# Performance of Menu-Augmented Soft Keyboards

**Poika Isokoski**
Department of Computer Sciences
FIN-33014, University of Tampere, Finland
poika@cs.uta.fi

**ABSTRACT**
We report results on the performance of the combination of soft keyboards and marking menus. A model of expert user performance indicated an 11 - 37% (depending on the keyboard layout) improvement in text entry rate over the same keyboard without the menu. To verify the advantage in real usage, we conducted two experiments using the QWERTY keyboard layout with and without the menu. The first experiment imitated nearly perfect cognitive performance and measured motor performance. Using the menu saved time. The second experiment measured performance in a realistic text entry task. Initially using the menu slows down text entry. By the end of the 20-session experiment both conditions were equally fast. With continued practice text entry is likely to be faster with the menu.

**Categories and Subject Descriptors**
H5.2 [Information Interfaces and Presentation]: User Interfaces - Evaluation/Methodology, Input Devices and Strategies.

**Keywords**
text entry, Fitts' law, marking menu, soft keyboard

## INTRODUCTION

A large number of text entry systems for computers without a full sized keyboard have been proposed recently (see [11] for an overview). Some of these utilize combinations of soft keyboards and menus. We call such systems menu-augmented soft keyboards. Before describing the new system that is our main topic, we will give a brief overview on those systems that seem, in retrospect, to be its most obvious ancestors.

### Previous Work

T-Cube by Venolia and Neiberg [20] is a well-known example of text entry with popup menus. It is operated by landing the stylus on a stationary ring-shaped menu and popping up a radial menu from which a character is selected by moving the stylus to the direction of a slice and lifting it. Another example of the use of popup menus is the POBox system [14]. It uses popup menus in the context of soft keyboards to present most likely word completions.

Work on T-Cube and POBox was in the context of pen-based computing. The aim was to increase text entry speed in comparison to other available systems. Later efforts with the same goal [16, 4] were in the context of mobile phones. The 9-slice ring-shaped keyboard of T-Cube was replaced with the 12 keys of the telephone keypad. Naturally, when used with the physical keypad popup menus are not drawn. Instead, hints of the available key combinations are printed on the keys. These systems can also be used with a stylus and a software-rendered keyboard [16].

Besides for speeding up text entry, menus have been used to modify the character being entered. For example Microsoft's soft keyboards in some Handheld computers (such as Compaq iPAQs) allow entering upper case characters by landing the stylus on a key and then sliding it upwards before lifting. Additionally, the recent Fitaly versions [19] allow entering the accented variations of characters by landing on a key (say 'a') and then sliding to different directions to choose characters (like 'å', 'ä', 'â', 'á' and 'à'). Left and right slides can also be mapped to backspace and delete.

### The Present Study

The goal of our work was to evaluate a new stylus-based text entry system, where a popup menu is added to a regular soft-keyboard to increase text entry speed. A recent student project with the same goal [6] suffered from unreliable software, but produced useful findings that have been taken into account in our implementation.

We examined menu augmented soft keyboards in three phases. First, we simulated their use with models based on Fitts' law and previously published data on menu selection performance. Then, we verified the modeling result in an experiment that measured the text entry capacity of the user's motor system with and without the menu. Finally, we undertook a longitudinal experiment where the users' performance was measured in a somewhat realistic task. Before reporting the results, we explain our menu-augmented soft keyboard in more detail.

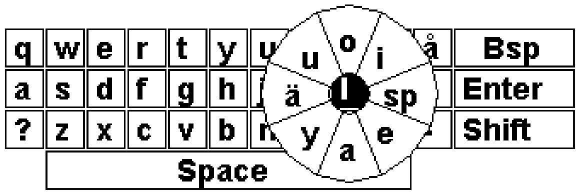


Figure 1. A QWERTY soft keyboard with a menu.

## MENU-AUGMENTED SOFT KEYBOARD

### The Basic Idea

A soft keyboard with a menu is shown in Figure 1. When the user sets the stylus down on a key ("l" in Figure 1), the corresponding character is entered. After that the user has two choices: to lift the stylus, or to do a marking-menu selection to enter another character.

Marking menus [7] are radial menus that pop up after a delay. The user selects a menu item by moving on it and then lifting the stylus. It is not necessary to wait for the menu to pop up. The selection can be done immediately by drawing the gesture as if the menu were visible.

The savings in stylus travel are illustrated in Figure 2. The upper part shows the stylus movements without the menu and the lower part with the menu on the left side of Figure 3. The text written in the example is "happy". In both cases writing starts by landing the stylus on "H". Then, if the menu is not available, the stylus is lifted and moved over "A". When the menu is available, "A" is selected by sliding the stylus down from "H". Another big saving occurs in the end when the long P-Y movement is replaced by selecting Y from the southwestern menu slice.

The menu does not interfere with normal use of the soft keyboard. Users that do not want to use it can ignore it.

### Menu Design

Throughout this paper we use menus with vowels, space and backspace in them. According to the digraph frequency table in [17] the English vowels and space constitute 49.9% of English text. However, because two consecutive characters cannot be entered using the menu, the menu on the left side of Figure 3 can be used to enter about 35% of the text (note that this means that about 54% of taps on the keyboard end in a menu selection).

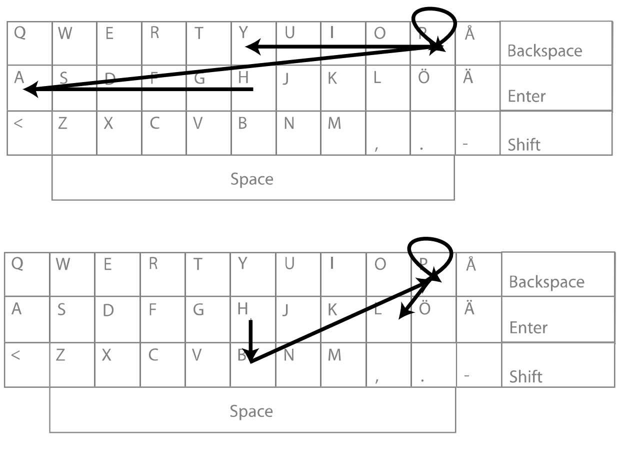

Q W E R T Y U I O P Å Backspace
A S D F G H J K L Ö Ä Enter
< Z X C V B N M , . - Shift
Space

Figure 2. Stylus travel without a menu and with it when writing "happy".

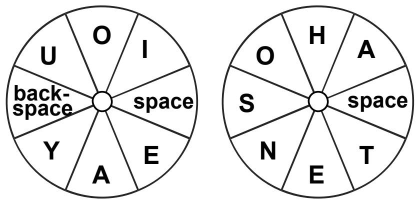


Figure 3. A menu with vowels (left) and an alternative design with the most frequent characters (right).

Instead of vowels the menu could be populated with, the most frequent characters. This tends to increase the frequency of digraphs with both characters in the menu, which means that one of the characters cannot be entered using the menu. Only about 26% of characters can be entered using the menu on the right in Figure 3.

Possibly more optimal menus than the one on the left in Figure 3 can be constructed. However, we expected the vowels, backspace and space to be an easy combination to remember and therefore better than many others.

Each key could have its own popup menu with the most probable following characters. In practice this leads to difficulties in learning to utilize the menu as exemplified by the Fluctuating Optimal Character Layout (FOCL) work [1, 9]. In an attempt to reap most of the benefits of the menu without the disadvantages we decided to experiment with a system where the menu is always the same.

We chose the breadth of the menu based on the results of Kurtenbach *et al*. [7] and McQueen *et al*. [15]. Eight slice menus seemed optimal because a greater number leads to degradation of both selection speed and error rate.

## MODEL

### Goal and Method of Modeling

The purpose of the modeling was to reveal the relationship between the motor efficiency of a soft keyboard with a menu and without it. The results were two-fold. First, we found a limit under which the menu selection time has to be in order to yield performance benefits. Second, we used previously measured menu selection times to compute the difference that we can expect the menu to make in text entry rate.

Models based on Fitts' law assume that the motor control over the stylus is the bottleneck that limits the user's performance. Consequently the results are applicable only to nearly perfectly trained expert user populations. We accepted this limitation and explored the non-expert performance experimentally.

We used the five keyboard layouts shown in Figure 4. Two of them were designed for two-handed use (Dvorak [2] and QWERTY) and three for stylus tapping (FITALY [19], OPTI II [12], and ATOMIK [21]).

QWERTY is the most popular of these layouts. It is the default layout on almost all systems that have a soft keyboard. The other layouts are research prototypes or commercially available systems with limited success. The dominant position of the QWERTY layout means that the immediate usefulness of menu-augmentation depends on whether it improves text entry rates with QWERTY.

Both traditional soft keyboard optimization and menu-augmentation aim to reduce stylus travel. It is interesting to compare the effectiveness of these methods and to see whether optimized soft keyboard layouts can be further improved by adding the menu.

### Fitts-digraph Model

Elementary keyboard optimization work was discussed by Card *et al.* [3] (example 5, p. 56), but the digraph table for modeling text entry was added later by Soukoreff and MacKenzie [17]. The Fitts-digraph model is widely used [11, 12, 13, 16, 21, 22, 23] tool for modeling expert stylus tapping performance.

The model works by breaking writing into elemental pointing tasks and then modeling each of these with Fitts' law. While the basic rationale of the model is generally accepted, some details of it vary in the literature. Our choices regarding these details were as follows:

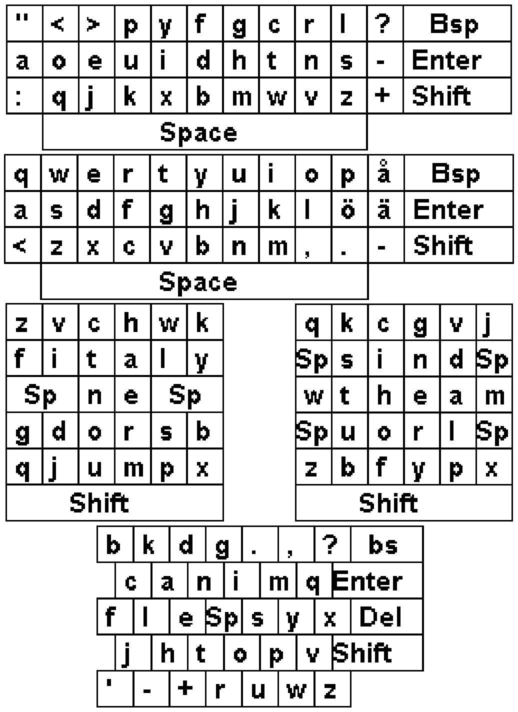


**Figure 4. The keyboard layouts used in this study (from top Dvorak, QWERTY, FITALY, OPTI II, and ATOMIK)**

**Fitts' law formula**: We used the Shannon formulation introduced by MacKenzie [8]. That is, movement time equals to $a + b \log_2 (A / W + 1)$.

**Key repeat**: We modeled repeating taps on the same key with the same Fitts' law equation as taps on different keys. Because the distance to be traveled is zero, the repeat taps are modeled with the intercept ($a$) value of the formula above. Repeating key presses are rather rare (1.9% in the table by Soukoreff and MacKenzie [17]). Thus, repeat key handling does not have a large effect on the results.

**Intercept value**: Sometimes the Fitts' law intercept value is set to zero [12, 13, 17], sometimes it is measured from experimental data [23]. We used a non-zero intercept. This fits with our choice of key repeat modeling.

**Multiple key instances**: Some layouts have more than one instance of the same key. The assumption is that the user chooses the closest one. Our model used the target key and the preceding key to decide which instance to choose.

**Wide keys**: Some keys have a significantly elongated shape. This means that using the center point of the key as the Fitts' law target in the model produces unrealistic results. We split the wide keys into several keys with width equal to that of the smallest key in the keyboard. As described above, the model selected the one that is closest to the preceding key.

**Key position and shape**: Our choices of key positions and shapes can be seen in Figure 4. One of the most notable choices is the shape of keys in ATOMIK. Originally they were hexagonal. Our version with rectangular keys is based on the version shown in Figure 26 by Zhai *et al.* [21].

### Menu Selection Model

Entering a pair of characters using the menu for the second one consists of two phases. First, the stylus is moved from a starting point onto the key of the first character. Second, it is dragged over one of the menu slices and lifted.

One way to model this is to use Fitts' Law for both phases (as, for example, in [16]). Unfortunately, the phases have interactions that limit the accuracy of this approach. For example Venolia and Neiberg [20] observed that there were differences in the overall task completion times depending on both the menu item direction and the location of the stylus on the keyboard. For example menu selection to the southeast was faster in the southeastern part of the keyboard.

Although menus have been studied extensively, most published results are either on mouse or keyboard use, or only report total selection times after a stimulus. Since we did not have data that would have allowed accurate treatment of all the subtleties, we used a constant movement time that was the same for all menu selections.

Although we used a simplified menu selection model, we still needed to know the size of the stylus movements because we needed a starting point for the following movement. We set the menu radius to be twice the width of a regular key in our keyboards. The endpoint of the menu selection was in the middle of the menu slice. These parameters are reasonable because the menu slices were big enough to contain the labels and users might be inclined to aim at the center of the menu item.

### Model Implementation

We built our model in the form of simulation software. It uses keyboard and menu layout descriptions, Fitts' law model, menu selection model, and text files to simulate stylus movement over the keyboard. The simulator was implemented in Java.

In summary, the model knows two primitive actions: tapping a key (modeled with Fitts' law), and menu selection (modeled by a constant time move to the center of the appropriate menu slice). It reads a text stream character by character and performs the keyboard actions needed for replicating the text.

## MODELING RESULTS AND DISCUSSION

### Upper Limit for Menu Selection Time

A simple way to estimate whether the menu can make text entry faster is to find out how long a menu selection can take and still yield performance benefit. If the numbers look good, further work is warranted. To find this limit we selected values for Fitts' law intercept and slope and ran our simulation software using a small text corpus (published by MacKenzie and Soukoreff [10]) to find out how long entering that text took without the menu functionality. Then we ran several more simulations with the menu enabled with varying menu selection time to do a binary search for the point where the menu no longer helped.

To observe how the crossover point behaves under different Fitts' law coefficients, we ran the simulation software to find a 20 by 20 matrix of the crossover points with both coefficients (*a* and *b*) ranging from 0 to 0.19.

This produced results like the one shown in Figure 5. Given values for Fitts' law intercept (*a*) and slope (*b*), the menu selection time needs to be below the plane in Figure 5 to save time. The plane is described by the equation $t = a+bk$. Where $t$ is the critical menu selection time, $a$ and $b$ are the Fitts' law coefficients used as simulation parameters, and $k$ is a coefficient whose value depends on the keyboard and

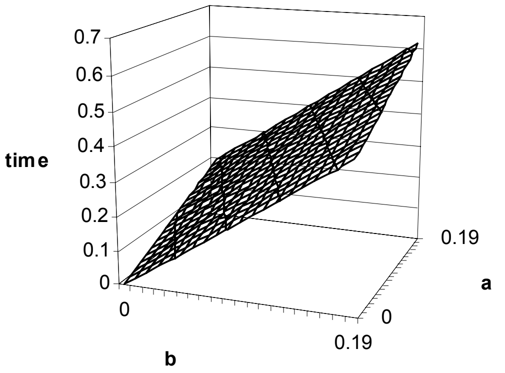


**Figure 5. The upper limit for useful menu selection time for the QWERTY layout.**

menu layout. The values for $k$ with the simulated keyboard layouts are shown in Table 1. These values let us compute estimates for the upper limit of useful menu selection time. As an example we used the Fitts' law coefficients measured by Zhai *et al.* [23] as shown in Table 1.

### Text Entry Rate

To find out how much the menu speeds up text entry, we turned back to the simulator. We used the menu selection times measured by McQueen *et al.* [15]. They report times broken down to preparation time and scripting time. Preparation time means the time when the stylus is not touching the display. This portion is modeled by the Fitts' law part of our model. Scripting time means the time between the stylus landing and lifting. This is roughly equivalent to the menu selection time component in our model. The results of McQueen *et al.* are for a 12-slice pie menu. The accuracy constraints in our 8-slice menu are lower allowing higher selection speed with the same error rate. The scripting time reported by McQueen *et al.* is 0.3 seconds in the beginning of the experiment of twenty 15-minute sessions and about 0.16 seconds at the end.

We used the number measured by the end of the experiment because we were modeling expert performance. Simulations with this menu selection time and the Fitts' law coefficients by Zhai *et al.* [23] gives the numbers shown in Table 2. The differences are substantial enough to show in experimental results at least with the Dvorak and QWERTY layouts.

**Table 1. Values for k and the upper limit of useful menu selection time for five keyboard layouts.**

| Keyboard | k | Upper limit |
|---|---|---|
| Dvorak | 2.61 | 0.083 + 0.127 x 2.61 = **0.41** s |
| QWERTY | 2.08 | 0.083 + 0.127 x 2.08 = **0.35** s |
| OPTI II | 1.35 | 0.083 + 0.127 x 1.35 = **0.25** s |
| ATOMIK | 1.21 | 0.083 + 0.127 x 1.21 = **0.24** s |
| FITALY | 1.18 | 0.083 + 0.127 x 1.18 = **0.23** s |

**Table 2. Simulated text entry rates with and without the menu.**

| | Text entry rate (words per minute) | | |
|---|---|---|---|
| Keyboard | No menu | Menu | Difference |
| Dvorak | 33.9 | 46.5 | 12.6 (37%) |
| QWERTY | 36.7 | 46.3 | 9.6 (26%) |
| OPTI II | 43.8 | 50.2 | 6.4 (15%) |
| ATOMIK | 44.2 | 49.5 | 5.5 (12%) |
| FITALY | 44.7 | 49.8 | 5.1 (11%) |

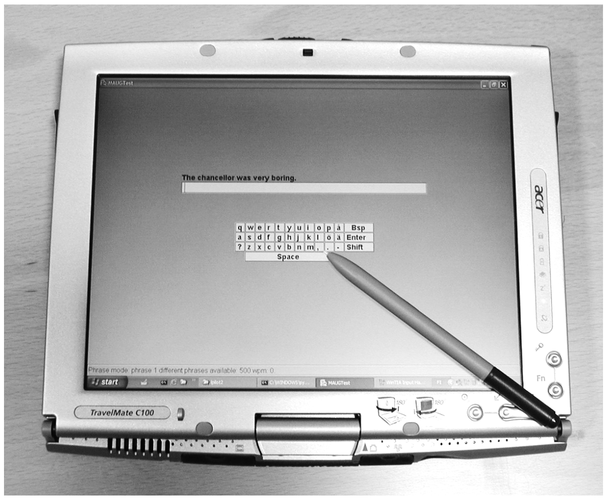


Figure 6. The computer and stylus used in the experiments.

The rank order of the keyboards in the no menu condition in Table 2 differs from previously published results. This is a matter of some concern. As detailed above, our modeling assumptions differ from previous work for example in the handling of the space keys. In addition, the previously published results have been computed using different text corpora. All the used corpora are argued to be representative of typical English but small differences do exist.

## EXPERIMENT 1

To do a reality check on our modeling results, we compared character entry speeds experimentally in two conditions: with a plain keyboard and with a menu-augmented keyboard.

### Method

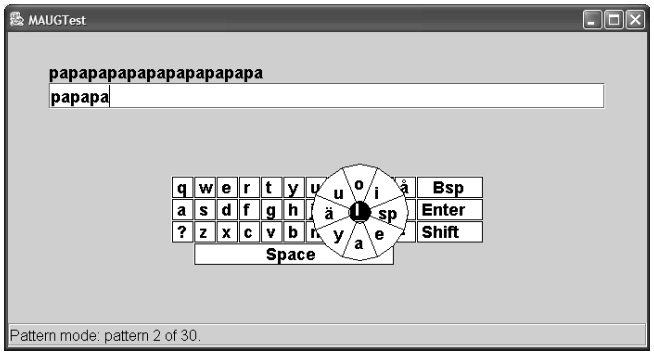
#### *Participants*

12 participants (8 male, 4 female) were recruited from within our research unit. The age of the participants ranged from 23 to 34 years (mean 26, SD=3.2). All were experienced computer users who had seen and operated a computer with a stylus. None, however, had used soft keyboard for text entry for extended periods. Similarly none had hands-on experience with marking menus.

#### *Apparatus*

We used an Acer TravelMate C102Ti laptop running Microsoft Windows XP TabletPC Edition. The stylus functionality in this model is implemented using a Wacom 10.4” module with the styli sensed using electromagnetic resonance. We used a stylus that is approximately the size of a regular ballpoint pen (140 mm long and 8 mm thick). The computer and stylus are shown in Figure 6.

The software was written in Java. Because mouse events are inherently unreliable in Windows (some get dropped if the computer is busy), and we did not find a way to access the Microsoft pen API from Java, we implemented the experimental software as a module in a text input architecture [5] that uses C++ code to access DirectInput8 API. The timestamps in the DirectInput events are propagated through the system. This excludes possible delay and delay jitter due to Java garbage collection and scheduling.


MAUGTest

papapapapapapapapapapapa

papapa

Pattern mode: pattern 2 of 30.

Figure 7. The task display in the pattern task.

The display of our software is shown in Figure 7. During the experiment the window was maximized to fill the whole screen. The mouse cursor was not shown over the keyboard. The width of the alphabet keys and the height of all keys was approximately 4.9 mm. A popular trick in soft keyboards is to show small keys that have a larger active area. This fools the users to aim more accurately at the center of the key. We utilized this trick too. The active areas of the keys fill the whole keyboard, but the graphical representations are separated by small gaps as seen in Figure 7. The calibration error between the stylus tip and the measured coordinates was typically low. The largest errors over the keyboard were in the order of 1.5 mm.

#### *Procedure and Design*

To make the task cognitively easy, patterns of 2-4 characters were entered repetitively. There were a total of 30 different patterns. Each pattern was repeated 11 times in a string like the one shown in Figure 7. The patterns were chosen to represent a wide range of key distances on the QWERTY layout and varying movement directions. The patterns also required the use of all of the menu items. Each participant entered the block of 30 patterns four times: once for practice and once for data collection with and without the menu. The order of the menu conditions was balanced between subjects but the order of the patterns within a block was always the one shown in Figure 8.

The software accepted only correct input. Incorrect characters caused a sound to be played. Upon hearing the sound a participant had to either keep entering the pattern to resynchronize on the next repetition or stop, find the error, and enter the correct character.

The design was within subjects with one factor with two levels (menu and no menu). The dependent variable was the time spent per character.

## Results

### *Data Considerations*

Data for pattern instances with errors in them were excluded because we were interested in fluent and correct performance only. The average exclusion rate varied between participants and patterns. However, at least 4 out of the 10 possible (the first one was always excluded) repetitions per pattern per participant were correct. On average 95% of the repetitions were included.

### *Time Per Character*

The per character timing results are shown in figure 8. Using the menu seemed to pay off, except when the distance between the keys in the no menu condition was very small. Even then using the menu was approximately equally fast. With the simple patterns that allowed menu use ("pa"-"no" in Figure 8) the time for the menuless condition varied depending on the distance between the keys. When the menu was used the time per character was around 200 milliseconds varying only slightly depending on the menu item chosen.

In the more complex patterns ("pal"-"päl" and "pala"-"pälä") the difference between the conditions was smaller. The task began to become cognitively difficult thus increasing the task time in a way that does not reflect the motor demands of the task.

In real text entry tasks the performance bottleneck is almost completely on the cognitive side. Especially in the beginning when users have to attend to planning of the menu selections. Therefore, our results so far have not really proven the value of adding a menu to soft keyboards. We have merely seen that when motor performance is the bottleneck, the menu helps. This is why we needed to conduct another experiment.

## EXPERIMENT 2

The goal of the second experiment was to find an estimate for how long it takes for users learn the menu usage well enough to take advantage of the smaller motor load.

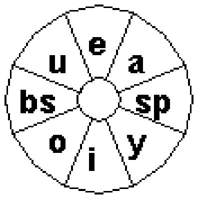


**Figure 9. The menu used in experiment 2.**

### Method

#### *Participants*

6 volunteers (5 male, 1 female) were recruited from our research unit. The ages ranged from 23 to 30 years (mean=25.5, SD=2.7). All were experienced computer users, but none were regular soft keyboard users. None of the participants were native speakers of English, but all had a good skill and used English in their daily work.

#### *Apparatus*

The hardware and software used in experiment 2 was the same as in experiment 1 except that the menu was partially optimized using the data measured for the two-character patterns in experiment 1. As shown in Figure 9, backspace and space are in the leftmost and rightmost slices. The vowels have been re-located according to their frequency so that the most frequent "e" is in the fastest remaining location and the least frequent "y" is in the slowest location.

#### *Procedure and Design*

The experiment consisted of 20 sessions per participant. Each session consisted of two 15-minute sub-sessions. One of these was for text entry with the plain QWERTY soft-keyboard and the other for using the same keyboard with the menu enabled. The order of the sub-sessions was reversed for each session to reduce the effect of learning and fatigue towards the end of the session.

The task was to enter the presented phrase as fast as possible while making as few errors as possible and then to press the enter key. Pressing the enter key erased the written text and caused a new phrase to be randomly selected and presented. The phrase set we used was a modified version of the set of 500 English phrases

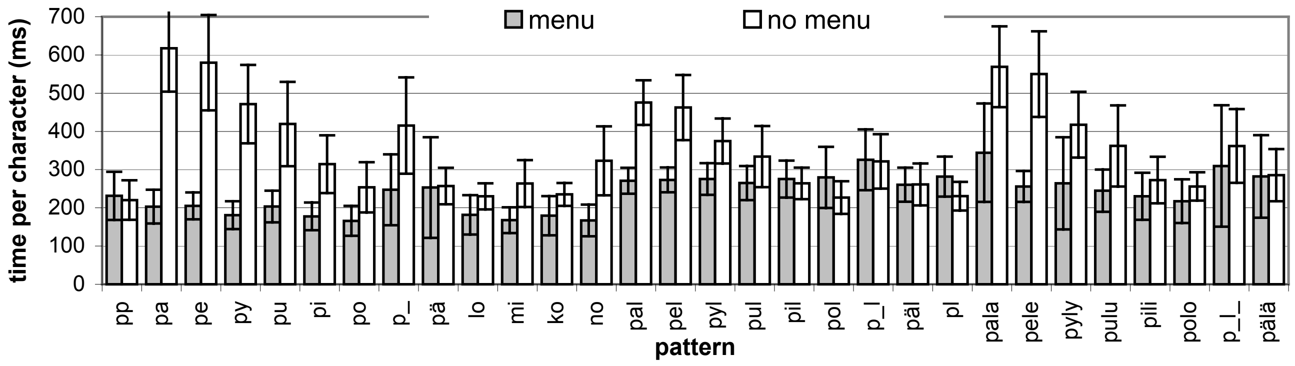


**Figure 8. Mean time per character for the 30 patterns in the experiment (error bars show ± one standard deviation).**

published by MacKenzie and Soukoreff [10]. The modifications consisted of adding upper case characters and punctuation to where it seemed grammatically appropriate.

The design of the experiment was a within subjects repeated measures design with session (1-20) and the availability of the menu (menu, no menu) as independent factors.

### Results

Because the task could be accomplished without using the menu, we needed to verify that the participants actually used the menu in the menu condition. The percentage of characters entered using the menu rose from 23% during the first session to 29% by the last session. Thus, the menu was indeed used in the menu condition.

The two systems were not equally fast ($F_{1,5}$=29.0, p<0.01). As seen in Figure 10 the menuless system was initially faster. The effect of the session was also significant ($F_{19,95}$=66.2, p<0.0001) indicating that performance improved over time. The interaction of system and session was significant as well ($F_{19,95}$=30.4, p<0.0001) meaning that that learning progressed at different rates on different systems. In the end both systems were approximately equally fast. The best-fitting power curves in Figure 10 suggest that had the experiment continued the system with the menu might have overtaken the menuless system in speed.

For error rate calculations we used the MSD/KSDPC approach of Soukoreff and MacKenzie [18]. Corrections while writing were allowed. Thus, comparing the presented and transcribed text using the minimum string distance (MSD) reveals only those errors that were left after corrections. This is why the ratio of needed characters and executed keystrokes needs to be computed. This measure is called Keystrokes per character (KSPC). Larger KSPC means more error correction activity. ANOVAs on MSD and KSPC figures revealed no significant differences between the systems or significant effects of session or the interactions of these. The mean error rate after corrections was 0.96% for the menu condition and 0.59% for the no menu condition. The corresponding KSPC values were 1.097 and 1.094.

Figure 11 shows the time spent per character broken down into time spent with the stylus down (downtime) and stylus up (uptime) for three different situations. First, when the menu is used (menu), second, when the menu is available, but not used for this character (tap), and third when menu is not available (no menu tap). This breakdown reveals that by the end of the experiment all downtimes are short (95- 151 ms). The big differences are in uptimes. The uptime of the menuless condition (no menu tap) is much lower than in the menu condition (menu and tap). Apparently the users were spending more time (170 ms more) in planning their next move when the menu was available.

## DISCUSSION AND CONCLUSIONS

Based on our simulations of user users' motor behavior it seemed that adding a marking menu to a soft keyboard would speed up text entry. A measurement of character entry speed in a simplified test situation supported this finding. Finally, a longitudinal experiment with a more realistic text entry task showed that the simulation results may hold for expert users, but reaching that level of expertise takes a long time. This should not be taken as a very discouraging finding. All text entry systems require extensive user learning. In comparison to some other speedy text entry systems, adding the menu seems attractive because it is compatible with traditional soft keyboarding. Users who do not want to use the menu do not need to know that it exists.

The menu design needs to be further explored. There may be better menu designs than the vowel menu we used. An appropriate size for the menu and the central "dead zone" should be found. Also, backspace in the menu was found to be useless. Something useful should take its place.

A menu-augmented soft keyboard needs some empty space around it so that all menu items can be selected even after landing on the peripheral keys. Whether this is acceptable

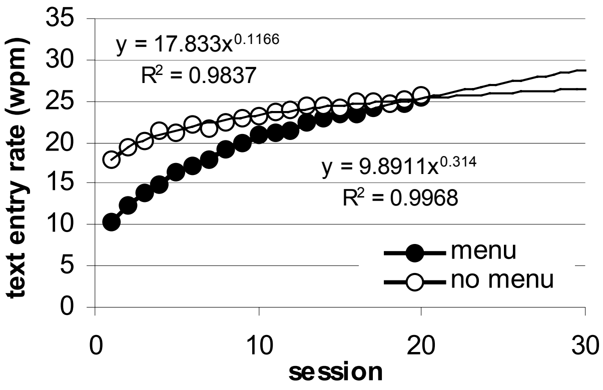


**Figure 10. Average text entry rate and extrapolations up to session 30 as a function of the session number.**

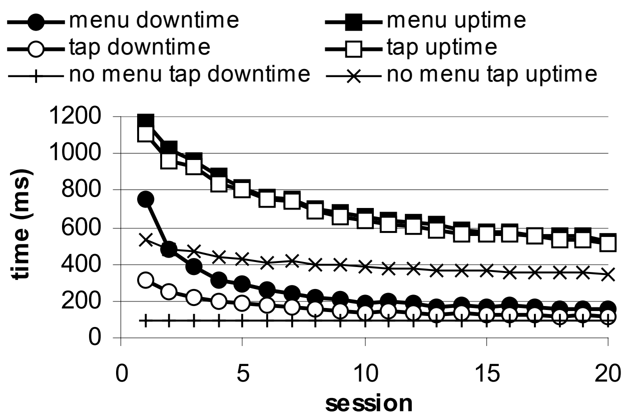


**Figure 11. Average per character stylus down and stylus up durations.**

in small devices remains to be seen. Some of the surrounding space can be used for placing infrequently used keys such as function keys and numeric keys that are unlikely to be followed by a menu selection.

The text entry rates we measured in experiment 2 were lower than others have reported. For example MacKenzie and Zhang [12] reported QWERTY tapping rates up to 40 wpm in a similar experiment. Possible reasons for the difference include our use of upper and lower case characters and punctuation as well as not allowing breaks during the half-sessions, allowing corrections, having non-native speakers writing English, a larger phrase set (MacKenzie and Zhang had 70 phrases), and shorter training sessions.

In conclusion, the menu-augmented soft keyboard is likely to be faster in expert use with the QWERTY layout. Because the QWERTY layout is widely used and the menu does not harm its normal operation, it may be worthwhile to add the menu to future soft keyboard implementations.

## ACKNOWLEDGMENTS

We thank Grigori Evreinov for the seminal discussions and Scott MacKenzie for comments in early phases of the project. This work was funded by the Tampere Graduate School on Information Science and Engineering (TISE) and Academy of Finland (grant 73987).

## REFERENCES

1. Bellman, T., and MacKenzie, I. S. A probabilistic character layout strategy for mobile text entry. *Proc. Graphics Interface '98*, Canadian Information Processing Society (1998), 168-176.
2. Brooks, M., Introducing the Dvorak Keyboard. http://www.mwbrooks.com/dvorak/
3. Card, S. K., Moran, T. P., and Newell, A. *The Psychology of Human-Computer Interaction*, Lawrence Erlbaum (1983).
4. Gnatenko, V. Multi-directional Input Keypad (the MIK) - text input solution for mobile devices. http://www.vitgn.com/
5. Isokoski, P., and Raisamo, R. Architecture for Personal Text Entry Methods, in Morten Borup Harning and Jean Vanderdonckt (Eds.), *Closing the Gap: Software Engineering and Human-Computer Interaction*, IFIP ( 2003), 1-8. http://www.se-hci.org/bridging/interact/
6. Jhaveri, N. Two Characters per Stroke – A Novel Pen-Based Text Input Technique, in Grigori Evreinov (ed.), *New Interaction Techniques 2003 (Report B-2003-5)*, Department of Computer and Information Sciences, University of Tampere (2003) 10-15. http://www.cs.uta.fi/reports/bsarja/B-2003-5.pdf
7. Kurtenbach, G., and Buxton, W. The limits of expert performance using hierarchic marking menus. *Proc. of INTERCHI 1993*, ACM Press (1993), 482-487.
8. MacKenzie, I. S. A note on the information-theoretic basis for Fitts' law. *Journal of Motor Behavior, 21* (1989), 323-330.
9. MacKenzie, I. S. Mobile text entry using three keys, *Proc. NordiCHI 2002*, ACM Press (2002), 27-34.
10. MacKenzie, I. S., and Soukoreff, R. W. Phrase sets for evaluating text entry techniques. *Ext. Abstracts CHI2003*, ACM Press (2003), 754-755.
11. MacKenzie, I. S., and Soukoreff, R. W. Text entry for mobile computing: Models and methods, theory and practice. *Human-Computer Interaction*, 17 (2002), 147-198.
12. MacKenzie, I. S., and Zhang, S. Z. The design and evaluation of a high performance soft keyboard. *Proc. CHI '99*, ACM Press (1999), 25-31.
13. MacKenzie, I. S., Zhang, S. X., and Soukoreff, R. W. Text entry using soft keyboards. *Behaviour & Information Technology*, 18 (1999), 235-244.
14. Masui, T. An Efficient Text Input Method for Pen-based Computers, *Proc. CHI '98*, ACM Press (1998), 328-335.
15. McQueen, C., MacKenzie, I. S., and Zhang, S. X. An extended study of numeric entry on pen-based computers. *Proc. Graphics Interface '95*, Canadian Information Processing Society (1995), 215-222.
16. Nesbat S. B. Fast, Full Text Entry Using a Physical or Virtual 12-Button Keypad. http://www.exideas.com/ME/whitepaper.pdf
17. Soukoreff, R. W., and MacKenzie, I. S. Theoretical upper and lower bounds on typing speeds using a stylus and soft keyboard. *Behaviour & Information Technology*, 14 (1995), 370-379.
18. Soukoreff, R. W., & MacKenzie, I. S. Measuring errors in text entry tasks: An application of the Levenshtein string distance statistic. *Ext. Abstracts CHI2001*, ACM Press (2001), 319-320.
19. The Fitaly Keyboard http://www.fitaly.com/fitaly/fitaly.htm
20. Venolia, D., and Neiberg, F. T-cube: A Fast, Self-Disclosing Pen-Based Alphabet. *Proc. CHI '94*, ACM Press (1994), 263 - 270.
21. Zhai, S., Hunter, M., and Smith, B.A. Performance Optimization of Virtual Keyboards, *Human-Computer Interaction*, 17 (2002), 229-270.
22. Zhai, S., Hunter, M., and Smith, B. A. The Metropolis keyboard - An exploration of quantitative techniques for virtual keyboard design, *Proc. UIST 2000*, *CHI Letters* 2(2), ACM Press (2000), 119-128.
23. Zhai, S., Sue, A., and Accot, J. Movement Model, Hits Distribution and Learning in Virtual Keyboarding, *Proc. CHI 2002*, *CHI Letters* 4(1), ACM Press (2002), 17- 24.