuts can provide speed advantages once learned, it has been shown that, in general, many users never make the transition to using shortcuts, even in heavily used applications like *Word* [Lane et al. 2005], and an important part of supporting novice users is helping them attain expert status.

Writing and manipulating mathematics is not the only application where the users would need to be quickly trained to efficiently select from a large number of menu items. Novice writers of other sorts of code could benefit as well; for instance, novice HTML coders must spend considerable time learning which tags to select from the large number available. We suspect that there are many other sorts of activities, particularly in data entry, classification, and retrieval, that could be made more practical by supporting a better balance between efficiency and learnability in applications where items must be selected from a large pool.

In the next section, we describe how keyboard-card menus present menu items, how they are navigated, and some related work. We go on to describe our study, which compared keyboard-card menus with dropdown menus, and discuss the study's results, which show significant advantages for keyboard-card menus.

2 Keyboard-Card Menu Design

Keyboard navigation for keyboard-card menus (and keyboard navigation for the dropdown menus in our study) is unusual in that users are required to press and hold *character* keys (i.e. to use the character keys themselves as modifier keys); we call these interactions "rolled-chords," or "rolled-chord shortcuts". In the second subsection, we include a discussion of the advantages and disadvantages of this approach relative to other ways the keyboard might be used in the process of navigating the menus.

2.1 Keyboard-Card Menus

Inspired, in part, by work on keyboard-based menus for wearable computers, seen in [Lyons et al. 2003], we have developed keyboard-card menus which lay out menu items in keyboard patterns that *show* users what keys to press, instead of simply listing menu items with shortcut key names; at least in principle users do not even need to know the name of the key they want to press. (An example keyboard-card menu–one used in the study described below–is shown in the lower half of fig. 1, and in fig. 2).

More specifically, the menu system consists of colored "keyboard-cards", each of which serves as a submenu (except for one card that serves as a root menu). Each keyboard-card has printed on it a set of squares arranged in a keyboard pattern. Printed on top of some (potentially all) of these squares are the menu items that can be selected by pressing the corresponding keys. To make the affordances of the menu system more obvious to users, squares with menu items are also lighter in color than the rest of the card and each square corresponding to a submenu has an arrow in the bottom right corner. As a secondary mechanism for associating the menu item with the key, and to make it even more obvious to users that they should use the keyboard rather than the mouse, each square with a menu item also has the corresponding key's character printed in white in a large font, behind the menu item text.

and monitor, not a touchscreen. The S key is pressed and held down to make fig. 2 appear on the screen. instead

Initially, when no keys are pressed, only the root card, with top-level menu items, is visible (as in the lower half of fig. 1). Whenever a key corresponding to a submenu (rather than a leaf in the hierarchy) is pressed–*and held down*–that submenu's keyboard-card is placed on top of the existing card(s), slightly offset to show the number of keyboard-cards below it (as in fig. 2, which would be displayed while the S key is held down). keyboard-card again. When one releases the key, the corresponding submenu's keyboard-card disappears. Holding down a key corresponding to a leaf in the menu hierarchy, in addition to performing whatever action is assigned, thickens the outline of the corresponding square on the fully-visible top card.

Because menu item selection is performed entirely from the keyboard, without the user having to move his hands, it can be very efficient. In addition, simply by navigating the menu hierarchy, users end up practicing, and thereby learning, shortcuts needed to access menu items, and may actually stop needing the menu altogether. This idea of "physical rehearsal"–teaching and reinforcing the physical act of using a shortcut during the process of navigating a menu hierarchy–can be found in work on gestures and "marking menus" by Kurtenbach and Buxton [Kurtenbach and Buxton 1994, Kurtenbach 1993].

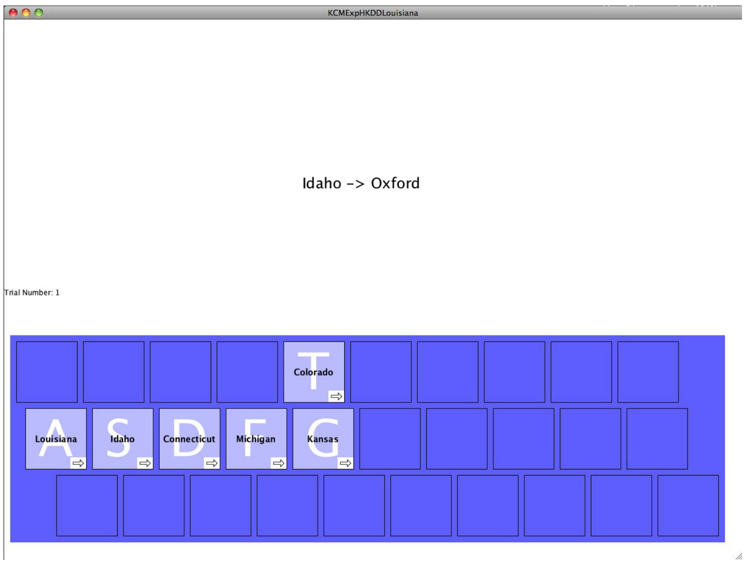

Figure 1: Participants in our study used the keyboard-card menu system in the bottom half of the window to select state-town pairs. When a key with a small arrow is pressed, a child keyboard-card, such as that in fig. 2, appears.

2.2 Rolled-Chord Shortcuts

The term "rolled-chord" comes from western classical music and refers to playing several notes together but initiating them at different times. Here we use the term to refer to pressing multiple computer keys at the same time, but initiating the presses in a particular order. However, while this term could be used to describe shortcuts involving special modifier keys (e.g. Ctrl-X, Alt-F4, Ctrl-Shift-S, etc.) we use the term to describe shortcuts where the modifier keys may in fact be letter keys. (In applications where users need to type normal text, we assume that a separate mode would exist, as with the text editor *vi* or its extension, *Vim* (http://www.vim.org). For instance, one might press *and hold* the F key, then press *and hold* the D key, and, finally, press the J key to select some particular menu item. In this paper, we consider only rolled-chord shortcuts using the letter keys plus the semicolon, comma, period, and forward slash keys on a QWERTY keyboard, since avoiding the use of special modifier keys has the advantage of reducing hand movement and therefore potentially increasing speed.

Rolled-chords are to be contrasted with "standard" chording in which the exact order of key presses does not matter because it is assumed that key presses are initiated simultaneously. Standard chording may be used to achieve high speeds of data entry–on specialized keyboards, text entry speeds of up to 300 words per

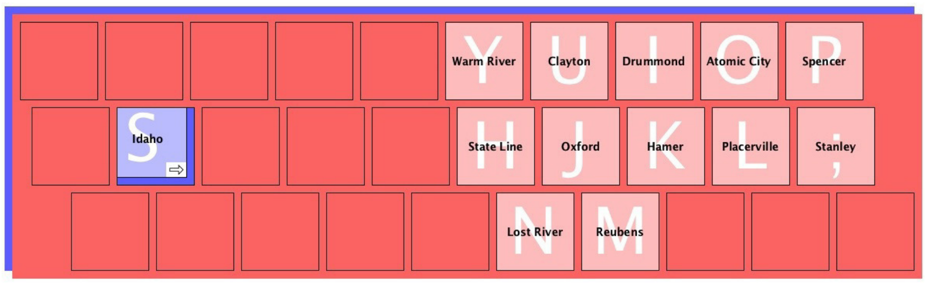

Figure 2: The keyboard-card menu from fig. 1 with the S key pressed. Note how a hole has been "punched out" of the card.

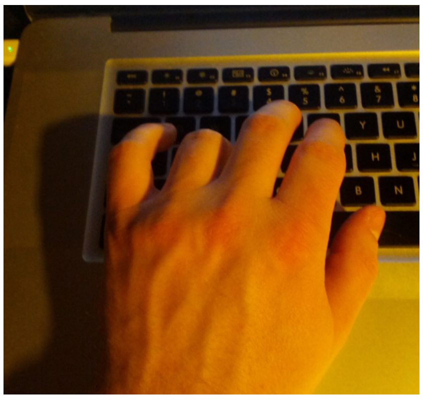

Figure 3: Keyboard-card menus are intended to be used with a normal keyboard and monitor, not a touch-screen. The S key is pressed and held down to make fig. 2 appear on the screen.

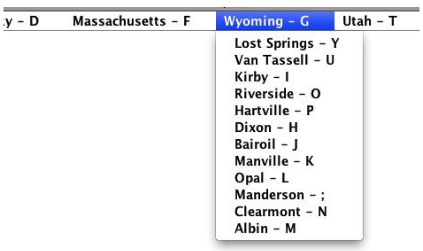

Figure 4: Part of a dropdown menu, navigable by pressing and holding keys rather than with the mouse, used in the study; note the letters on each item. Also note that the text is the same font and size as in the keyboard-card menus.

minute are possible [Shneiderman and Plaisant 2010]. While one would expect standard chording to be faster because no time is needed to ensure a particular ordering of key presses, ordering does provide more potential shortcuts and, in particular, more shortcuts that may be accessed without moving one's fingers off the home row of keys. Furthermore, for sequences of shortcuts where the initial keys are identical, we can allow users to hold down those initial keys while pressing and releasing the final keys; for instance, over the course of entering the sequence of shortcuts "F-J, F-K, F-J", the F key does not need to be released (we will use square brackets to indicate where keys are held down; e.g. "F-[J, K, J]" indicates that the F key is held down while J, then K, then J, again, are pressed).

Rolled-chord shortcuts may also be contrasted with the use of short sequences of typed characters (e.g. ": q ENTER" is used to quit in *vi*). While such sequences remove limitations on the numbers of shortcuts that can be provided, rolled-chords shortcuts (we suspect that having more than three, and possibly more than two, keys per shortcut would be impractical for many users) and, if absolutely needed, sequences of rolled-chords could still be treated as shortcuts. still provide hundreds of possible shortcuts. To see why, first consider only the shortcuts accessible by pressing two keys. Some pairs of keys are clearly at least relatively awkward to press together. For instance, pressing the W and the X keys together might best be accomplished by holding the W key with the ring finger while using either the middle or index finger to press the X key. Of course this is not a terribly intuitive approach, since normally W and X are both pressed using the ring finger. In addition, using the middle finger can be quite uncomfortable because of the tight connection between the middle and third fingers, and using the index finger may require a larger than normal amount of hand movement.

We avoid these problems altogether if we consider just the two-key rolled-chord shortcuts invokable by pressing a key on one side of the keyboard followed by pressing a key on the other side of the keyboard (using first one hand and then the other; in the study described below we actually consider just a small subset of these). Each side has 15 keys, so the total number of such available shortcuts is (2 sides) * (15 initial-keys/side) * (15 secondary-keys/initial-key) = 450. For comparison, Microsoft Word for Mac 2011 (http://www.microsoft.com/mac/downloads) contains approximately 230 leaves in its menu hierarchy, if one ignores the Font submenu which adds several hundred more (this is not to say that it would necessarily make sense to add rolled-chord shortcuts to any or all of the items in Microsoft Word's menu hierarchy, but, instead to suggest that many computer end-users and developers have already been exposed to menu hierarchies of such a scale).

We also hypothesize that rolled-chord shortcuts involving three (or perhaps even more) keys may be of practical value, despite being more complicated. First, consider text editing in *vi*, again, and the problem of moving the cursor around in the buffer. Basic cursor movement is accomplished by pressing the H key to move left one character, the L key to move right one character, the J key to move down one line, and the K key to move up one line. In addition, one can move around in larger increments, e.g. forwards or backwards a word, or to the beginning or end of a line, and up and down by paragraph or by page. These larger increments require entirely different keys (e.g. W for forward a word), which we suspect means that many users just avoid pressing the faster movement keys (they may not fully recall what key to press and also wish to avoid accidentally deleting text).

With an alternative text editor using rolled-chord shortcuts, one might use multiple keys in the left hand to select the speed of movement while still using H, J, K and L in the right hand to select the direction. For instance, one might press and hold F to go into a "move" mode and press J to move down a single line followed by L to move forward a character within that next line (recall that this would be "F-[J, L]" , i.e. the F key would not need to be released between J and L presses). One could then add the D key to change the rate of movement to, say, 5 lines/characters per H/J/K/L press (i.e. "F-[D-[... H/J/K/L key presses here ...]]", so "F-[D-[J, L]]" would move the cursor down 5 lines and then over 5 characters). One could, instead of D, add the S key to change the rate to, say, 20 lines/characters (so "F-[S-[J, L]]" would move the cursor down 20 lines and then over 20 characters). Or, instead of D or S, one could press A to move 80 lines/characters at a time.

Because the F key is used in conjunction with the D, S, and A keys in this way, and the keys are designed to fit under the first, second, third and fourth finger (starting with the index finger), it is likely that the first two keys in the three-key sequences are at least relatively easy to press and hold down together, and the intensity of the operation is roughly correlated with the center of gravity of the left hand relative to the keyboard and with its degree of rotation. Because of this correlation, and also because this use of three-key shortcuts could be generalized to applications, besides cursor movement in text editors, where varying degrees of intensity are called for (e.g. zooming and panning in maps, from the keyboard), this style of cursor movement might be considered more organized and intuitive than that of *vi* and other editors. The approach may also be used with other sets of keys (e.g. R, E, W and Q might be used in the alternative text editor to move the cursor while selecting text).

A second instance where having three-key rolled-chord shortcuts might be practical is simply to "extend" the number of keys (rather than intensifying the keys) that can be pressed after pressing the initial key. Suppose, again, that the first key we press is F, but this time the items under the right-hand keys are codes for various medical procedures commonly associated with a hospital department. If we have a list of up to 60 such procedure codes, we could extend the 15 right-hand keys using the D, S, and A keys, as in the text editing example above, while keeping them associated with the "F" department.

Third, three-key rolled-chord shortcuts may still be very fast, even relative to two-key rolled chord shortcuts. This may be especially true when all three keys come from the center row of the keyboard (e.g. A-[F-[J]]) and so do not require finger joint extensions or contractions.

Rolled-chords also have several important advantages over simple key sequences. First, in sequences like “F-J, F-K, F-J” we may reduce the number of keystrokes needed, as explained above (“F, J, F, K, F, J” vs. “F-[J, K, J]”). Second, rolled-chord shortcuts enforce the rule, when assigning shortcuts to commands, that the keys pressed in a shortcut never use the same finger twice. We suspect that following this rule increases speed; it also, as pointed out in [Purdy 2012], removes the possibility of accidentally entering a repeated letter sequence like “F, F” because of an operating system’s key press repeater. Third, when mistakes are made in selecting an initial key of a shortcut, no additional keystroke (e.g. "Backspace" or "Delete" key press) is needed to undo the mistake–the user simply releases the key. This feature is especially important not only because it makes mistakes more preventable, but because it means that *when the available shortcuts are displayed in a navigable hierarchy, as with keyboard-card menus, users can “peek” into submenus without having to press an additional key to back out again.* We expect that this will help novice users considerably in exploring hierarchies and in searching for items that have non-obvious locations within the menu hierarchy.

Finally, one should not overlook the possibility of using forests of menu trees, and of using the shortcuts in one collection to move to another. When all rolled-chord shortcuts are single-key, this would simply degenerate to simple key sequences. Many applications might require a balance between one large tree and many small ones.

3 Related Work

Encouraging shortcut use within graphical user interfaces is an active area of research. A theoretical explanation of the problem of low levels of shortcut usage can be found in *Paradox of the Active User* [Carroll and Rosson 1987] which notes that people are biased towards using software to solve their immediate problems, rather than towards exploring what the software can do, and also generally prefer to use known solutions rather than new ones. The severity of the problem has been empirically verified in [Lane et al. 2005], and in trying to solve it researchers have gone so far as suggesting that the option to click on a dropdown menu item should be disabled when a keyboard shortcut is available [Grossman et al. 2007].

Our general approach–graphically supporting users in entering commands from the keyboard–has been taken before in a several alternate ways. “GEKA” [Hendy et al. 2010] populates a list of command names and shortcuts, refining the list as users type, allowing a command line-like experience with a graphical user interface. "HotKeyCoach" [Krisler and Alterman 2008] uses a transparent popup window to inform and remind users of available (traditional) shortcut alternatives in a non-disruptive way as they work with an application. "Blur" [Scarr et al. 2011] combines features seen in "GEKA" and "HotKeyCoach", notifying users when a command could be invoked by pressing escape and then typing "hot commands" (e.g. "align left") into a transparent popup window; it also provides lists of command recommendations. "ExposeHK" shows users available (traditional) keyboard shortcuts when a user presses a modifier key; it replaces toolbar icons with printed shortcuts, expands all dropdown menus at once (with shortcuts printed next to the menu item name), and uses tooltips to show shortcuts for icons in a Microsoft *Office 2007*-style ribbon [Malacria et al. 2013].

The line of work by Kurtenbach et al. on "marking menus" (which extend "pie menus" by allowing expert users to select items without looking at the menu) is in many ways parallel to ours, a major difference simply being the intended set of applications: they assume a context in which users need some sort of pointing device most of the time (e.g. technical drawing), while we assume a context where most of the time users want to have both hands on the keyboard (e.g. text editing). In particular, the design of keyboard-card menus follows three principles used in the design of marking menus [Kurtenbach 1993]: *Self-revelation*: interactively telling users what commands are available and what these commands do, *Guidance*: showing users how to invoke commands during the process of self-revelation, and *Rehearsal*: guiding novice users through the same physical motions that an expert would use A more recent alternative to marking menus, but still following these design principles, is "OctoPocus" [Bau and Mackay 2008]. Kurtenbach et al. also address the issue of providing large numbers of quickly accessible menu items in work on the "HotBox" [Kurtenbach et al. 1999].

Work that most closely relates to ours may be explorations of interaction with multi-touch surfaces, since these provide a new opportunity for computer chording. Relevant work includes "Multi-finger Chorded Toolglass" [Malik 2007], "Multi-Touch Menu" [Bailly et al. 2008], "Finger-count shortcuts" [Bailly et al. 2010], "Multi-touch Marking Menus" [Lepinski et al. 2010], and, perhaps most significantly, "Arpege" [Bau et al. 2010]. An advantage of using a multitouch surface is that thumb mobility may be exploited, as seen in [Bailly et al. 2008] and [Malik 2007]. However, physical keyboards appear to be faster to type with than virtual keyboards on a multi-touch surface [Varcholik et al. 2012], and using them in presenting chords does not create occlusion problems (involving the hands) which, as highlighted in [Bau et al. 2010], would likely make the presentation rolled chords much more complicated.

4 Study Description

4.1 Study Motivation

Keyboard-card menus are one of many possible ways to present rolled-chord shortcuts. Using dropdown menus with menu items marked with letters, as in fig. 4, as a baseline for comparison makes sense because of their advantages over keyboard-card menus and other types of menus, including that most existing applications already use dropdown menus, making the incorporation of rolled-chord shortcuts into the design of these applications somewhat more straightforward for software developers, users may feel more comfortable with them, since they are almost certainly more familiar with them, they do not hide submenu siblings (e.g. "Massachusetts - F" and "Utah -T" are still visible in fig. 4), and they are more compact (at least for relatively small hierarchies).

Reducing the advantage of compactness, screen resolutions have improved dramatically over the past decade (see Browser Display Statistics at W3Schools, `http://www.w3schools.com/browsers/browsers_display.asp`) and secondary monitors are often available. Some advantages of keyboard-card menus over dropdown menus are: since the size and shape of keyboard-card menus is relatively constant, the content being manipulated could never possibly be covered up by the menu, since dropdown menus typically have only a single row of items at the root level, they may allow for broader overall menu hierarchies, which have advantages over deep hierarchies [Norman 1991], and the keyboard pattern makes using of the keyboard rather than the mouse the obvious choice.

The last point may be the most important, though it is unclear how many users would decide to use the mouse rather than the keyboard if presented with labeled dropdown menus. Finally, there is our *experiment hypothesis*: new users are able to learn menu items' shortcuts faster. By this we mean both that the number of selections needed to memorize a shortcut is low and that the time to enter a shortcut that has not been memorized is low.

4.1.1 Initial Rationale for Experiment Hypothesis:

There are several reasons to think that keyboard-card menus might help users learn shortcuts faster. First, although the size of each menu item's text is the same in both the keyboard-card and dropdown menu presentations, the character used to access the menu item is bigger, and therefore may be easier to read. Second, the greater amount of space between menu items may also make misreading the character associated with a menu item more difficult. Third, placing menu items in a two-dimensional array might help users exploit approximate information about locations because there are more characteristics that can be remembered. For instance, given set of menu items, if these menu items are displayed in a one dimensional list, one might be able to remember that menu item

"A" is above menu item "B", but, if the items are displayed in a two-dimensional array, one might be able to remember that "A" is above "B" *and* one might be able to remember that "A" is to the left of "B". Fourth, with keyboard-card menus there is an alternate, and possibly faster, way to associate the key with the menu item, since the system could be used without looking at the character printed under the menu item.

After the experiment, we thought of a fifth, perhaps most likely, reason keyboard-card menus help users learn shortcuts faster. This is given in our "Discussion" section, below.

4.2 Study Design

To see if the keyboard-card menu improves rolled-chord shortcut learnability relative to a dropdown menu presentation (i.e. relative to dropdown menus labeled as in fig. 4 that one navigates by pressing and holding keys), we performed a 20 participant (8 male) within-subjects comparison with adult self-described touch-typists as our participants, based loosely on [Ahlström et al. 1999] and [Grossman et al. 2007]. Participants were recruited through a mass email and word of mouth at a large state university in the United States. After a 1 minute typing test (using the "Space Cowboys" test on typingtest.com), each participant performed up to 720 selection tasks ("trials") over the course of at most 50 minutes, using one of the two menu systems, followed by half of a brief questionnaire. The trials were then repeated, over the course of a second period of at most 50 minutes, using the second menu system with a second hierarchy of menu items, followed by the second half of the questionnaire. Before each 50-minute period, participants performed 10 warm-up trials. Participants were told that they could ask questions or take breaks at any point that would not count against the 50 minutes for each menu system, and participants were asked, at the 25-minute mark, if they would like to take a break. Participants were also told that two thirds of the compensation would be prorated based on how many of the 1440 trials they completed.

In each trial, each participant was first prompted by text on the screen to press SPACEBAR, which started an invisible timer for the trial. They were then presented with text of the form "U.S. state -> small town", e.g."Delaware -> Little Creek" (see fig. 1); while the states were presumably recognizable by participants, the towns were unlikely to be recognized since they were selected from lists of the smallest towns in those states. The trial ended when the town was correctly selected from the submenu labeled with the state's name, at which point the elapsed time was recorded and the participant was again prompted to press SPACEBAR to start the next trial. The number of incorrect selections of town names during each trial was also recorded (usually this was zero).

Two non-overlapping hierarchies of states and towns were used. Each consisted of 6 states and 72 towns, 12 from each state. The states were assigned the A, S, D, F, G, and T keys and the towns were assigned the Y, U, I, O, P, H, J, K, L, semicolon, N, and M keys (as in fig. 1, fig. 2, and fig. 4). From each hierarchy, 14 towns, 2 or 3 from each state, were randomly selected for use in trials. The 720 trials for each menu system were divided into 12 blocks of 60 trials, and the numbers of occurrences of each town in each block were 12, 12, 6, 6, 4, 4, 3, 3, 2, 2, 2, 2, 1, and 1, with the order varying randomly from block to block. This organization of hierarchies and trials, and the Zipfian distribution used ("which has been shown to represent command use frequency in real applications"), was used in [Grossman et al. 2007].

While the same 14 towns for each hierarchy were used across participants, the numbers of occurrences were paired, randomly, with different towns for different participants (the pairing did not change across blocks for a given participant). For instance, participants A and B would both see "Delaware -> Little Creek", but participant A might see it 6 times in every block while participant B might only see it 2 times in every block.

The numbers of participants were balanced between the four possibilities determined by whether the keyboard-card or dropdown menu was seen first and by which set of states and towns was used with which menu type.

The software for the experiment was run on a 2.8 GHz Intel Core 2 Duo MacBook Pro with Mac OS X Snow Leopard using Java 1.6. However, participants viewed the software on an 18 inch, 1280 x 1024 pixel separate monitor (Dell Model No. 1800FP) and used a separate keyboard (Dell Model No. L100). The experiment was performed in a faculty member's office on a university's campus.

5 Study Results

The average selection times for the two different menu systems can be seen in fig. 5. Note that while outlier trials were removed (a trial was considered an outlier if its time was greater than 3*(Q3-Q1)+Q3, where Q1 and Q3 were the first and third quartile values; 0.55% of trials were considered outliers), average times for participants who did not complete all 1440 trials were left in. Averaged trial completion times for each block are only complete for all participants and menu types in blocks 1-8. One participant was only able to complete trials in blocks 1-8 for both menu types. Two other participants were only able to complete trials in blocks 1-10 using the dropdown menus, but were able to complete trials in all blocks using the keyboard-card menus. The remaining 17 participants were able to complete trials in all blocks.

To analyze this data we fit quadratic equations, as first approximations, to the completion time data for each participant (with trial number as the independent variable and completion time as the dependent variable), leaving out times

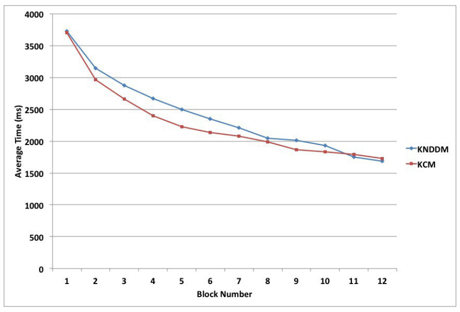

Figure 5: Average trial completion times, in milliseconds, for the keyboard-navigated dropdown menus and keyboard-card menus in the study. Note that data is only complete for blocks 1-8.

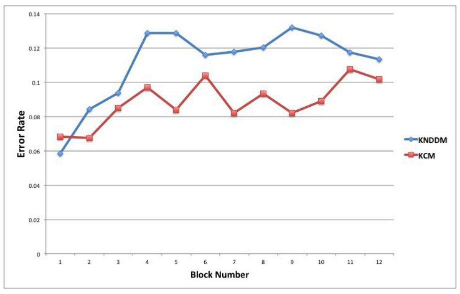

Figure 6: Error rates for the keyboard-navigated dropdown menus and keyboard-card menus in the study. Note that data is only complete for blocks 1-8.

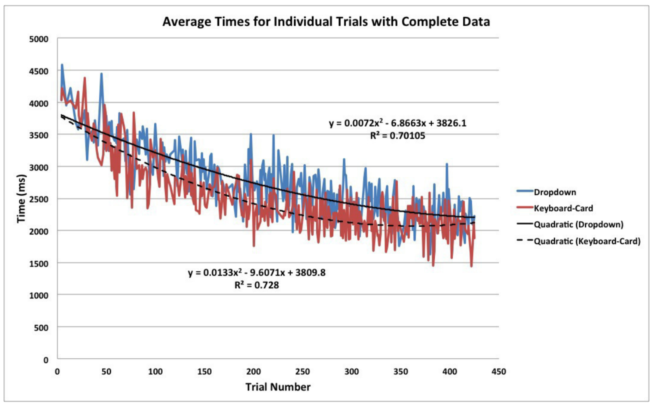

Figure 7: Average times across participants at individual trial numbers where data is complete. These individual trials come from blocks 1-8. Also included are quadratic curves fit to the data.

for trials where any participant was missing data (due either to an outlier completion time or to not being able to complete all trials), the result of which was complete data for 323 trials over blocks 1-8; the averages of the times for these individual trials are shown in fig. 7. Paired t-tests on the coefficients of the fitted quadratics showed statistically significant ($p = 0.0458$) differences between the average second-order coefficients for keyboard-card and dropdown menus (i.e.

there were statistically significant differences in the "curviness" of the data for the keyboard-card and dropdown menus). In addition, we performed a repeated measures ANOVA on the average completion times for blocks 1-7 (where all participants completed all trials), which showed statistically significant differences between the two menu types (p = .044) as well as between the blocks (p < .001).

Average error rates for each trial and menu type are shown fig. 6. Overall error rates across blocks 1-7 for the dropdown menus and keyboard-card menus were 10.4% and 8.4% respectively, where errors were defined as the incorrect selection of a leaf in the menu hierarchy and the error rate for a block was calculated as $e/(e+t)$ where $e =$ the number of errors in the block's non-outlier trials and $t =$ the number of non-outlier trials in the block. This difference was not statistically significant. However, t-tests on the data for the 5th, 9th and 10th block differences did show statistical significance (p-values were .032, .008, and .049 respectively; note that 9th and 10th blocks exclude data for one participant). Though a repeated measures ANOVA on the error rates for blocks 1-7 also failed to show statistical significance, fitting quadratics to the error rates for each participant at each block where data was available, and then performing t-tests on the coefficients *nearly* showed statistical significance at the 5% level for the quadratic and linear coefficients (p-values were .0544 and .0624 respectively)

Several interesting correlations can also be seen in the overall (i.e. across blocks) data for each participant. Error rates for dropdown menus were correlated (R =0.718) with the error rates for keyboard-card menus and overall average times for dropdown menus were correlated (R =0.911) with overall average times for keyboard-card menus. However, error rates and average times were negatively correlated for both dropdown menus and keyboard-card menus (R = -0.510 and -0.479, respectively), so it appears that some participants chose to concentrate more on being accurate, at the expense of speed, and some chose to do the opposite.

In addition, the difference between the error rates for dropdown and keyboard-card menus was positively correlated with the error rate for dropdown menus (see fig. 8) and the difference between the average times for keyboard-card and dropdown menus was positively correlated with the average times for dropdown menus (see fig. 9). In other words, participants whose error rates were higher made even more mistakes when using the dropdown menus, and participants whose average times were slow were even slower when using dropdown menus.

Results from the questionnaire can be seen in the table, where Q1="How quickly do you feel you were able to become familiar with the menu item locations over the course of performing the trials?", Q2="At the beginning of the set of trials for this menu system, how quickly do you feel you were able to select menu items?", and Q3="At the end of the set of trials for this menu system,

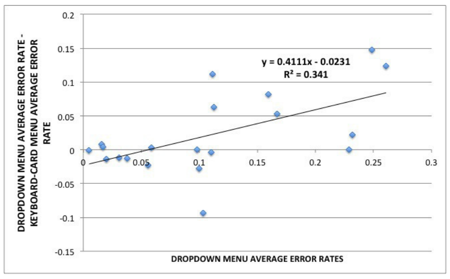

Figure 8: The differences between the overall average error rates for dropdown menus and keyboard-card menus, for each participant, plotted against the error rate for dropdown menus.

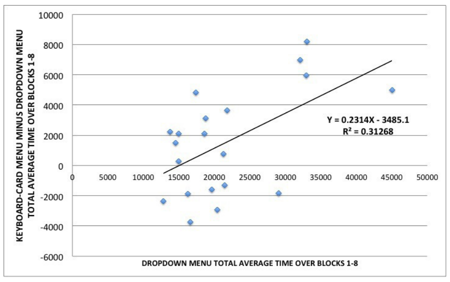

Figure 9: The differences between the overall average times for dropdown menus and keyboard-card menus, for each participant, plotted against the time for dropdown menus.

how quickly do you feel you were able to select menu items?" Ratings were given on a scale from 0="Very Slowly" to 9="Very Quickly". Though the results of these ratings were favorable for keyboard-card menus, t-tests did not reveal statistically significant differences between the average ratings (p-values given in the table). These results were not correlated with typing speed.

	KCM Ave.	**KCM Std. Dev.**	**DDM Ave.**	**DDM Std. Dev.**	**P-values**
Q1	6.0	1.5	4.8	2.4	.165
Q2	3.0	1.8	2.2	1.6	.119
Q3	7.3	1.3	6.7	2.1	.525

Table 1: Questionnaire Results.

In addition, two open-ended questions were given in the questionnaire: Q4 = "For this menu system, did you notice anything that made selecting items difficult? If so, please list and explain." and Q5 = "Having used both menu systems, which do you prefer? Why?". Q4 was repeated twice, once in the first half of the questionnaire (taken immediately after using first menu type) and once in the second half at the end of the session. Q5 was the last question in the second half of the questionnaire.

For Q5, 12 of the 20 participants wrote that they preferred the keyboard-card menu, while 8 wrote they preferred the dropdown menus (though with one of the 8, the explanation appeared to give an argument in favor of the keyboard-card menus, so a mistake may have been made in that case which would make the numbers 13 and 7). Explanations for the preference towards the keyboard-card menus included: "I seemed to pick up the shortcuts faster. It felt easier to use", "There was more visual association in the second", "...it gave me a visual of where I needed to type with the keyboard image on the screen. I thought I learned the second menu faster.", "I felt I got accustomed to it more easily [and] quicker. I believe I finished the 720 trials in the first faster.", "...it was easier for me to understand it", "the layout is more intuitive and it's easier on the eyes.", "...easier to become familiar with choices", and "Modeling where to press on the keyboard was helpful, as were the color shifts. It was also easier on the eyes (looking at it was less of a hassle)."

On the other hand, some of the explanations of preference from participants favoring the dropdown menus included: "I feel like I was faster with the second procedure. However I believe I made more mistakes along the way.", "The [keyboard-card menu] might be better for those that use the 'vulture' method of typing as it shows where the key is. The extra space in this layout just makes it take longer to read the flashing colors are distracting once the combinations are memorized.", "although it's slow starting out, I memorized the keys faster, plus the first one made me feel a little sick D: ", "more familiar with "dropdown" menu", "I like the first menu system because it takes up significantly less space, and it has more flexibility.", "because then I didn't try to look and check each time.", and "I was able to catch on quicker".

Answers to Q4 after using the keyboard-card menu included: "Had trouble w/ Kentucky –> might help to alphabetize the key strokes, e.g. 'D' for 'Delaware', "I was thrown off that Indiana was 'A' rather than 'I', etc. Also, it was kind of hard to read the words printed over the letters on the screen.", "I didn't like the change of colors from blue to red –> hard to get used to", "At the beginning the color changes were distracting toward my goal, but the visual layout soon became helpful.", "The visual distance between the prompt + the on-screen keyboard", "Keys close to each other", "I couldn't remember where half the towns were. Seemed to take a long time to locate some cities.", "I still wanted to sometimes hit the key that corresponded to the letter of the word.", "Same first letter for states would have made it easier", "The background letter images made it harder to read the words.", "That the names on the left disappeared so if you made the wrong selection you had to depress the key to continue scanning the list.", "Had to read from left to right at first.", "The words sometimes mix with the letter in the key", "Confusing at first", "expanding beyond the home row key set, such as using the key 'T'.", and "having to move up for colorado and right for kansas threw me off a little. fingers had gotten used to staying still".

Answers to Q4 after using the dropdown menu included: "Despite the number of trials, it took me forever to learn which letter corresponded to which state (some in the middle I didn't learn). Were menu items ordered in any way? I didn't really pick up on it if they were. Holding down one key and then pressing the other could be awkward when the keys were close together. Maybe the commonly-used items could have moved to the top of each menu to make it easier. Tasks were rapid-fire–I thought I was trying to go a little too fast sometimes", "With the state/city combination I was tempted to do the combination of keys that corresponded with the first letters of the word. ex Kentucky -> Fairfield I wanted to do K - F.", "There were way to many choices", "beginning letter of state [and] beginning letter of city", "I was inclined to associate letters with the names of the places (L - little creek) so the lack of matching made it more difficult. Also, when I had to shift from the standard typing position (G/H for instance) my response time slowed", "It's much more difficult to read off a dropdown list than with the keyboard layout. Also, the dropdown list is not in alphabetical order, which oftentimes mislead me to the wrong answer.", "One thing that I found difficult was when the letters of two options were close together on the menu. I kept on switching them around in my head and sometimes forgot which was which." , "I felt like my eyes were doing more work looking around the screen than [with the keyboard-card] trials. Also, having towns on keys which didn't correspond w/ their first letter made it difficult at first.", "My first instinct would be to hit 'L' for Louisiana rather than 'A', 'C' for Connecticut rather than 'D', etc. Also, the cities under each state's menu were not actually cities in these states, so it was hard to remember their names.", "It was hard to keep track of the Connecticut/Canaan and Connecticut/Cornwall difference. It was very difficult locating the names on the dropdown lists.", "They weren't in alphabetical order", "At the start, I had to scan the list to find out which key belonged to which city. That lost me some time", "The place to look for second menus keep changing and sometimes I would even have to spend time looking for where the menu is.", "Some menus were in alphabetical order & others weren't. Also as one key had to be held down this slowed me as I became more familiar & memorized combos because sometimes they weren't recognized when typed too fast if my finger didn't stay on the 1st key", and "Utah -Scofield were right next to one another [using the T and U keys], which made it awkward. Also, there was no alphabetic order to the menu, so it took a long time to find unfamiliar items.".

6 Discussion

The selection time data supports the hypothesis that the keyboard-card menu's presentation makes rolled-chord shortcuts easier to learn than does the dropdown menu's presentation. We further hypothesize, first, that the eventual convergence of the two curves in fig. 5 means that by the 12th block users had (mostly) memorized the 14 shortcuts (and so were not using either presentation), and second, that the initial divergence of the two curves indicates that the keyboard-card menu's presentation mostly helped users in finding and recalling shortcuts they had seen but not fully memorized.

Although in fig. 5 the separation between the curves appears to be small, it may be exaggerated considerably in more realistic applications by two factors. First, the period in which users have seen menu items but have not yet memorized their locations (i.e. the middle section of fig. 5) may be extended over a greater period of time (e.g. hours or days, rather than tens of minutes) affecting the number of menu selections needed for memorization. Second, the effect of errors on the average selection times would likely be much greater, since only the time needed to make an error, recognize it, and make a correct selection is included in the fig. 5 data; in realistic applications, the time to undo an error would also have to be included.

The negative correlation seen between error rates and average times shows users making a tradeoff between speed and accuracy. However, the data seen in fig. 8 and fig. 9 suggests that use of a keyboard-card menu instead of a dropdown menu can help both users who prefer speed to accuracy and users who prefer accuracy to speed.

After performing the study, we have come up with a relatively simple explanation for the differences between both the average time curves *and* the error rate curves, not among the four given in the "Study Motivation" section above: keyboard-card menus make it easier for users to check their knowledge of the shortcuts. For instance, suppose a user has gained only an intermediate level of experience with the shortcuts and, therefore, wants to check that the shortcut "D-[L]" is the one he wants to use. To check this using the dropdown menu system *without actually invoking the shortcut*, he would have to press and hold the D key and then visually scan through the list of items that appear until he either sees the L key in the list or the menu item itself. In contrast, after pressing the D key using the keyboard-card system, he can move his gaze much more directly to the position of the menu item associated with the L key because he knows where the L key is within the keyboard layout.

From the written responses to the questionnaire, several issues emerged as significant. First, the menu items were not organized either alphabetically or by item frequency. Second, how far users had to move their fingers to invoke a shortcut (e.g. the U key vs. the J key) mattered to them. Third, users' normal expectation is that the shortcut key used will be the same as the first letter of the menu item. Finally, some users found the bright flashing colors bothersome. How much these issues actually slowed users down, and how much they affect users' overall feelings about the shortcuts, remain to be investigated. However, they may be used as an initial set of heuristics in future keyboard-card menu applications (e.g. given a choice between associating a menu item with the U key or the J key, pick the J key).

7 Conclusion

While selecting from a large set of menu items in an efficient manner is certainly possible, expecting users to learn how to do so using traditional shortcuts displayed in dropdown menus is often unrealistic. In order to open up new realms of computer applications to a wider range of users (college students typing of mathematics at the computer, for instance), we are rethinking shortcuts and, in particular, how to present them. We have developed keyboard-card menus, and have tested them against dropdown menus as a way of presenting how to enter shortcuts, using rolled-chords as our shortcut choice in the experiment. Our data shows that keyboard-card menus, taking advantage of increasing amounts of available screen space with their keyboard-shaped presentation, can be more effective than a presentation based on dropdown menus as a method for presenting large numbers of shortcuts in an easy-to-learn way.

Acknowledgements

We would like to thank our participants for their time and effort. This research was supported by NSF award CCF-1250306.

References

[Ahlström et al. 1999] Ahlström, D., Cockburn, A., Gutwin, C., Irani, P.: "Why it's quick to be square: modeling new and existing hierarchical menu designs"; Proc. CHI, ACM (2010).

[Bailly et al. 2008] Bailly, G., Demeure, A., Lecolinet, E., Nigay, L.: "MultiTouch menu (MTM)"; Proc. of the 20th International Conference of the Association Francophone d'Interaction Homme-Machine, (2008), 165-168.

[Bailly et al. 2010] Bailly, G., Lecolinet, E., Guiard, Y.: "Finger-count & radial-stroke shortcuts: 2 techniques for augmenting linear menus on multi-touch surfaces"; Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM (2010), 591-594.

[Bau and Mackay 2008] Bau, O., Mackay, W. E.: "OctoPocus: a dynamic guide for learning gesture-based command sets"; Proceedings of the 21st annual ACM symposium on User interface software and technology, ACM (2008), 37-46.

[Bau et al. 2010] Bau, O., Ghomi, E., Mackay, W.: "Arpege: Design and Learning of multi-finger chord gestures"; Submitted to ACM TOCHI, (2010).

[Carroll and Rosson 1987] Carroll, J. M., Rosson, M.: "Paradox of the active user"; Interfacing thought: cognitive aspects of human-computer interaction, MIT Press (1987), 80-111.

[Graham et al. 1998] Graham, S., Berninger, V., Weintraub, N., Schafer, W.: "Development of handwriting speed and legibility in grades 1–9"; The Journal of Educational Research, 92, 1 (1998), 42-52

[Grossman et al. 2007] Grossman, T., Dragicevic, P., Balakrishnan, R.: "Strategies for accelerating on-line learning of hotkeys"; Proc. CHI, ACM (2010), 1591-1600.

[Hendy et al. 2010] Hendy, J., Booth, K., McGrenere, J.: "Graphically enhanced keyboard accelerators for GUIs"; Proc. GI '10, Canadian Information Processing Society (2010), 3-10.

[Klemmer et al. 2002] Klemmer, R. S., Thomsen, M., Phelps-Goodman, E., Lee, R., Landay, J. A.: "Where do web sites come from? Capturing and interacting with design history"; Proc. CHI, ACM (2002), 1-8.

[Krisler and Alterman 2008] Krisler, B., Alterman, R.: "Training towards mastery: overcoming the active user paradox"; Proceedings of the 5th Nordic conference on Human-computer interaction: building bridges, ACM (2008), 239-248.

[Kurtenbach 1993] Kurtenbach, G. P.: "The design and evaluation of marking menus."; PhD thesis, University of Toronto (1993).

[Kurtenbach and Buxton 1994] Kurtenbach, G., Buxton, W.: "User learning and performance with marking menus"; Proc. CHI, ACM (1994), 258-264.

[Kurtenbach et al. 1999] Kurtenbach, G., Fitzmaurice, G. W., Owen, R. N., Baudel, T.: "The Hotbox: efficient access to a large number of menu-items"; Proc. CHI, ACM (1999), 231-237.

[Lane et al. 2005] Lane, D., Napier, H., Peres, C., Sandor, A.: "The hidden costs of graphical user interfaces: The failure to make the transition from menus and icon toolbars to keyboard shortcuts"; International Journal of Human-Computer Interaction, 18, (2005), 133-144.

[Lepinski et al. 2010] Lepinski, G. J., Grossman, T., Fitzmaurice, G.: "The design and evaluation of multitouch marking menus" Proc. CHI, ACM (2010), 2233-2242.

[Lyons et al. 2003] Lyons, K., Patel, N. J., Starner, T.: "KeyMenu: A keyboard based hierarchical menu"; Proc. ISWC '03, IEEE Computer Society (2003), 240-241.

[Malacria et al. 2013] Malacria, S., Bailly, G., Harrison, J., Cockburn, A., Gutwin, C.: "Promoting hotkey use through rehearsal with exposehk"; Proc. CHI, ACM (2013), 573-582.

[Malik 2007] Malik, S.: "An exploration of multi-finger interaction on multi-touch surfaces"; Diss., University of Toronto (2007).

[Norman 1991] Norman, K. L.: The Psychology of Menu Selection: Designing Cognitive Control at the Human/Computer Interface; Ablex Publishing Corporation, Norwood, NJ (1991).

[Purdy 2012] Purdy, K.: "Make Chrome Less Distracting with Vimium (and These Settings)"; Lifehacker (July 11, 2012) `http://lifehacker.com/5925220/make-chrome-less-distracting-with-vimium-and-these-settings/`

[Scarr et al. 2011] Scarr, J., Cockburn, A., Gutwin, C., Quinn, P.: "Dips and ceilings: understanding and supporting transitions to expertise in user interfaces"; Proceedings of the 2011 annual conference on Human factors in computing systems, ACM (2011), 2741-2750.

[Shneiderman and Plaisant 2010] Shneiderman, B., Plaisant, C.: Designing the User Interface: Strategies for Effective Human-Computer Interaction, 5th Ed.; Addison-Wesley, Boston (2010).

[Varcholik et al. 2012] Varcholik, P. D., LaViola, J. J., Hughes, C. E.: "Establishing a baseline for text entry for a multi-touch virtual keyboard"; International Journal of Human-Computer Studies, Special issue on Developing, Evaluating and Deploying Multi-touch Systems, 70, 10 (2012), 657-672