Mobile Text Entry Using Three Keys

Scott MacKenzie

Dept. of Computer Science
York University
Toronto, Ontario, Canada N3J 1P3
+1 416-736-2100
smackenzie@acm.org

ABSTRACT

Six techniques for three-key text entry are described. The techniques use Left- and Right-arrow keys to maneuver a cursor over a linear sequence of characters, and a Select key to select characters. The keystrokes per character (*KSPC*) for the methods varies from 10.66 to 4.23. Two techniques were chosen for formal evaluation. Method #2 positions characters in alphabetical order, while Method #6 uses linguistic enhancement to reorder characters following each entry to minimize the cursor distance to the next character. Both methods position SPACE on the left and use a snap-to-home cursor mode, whereby the cursor snaps to SPACE after each entry. Entry rates were about 9-10 wpm for both techniques, as measured in an experiment with ten participants. Interaction issues are examined, such as the challenges in using linguistic knowledge to accelerate input, and the opportunity for using typamatic (viz. auto-repeat) keying strategies to reduce the number of physical keypresses.

Keywords

Mobile text entry, linguistically enhanced text entry, text entry performance evaluations, typamatic keying

INTRODUCTION

Current research in mobile text entry includes significant interest in the use of small physical keyboards. This is fueled in part by the phenomenal success of so-called *SMS messaging* on mobile phones. The ability to discretely, asynchronously, and at very low cost, send a message from one mobile device to another has proven hugely successful, particularly in Europe. And the statistics are staggering: Volumes are now approaching 1 billion messages per day! [5] Given the limited capability of the mobile phone keypad, it is not surprising, therefore, that mobile text entry research includes numerous efforts to develop new or improved text entry techniques for mobile phones or other anticipated products supporting similar services.

NordiCHI 10/02 Århus, Denmark

Keyboard Configurations for Mobile Text Entry

Among the available configurations for keyed mobile text entry are devices with 5 keys, 8-12 keys, or 26+ keys. Five-key text entry, although not common, is supported on some two-way pagers, such as the *AccessLink II* by Glenayre Electronics (Charlotte, NC). Fours keys move a cursor about a two-dimensional on-screen keyboard while a fifth key selects a character, delivering it to the message buffer.

The traditional 12-key phone keypad – with A-Z encoded on eight keys – is widely used for text entry, as already noted. The most common input technique is *Multitap*, but linguistically enhanced techniques also exist, such as *T9* (Tegic Communications, Seattle, WA) or *LetterWise* (Eatoni Ergonomics, New York, NY) [6, 9, 12, 15].

As well, some devices bear a complete but miniature Qwerty keyboard, such as the *Blackberry* by Research In Motion (Waterloo, Canada), the *EL-6810B* organizer by Sharp Electronics (Mahwah, NJ), or the *Communicator* by Nokia (Helsinki, Finland).

In this article, we explore a potential input technique that requires just three keys: Left and Right arrow keys and a Select key. Figure 1 positions this technique in a *number-of-keys* continuum with the techniques just described.

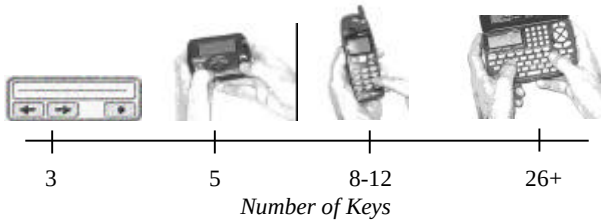

Figure 1. Keyed mobile text entry by number of keys

The form factor for the three-key concept in Figure 1 is just an example, and is by no means suggested as the preferred embodiment. A variety of other configurations are possible, such as embedding the keys in clothing on the wrist or forearm, or using finger-activated contact switches. The latter is one possible application of Lehikoinen and Röykkee's *N-Fingers*, a general purpose

input device for navigation and control [7]. With *N-Fingers*, contact between the thumb and the ring, middle, and index fingers can be mapped to the Left, Right, and Select keys for three-key text input. However, our goal here is to explore implementation and interaction issues for three-key text entry, without linking these to a specific form factor.

Text Entry Using Three Keys

The method described here has been called the *date stamp method* [1, 8]. It is so named because of similarity to a teller's date stamp, where characters are found by rotating a wheel containing the entire character set. As a text entry method, we assume Left and Right arrow keys maneuver a cursor over a linear sequence of letters and a Select key enters a letter. Arcade game players often use this technique to add one's name to a list of high scorers.

There are numerous ways to implement three-key text entry using Left and Right arrow keys and a Select key. Six possibilities are examined here (see Table 1).

Table 1
Six Methods of Three-Key Text Entry

Method	Character Arrangement [a]	Cursor Mode	*KSPC*
#1	_abcdefghijklmnopqrstufwxyz	Persistent	10.66
#2	_abcdefghijklmnopqrstufwxyz	Snap-to-home	10.62
#3	abcdefghijklm_nopqrstuvwxyz	Persistent	9.18
#4	abcdefghijklm_nopqrstuvwxyz	Snap-to-home	6.45
#5	_…… (FOCL *Level 1*) [b]	Snap-to-home	5.05
#6	_…… (FOCL *Level 2*) [b]	Snap-to-home	4.23

[a] SPACE is represented as '_'

[b] FOCL = fluctuating optimal character layout (see text)

A good start is just to arrange letters alphabetically with a SPACE at the left:

```
_abcdefghijklmnopqrstuvwxyz
```

Interaction proceeds by moving a cursor with the Left and Right arrow keys and entering characters with the Select key. We call this Method #1. The cursor mode for this method is *persistent* (see Table 1). This means after a character is selected, the cursor persists at the selected position. Movement proceeds from this position to the next character.

The column *KSPC*, for *keystrokes per character*, in Table 1 is particularly important in characterising and comparing text entry methods (see [8] for a detailed discussion). *KSPC* is the number of keystrokes, on average, to generate each character of text in a given language using a given text entry technique. With the Left, Right, and Select keys operating as just described, the number of keystrokes required to enter a character depends only on the preceding character; thus, we can compute *KSPC* for Method #1 using a digram-frequency table for a given language.

Our analyses here are based on the British National Corpus [2]. We work primarily with two forms of the corpus, a word-frequency list (64,566 words, frequencies totaling 90,563,946) and a digram-frequency list (27 ? 27 = 729 digrams, frequencies totaling 505,863,847). Below are the five most-frequent entries in the digram-frequency list, appended with keystrokes for Method #1:

```
e_ 18403847 LLLLLS
_t 14939007 RRRRRRRRRRRRRRRRRRRS
th 12254702 LLLLLLLLLLLS
he 11042724 LLLS
s_ 10860471 LLLLLLLLLLLLLLLLLLS
```

So, entering SPACE after *e* requires six keystrokes (LLLLLS), a very frequent act in English. With a full keystroke-appended digram table, *KSPC* is computed by summing the weighted keystroke counts. For Method #1, we obtain

$$KSPC = 10.66 \quad (1)$$

In Method #1, the cursor is persistent: It maintains its position after each character entered. Since SPACE occurs with the greatest frequency in text entry tasks, it is worth considering a *snap-to-home* mode, whereby the cursor jumps to the SPACE character after each entry. Thus, inputting a SPACE requires just one keystroke, regardless of the preceding character. We call this Method #2. The improvement is only slight, however:

$$KSPC = 10.62 \quad (2)$$

In theory, Method #2 requires just two keys because all cursor key motion is *to the right*. However, in practice, a Left key is still needed to correct for the occasional overshoot. This is examined in more detail later.

Another possibility is to position the SPACE character in the middle of the alphabet:

```
abcdefghijklm_nopqrstuvwxyz
```

Thus, SPACE is well-situated for English text entry. This letter arrangement combined with a persistent cursor bears further improvement (Method #3):

$$KSPC = 9.18 \quad (3)$$

However, a good leap forward is produced by combining a central SPACE character with a snap-to-home cursor (Method #4):

$$KSPC = 6.45 \quad (4)$$

English text is produced with about 40% fewer keystrokes per character using Method #4 than using Method #1.

Typamatic Cursor Movement

Despite the improvement with Method #4, entering text with 6+ keystrokes per character seems onerous. However, lurking within the apparently high overhead of cursor

movement is an opportunity for accelerated input. Consider, for example, the sequence *s*-SPACE shown earlier as 19 presses of Left followed by a single press of Select. Are 20 keystrokes really required? Perhaps not.

In practice, keyboards often include a *typamatic*, or *auto-repeat*, feature. Pressing and holding a key longer a certain delay threshold initiates a continuous fixed-rate stream of *virtual key presses*. There is clearly an opportunity to exploit this feature with the three-key text entry methods described here. In other words, the user may enter *s*-SPACE with Method #1 by pressing the Left key, holding for an auto-repeat interval as the cursor moves, then releasing the Left key and pressing the Select key. We will examine this in more detail later.

Linguistic Enhancement

Another possibility to reduce *KSPC* is to add linguistic knowledge to dynamically rearrange letters after each entry. The goal is to minimize the cursor-key distance to the next character. Bellman and MacKenzie [1] called this technique *FOCL*, for *fluctuating optimal character layout*. Their study focused on pager-style five-key text entry using a Select key, Up, Down, Left, and Right arrow keys, and an on-screen keyboard with letters arranged in three rows. Our focus here is on a much simpler interaction with only three keys and with letters arranged in a single row; thus the opportunity for exploiting typamatic input is greater.

Two levels of FOCL-style interaction are considered. Both position the SPACE character on the left and assume a snap-to-home cursor mode. With FOCL *Level 1*, the letters are rearranged after each entry considering their likelihood of following the character just entered. The most-likely next letter is adjacent to the SPACE, and so on. FOCL *Level 2* is the same except the order is determined by the two preceding characters. The idea is simply to further reduce the cursor-key distance to the next letter by adding more linguistic knowledge to the system.

Building a FOCL table is straight-forward. Table 2 shows the five most-common sequences for each level.

Table 2
FOCL *Level 1* and *Level 2* Examples

Preceding Character(s)	Next Character (ordered by probability)
	***** *FOCL Level 1* *****
_	taisowhcbpfmdrlengyuvkjqzx
e	rndsalectmvxyipfwgoqhkbujz
t	hoiearsuytlwcmnfbpgdzjkvqx
a	ntrlscdiybvmgpkufwhxjzeoqa
o	nrufmtwlospvdcikbgayhexjzq
	***** *FOCL Level 2* *****
e_	taisowhcbpfmdrlengyuvkjqzx
_t	horeaiwuysbcdfgjklmnpqtvxz
th	eaiorsuydlwmfncphqbtgkjvxz
he	rynimasdlteocfwpvqgxubkhjz
s_	taisowhcbpfmdrlengyuvkjqzx

Note that the SPACE character is the home position and is always entered with one keystroke. Also, our computations are based on the word-frequency reduction of a corpus, therefore, word transitions are not considered. Thus, FOCL *Level 2* degrades to FOCL *Level 1* for *letter*-SPACE digrams. This occurs for 27 of the 729 digrams, and is seen in the e_ and s_ entries in Table 2.

Given keystroke-appended digram tables for the two levels of FOCL just described, the *KSPC* characteristic is easily computed. The result for FOCL *Level 1* is

$$KSPC = 5.05 \quad (5)$$

and for FOCL *Level 2*,

$$KSPC = 4.23 \quad (6)$$

These appear in Table 1 as Method #5 (FOCL *Level 1*) and Method #6 (FOCL *Level 2*).

English text is produced with about 60% fewer keystrokes per character using Method #6 than using Method #1. However, it is naïve to suggest that a corresponding increase in text entry throughput will occur. Adding linguistic knowledge to the system changes the interaction considerably as numerous new issues surface.

There is clearly an increased attention demand with FOCL-style interaction, since users must react to a new letter arrangement after each entry. Is the cost of the added attention demand offset by the significant reduction in keystrokes? This remains to be seen.

Among other interaction issues with FOCL-style input is *chunking*. Despite the new arrangement of letters following each entry, users may acquire motor memory for frequent letter patterns, such as *the*, *and*, *in*, *to*, *ing*, *tion*, *for*, *is*, and so on. These will typically require just one or two presses of the Right-arrow key for each letter. There is the potential for users to develop the facility to enter such patterns without attending to the new letter arrangement. That is, they may proceed expeditiously using motor memory. This effect is only likely to surface after considerable practice, however.

Finally, it is important to remember that users attention is focused only on the display for three-key text entry. This is not the case for typical text entry on mobile phones, where users attend both to the keys (*Which key contains the desired letter?*) and to the display (*Was the correct letter generated?*). This also mitigates the impact of the attention demand with FOCL-style interaction.

In summary, we have described six possible implementations for three-key text entry on mobile systems. We have also discussed several important interaction issues that may impact performance, including the number of keystrokes required for each character entered, the possibility of using typamatic keying, and the increased attention demand in adding linguistic knowledge

to the system. Our next step is to conduct an empirical evaluation to measure text entry performance and other aspects of the interaction, and to solicit feedback from potential users of these techniques.

We implemented each of the six techniques described above and tested them informally with pilot subjects. On balance, we considered Method #2 and Method #6 the most promising and choose these for formal evaluation. They are similar in that both techniques position SPACE on the left with a snap-to-home cursor, different in that Method #6 works with FOCL-style interaction. The methodology and results are described in the following sections.

METHODOLOGY

Participants

Ten paid volunteer participants (8 male, 2 female) were recruited from the local university campus. Participants ranged in age from 20 years to 49 years (*mean* = 30.1, *sd* = 8.5). All were daily users of computers, reporting 3 to 12.5 hours of usage per day (*mean* = 7.9, *sd* = 3.3). Self-assessed typing speeds ranged from 35 to 105 words per minute (*mean* = 62.7, *sd* = 22.5). Six users described themselves are "regular users of computer games".

Apparatus

The experiment was conducted in a quiet office using a 400 MHz Pentium-class desktop computer running under Microsoft *Windows 98*. The system included a 19" colour monitor and a standard mouse and keyboard. The default keyboard mapping for the input keys was Left-arrow = Z, Right-arrow = X, Select = Enter. During entry, the middle finger and index finger on the left hand pressed the Left and Right arrow keys, respectively, while the index finger on the right hand pressed the Select key. These mappings could be changed, if requested by participants. All other keystrokes were ignored.

The typamatic behaviour of the keyboard was configured via the system's control panel. We used the shortest auto-repeat delay and the shortest repeat interval (i.e., fastest repeat rate). These were considered reasonable based on pilot tests with the experimental software, and previous research citing a user preference for the fastest available cursor speed [4]. A simple experiment was conducted to measure the actual repeat delay and repeat interval on the experimental system. They were measured as follows:

$$t_{\text{DELAY}} = 176 \text{ ms} \quad (7)$$

and

$$t_{\text{TYPAMATIC\_REPEAT}} = 32.1 \text{ ms} \quad (8)$$

The repeat interval of 32.1 ms corresponds to an auto-repeat rate of 31.2 characters per second. Auto repeat begins following the 176 ms delay interval.

The experimental software was an in-house Java application for text entry evaluation. Upon launch, the program reads a file containing a series of text phrases. During execution, phrases are selected randomly and presented to the participant for input.

The phrase set contained 500 phrases ranging from 16 to 43 characters (*mean* = 28.6). There were 2712 total words, including 1163 unique words. Words ranged from 1 to 13 characters (*mean* = 4.46). The correlation between the letter frequencies in the phrase set and those in our reference corpus was $r = .9541$.

Screen snaps of the software in use are shown in Figure 2a for Method #2 and Figure 2b for Method #6. The top line shows the presented text phrase, while the middle line shows the progress of input. The bottom line shows the letter sequence according to the input method. The cursor position appeared as a blue box around a white character. Errors could not be corrected.

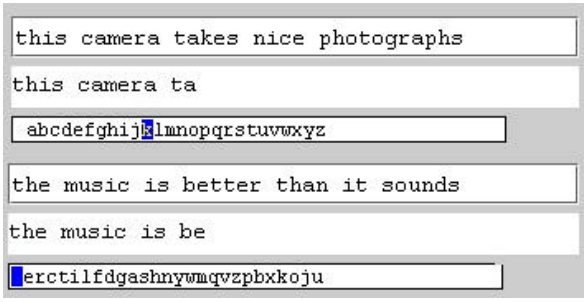

Figure 2. Screen snaps of the text-entry evaluation software (a) Method #2 (b) Method #6.

Procedure

Participants completed a pre-test questionnaire soliciting demographic and computer usage information (results cited above) and a post-test questionnaire on their subjective impressions of the methods (discussed later).

Prior to collecting data, the experimenter briefly explained the task and demonstrated the software. The instructions were to enter a series of text phrases "as quickly and accurately as possible" using the specified input technique. Participants were instructed to ignore mistakes and to continue with the rest of a phrase in the event of an error.

The operation of the Left, Right, and Select keys was explained, as was the general idea of linguistic enhancement for Method #6.

Participants were then allowed to enter a few warm-up phrases and ask questions about the procedure. They were also given an opportunity to choose a different mapping for the Left, Right, and Select keys. All but one participant felt comfortable with the default mappings. For the other participant, a slight change was introduced through the software's configuration file.

Data collection began with the first keystroke for each phrase and ended with the last keystroke. Participants

were allowed to rest at their discretion between phrases. Each participant was scheduled for a one-hour appointment, resulting in about 25 minutes of data collection for each entry method.

The experiment was a within-subjects design with two conditions: Method #2 vs. Method #6. The order of conditions was counterbalanced. Half the participants entered text first using Method #2, then using Method #6. For the other half, the order was reversed.

The software recorded a timestamp and key code for each keystroke, saving these in files for follow-up analyses.

RESULTS AND DISCUSSION

In all, participants entered 1354 phrases of text, including 673 phrases for Method #2, and 681 phrases for Method #6. Our analyses begin with measures for speed and accuracy.

Speed and Accuracy

The overall results for text entry speed are shown in Figure 3. At 9.61 wpm, the entry rate for Method #6 was 5.6% faster than the 9.10 wpm rate observed for Method #2. The difference was not statistically significant, however ($F_{1,9} = 2.843$, $p > .05$).

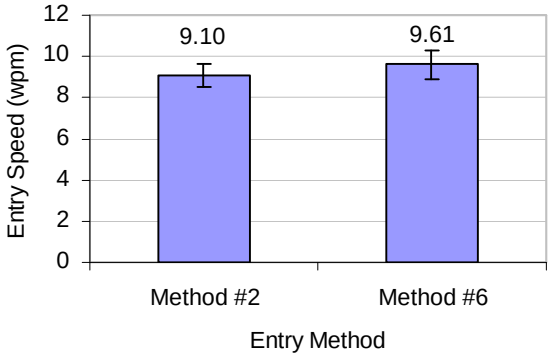

Figure 3. Entry speed (wpm) by method (Note: Error bars span one standard deviation)

Overall, the results for entry speed seem low and somewhat disappointing. Neither method exceeded 10 wpm and FOCL-style linguistic enhancement failed to yield a significantly higher text entry throughput. Importantly, the techniques tested were unfamiliar to all participants and the measurements reported are the mean over just 25 minutes of practice. By comparison, Bellman and MacKenzie [1] reported text entry rates of 10-11 wpm for two pager-style five-key techniques, but these were achieved on the tenth session of testing. Rates were only 5-6 wpm on the first session. Similarly, MacKenzie et al. [9] tested two text entry techniques for mobile phones and measured rates of 15-20 wpm after 20 sessions of practice. However, on the first session, the rates were just over 7 wpm. James and Reischel [6] also tested two text entry techniques for mobile phones. For the novice group (tested in one session only), they reported rates of 8-9 wpm.

Other mobile text entry techniques include handwriting with automatic recognition and stylus tapping on a graphical qwerty keyboard. First-session rates are typically in the range 15-28 wpm [10, 13], but participants import substantial prior skill, due to life-long experience with handwriting and qwerty-style keyboards. The situation is quite different when users confront a graphical keyboard with an unfamiliar layout, however. MacKenzie and Zhang [14] measured stylus tapping rates on a graphical keyboard with a randomized letter arrangement. Participants' text entry rates were 5-6 wpm.

Considering the above examples in the research literature and that the techniques tested here are at the low end of the number-of-keys continuum for keyed text entry (see Figure 1), the results in Figure 3 seem quite reasonable.

Error rates were computed using the minimum-string-distance method [11, 16]. The results are shown in Figure 4. Both methods demonstrated an error rate a little over 2%. As evident by the wide error bars, there was substantial variation in the error rates across subjects. For Method #2, subjects' error rates varied from 0.78% to 4.21% (*sd* = 0.63), and for Method #6 from 0.57% to 4.58% (*sd* = 0.63). Not surprisingly, the difference in error rates was not statistically significant ($F_{1,9} = 0.241$, ns).

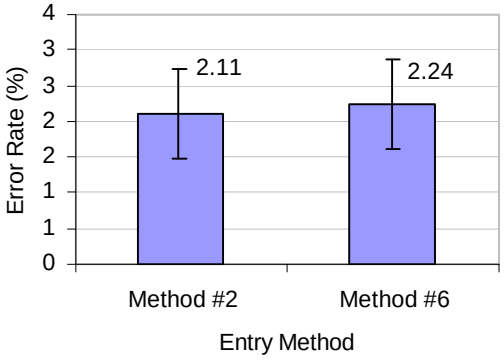

Figure 4. Error rate (%) by method (Note: Error bars span one standard deviation)

Despite the lack of significant differences between the two entry methods on the dependent measures for speed and accuracy, there are substantial differences between the two methods. These are borne out in more detailed analyses on other dependent measures.

Keystrokes Per Character (KSPC)

Earlier we reported the *KSPC* values for each of the proposed methods for three-key text entry. The values in Table 1 are *computed*, however, and may differ from the *observed* number of keystrokes per character. Figure 5 shows both the computed and observed values. The observed values are presented in two forms: *including* and

excluding typamatic keystrokes. Including typamatic keystrokes means all keystrokes – including virtual keypresses during auto-repeat – are counted. Excluding typamatic keypresses means the virtual keypresses – those occurring automatically after the auto-repeat delay – are not counted.

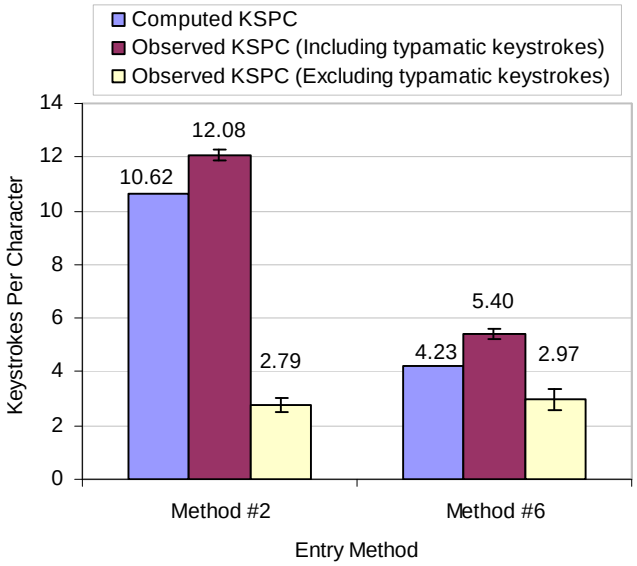

Figure 5. Keystrokes per character (*KSPC*), computed and observed, by entry method

For both entry methods, the observed *KSPC* (including typamatic keypresses) was higher than the computed *KSPC*. There are at least three reasons for the differences. The first is due to minor linguistic differences between the particular set of phrases entered and the language model. The effect of this difference on *KSPC* might be plus or minus, depending on the statistical structure of each phrase. The effect should be minor, however, due to the high correlation between the letter frequencies in the phrase set and those in the reference corpus.

The second is due to errors. An undershoot error tends to decrease *KSPC*, while an overshoot error increases *KSPC*. Assuming undershoot errors and overshoot errors occur with approximately the same frequency, the expected effect on *KSPC* is neutral.

The third is due to non-optimal entry. If the participant overshoots the intended character, then backs up and correctly enters the character, extra keystrokes are incurred. The effect is always to increase *KSPC*. Since the first two effects are small or neutral, this effect is likely the dominant reason *KSPC*-observed (middle bar in Figure 5) differs from, and is higher than, *KSPC*-computed.

The difference shows *inefficiency* in the interaction. Comparing the left two bars in Figure 5 for each method, we see that users entered more keystrokes than necessary: 13.7% more for Method #2 and 27.7% more for Method #6. That the percent difference is higher for Method #6 means participants did not "cash in" on the benefits of FOCL as much as they could have. The most likely reason is that participants tended to overshoot and adjust more often with Method #6 than with Method #2.

The observed *KSPC* figures, excluding typamatic keystrokes, are also shown in Figure 5. Evidently, exploiting typamatic input has considerable impact on the work required to enter text using the three-key methods under investigation. Comparing the first and third bars in Figure 5 for each method, typamatic input substantially reduced the number of physical keypresses for each method: by 73.7% for Method #2 and by 29.8% for Method #6. The reduction is not as pronounced with Method #6, perhaps due to a floor effect; that is, *KSPC*-computed is low to begin with. In fact, the *KSPC*-observed value, excluding typamatic keystrokes, was slightly higher for Method #6 than for Method #2. This suggests participants were "doing more work" with Method #6.

Typamatic Events

Even though Method #6 requires less than half the keystrokes per character than Method #2, a corresponding increase in text entry throughput did not materialize. There are at least two explanations. The first is the added attention demand in finding the correct letter in a fluctuating layout, as noted earlier. The second is that the average distance of typamatic movement is greater with Method #2; thus, typamatic movement tends to benefit Method #2 more than Method #6. The latter effect is shown in Figure 6.

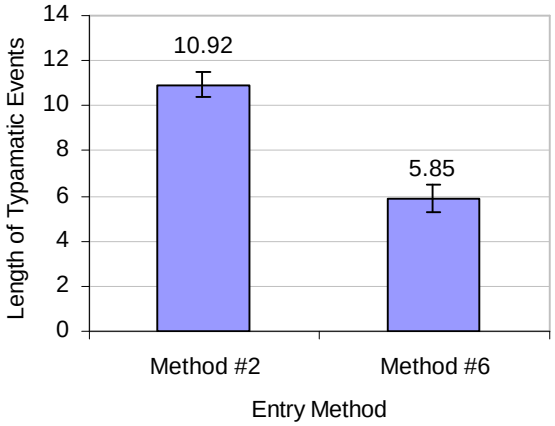

Figure 6. Length of typamatic events by entry method (Note: Error bars span one standard deviation)

A typamatic event is defined as any key sequence that begins with a physical keypress and extends to one or more virtual keypresses through auto-repeat. The length of the event is the sum of the physical and virtual keypresses. With Method #2, the average length of a typamatic event was 10.92 keystrokes. This is almost twice the same figure for Method #6 (5.85 keystrokes). The lower value for Method #6 is likely due to FOCL-style interaction. That

is, the inherently lower *KSPC* with FOCL-style interaction also reduces the opportunity for typamatic keying.

If typamatic keying was not available, greater differences in throughput might occur, with Method #6 faster than Method #2 simply due the reduced keystrokes to enter each character. However, this is a moot point since the difference would yield an even slower throughput for Method #2, rather than a higher throughput for Method #6.

Keystroke Categories

One final analysis is presented to demonstrate the significant impact of typamatic keying on the entry methods under investigation. Figure 7 shows a categorization of keystrokes by method.

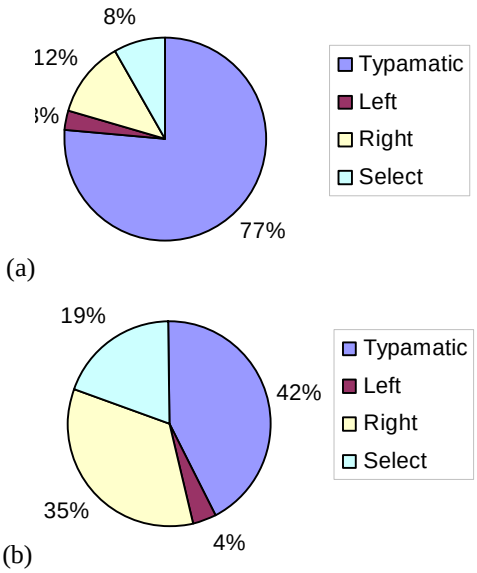

Figure 7. Keystroke categories by method
(a) Method #2 (b) Method #6

Left, Right, and Select keystrokes are physical keypresses, while Typamatic keystrokes are virtual keypresses. For Method #2, about 77% of all keypresses are of the latter type. The same figure for Method #6 is considerably less – just 42%, for reasons noted earlier. Right-arrow keystrokes are correspondingly higher for Method #6, and this is likely related to the preceding comment. In other words, the inherently lower *KSPC* with Method #6 is coincident with a higher tendency to use short bursts of Right-arrow keystrokes instead of typamatic keying.

Motor-Sensory Issues

The last point above is worthy of separate discussion, as it carries an interesting mix of motor-sensory issues. These are briefly examined here. Consider the following two keying situations: (R = Right-arrow key, S = Select key)

RRRRRRRRRRRRRRRRRRRS

RRRRS

The first example clearly presents an opportunity for typamatic keying, but which keying strategy is best for the second example? A keystroke-level analysis can help answer this question. We can ignore the final Select keypress because it is the same for both strategies. The time for n physical presses on the Right-arrow key is estimated as

$$t_{PHYSICAL} = n ? t_{REPEAT} \quad (9)$$

where t_{REPEAT} is the time for each press where a key is pressed repeatedly.

The formula is slightly different for typamatic keying, because the first keypress is physical and the rest are virtual and follow a delay interval. The estimated time for n cursor key presses using typamatic keying is

$$t_{TYPAMATIC} = t_{DELAY} + (n - 1) ? t_{TYPAMATIC\_REPEAT} \quad (10)$$

The experimental settings for t_{DELAY} and $t_{TYPAMATIC\text{-}REPEAT}$ were given earlier in equations 7 and 8. A reasonable value for t_{REPEAT} is 140 ms [3, p. 60]. The crossover point – the number of keystrokes above which typamatic keying is faster – is found simply by equating the right-side of equations 9 and 10 and solving for n. The result is $n = 1.33$. This figure is remarkably low. The implication is that for two or more presses of the same key, the result is more quickly achieved using a typamatic keying strategy.

The crossover point varies with t_{REPEAT}, t_{DELAY}, and $t_{TYPAMATIC\text{-}REPEAT}$. While t_{REPEAT} is constrained by the human motor system, t_{DELAY} and $t_{TYPAMATIC\text{-}REPEAT}$ are system-dependent, and, therefore, tunable. For example, doubling then trebling these parameters increases the crossover point to $n = 3.80$ and $n = 9.88$, respectively.

However, there are other, arguably more important, issues, such as the user's ability to sense and respond to the advancing motion of the cursor. To effectively use typamatic keying the user must monitor the movement of the cursor and release the arrow key within a window of time equal to $t_{TYPAMATIC\text{-}REPEAT}$. This window was extremely narrow in our experiment: just 32.1 ms! (See equation 8.) Most likely, participants used typamatic keying as a strategy to "get to the vicinity of" the desired character, with a final adjustment if necessary. The presence of Left-arrow keypresses (see Figure 7) is an indication of the occasional need to correct for an overshoot in typamatic keying.

An interesting research topic, therefore, is examining the interaction between users' ability to employ typamatic keying and the parameters that affect performance, such as t_{DELAY}, $t_{TYPAMATIC\text{-}REPEAT}$, and keying distance.

Participant Questionnaire

The post-test questionnaire solicited general comments and a response to three statements. The statements and response means are shown in Figure 8.

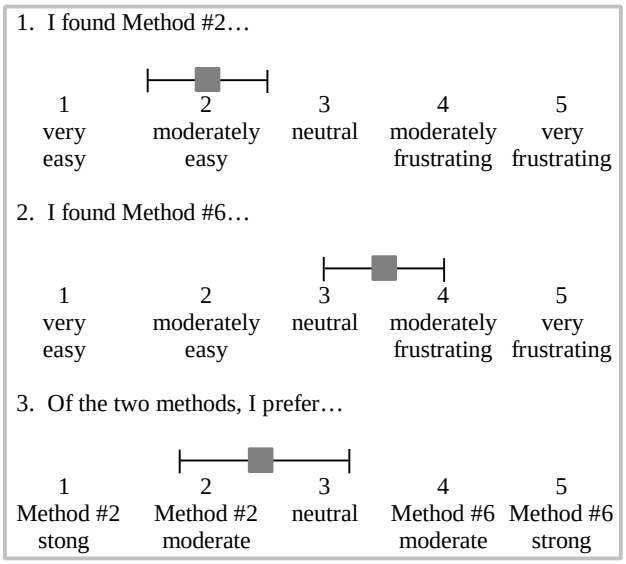

Figure 8. Post-test questionnaire results (Note: Error bars show +/-1 standard deviation)

Overall, Method #2 was preferred. Among the general comments, some participants felt Method #6 required more concentration. The added keying for Method #2 received a few comments. One participant noted the high cost of mistakes with Method #6, referring presumably to the unpredictable letter arrangement following a mistake. Another participant expressed extreme frustration with Method #6, feeling that the computer was trying to "trip you up" by shuffling the letters after each entry.

CONCLUSION

Six techniques for three-key text entry on mobile systems were presented. The techniques use Left- and Right-arrow keys to maneuver a cursor over a linear sequence of characters, and a Select key to enter characters. The keystrokes per character (*KSPC*) for the methods varies from 10.66 to 4.23. Method #2 and Method #6 were formally evaluated in an experiment with ten participants who entered text for about 25 minutes with each technique. Text entry throughputs were 9.10 wpm and 9.61 wpm for Method #2 and Method #6, respectively. The opportunities for typamatic keying are particularly interesting for the techniques described here because cursor distances are substantial and characters are arranged in a single row.

Adding linguistic knowledge with Method #6 did not increase text entry throughput. This is attributed to the added attention demand and to increased typamatic keying for Method #2. At *KSPC* = 10.62, Method #2 requires more cursor movement than Method #6, and, therefore, benefits more from typamatic keying. Further research is warranted to establish the optimal delay time and repeat interval for text entry systems using typamatic keying.

Acknowledgement

Thanks to Chris Klochek for contributing to the software development and for testing the experimental protocol.

References

1. Bellman, T., and MacKenzie, I. S. A probabilistic character layout strategy for mobile text entry, *Proc GI '98*, 1998, 168-176.
2. BNC: ftp://ftp.itri.bton.ac.uk/bnc/ (file repository for British National Corpus), (2002).
3. Card, S. K., Moran, T. P., and Newell, A. *The psychology of human-computer interaction*, Hillsdale, NJ: Lawrence Erlbaum, 1983.
4. Gould, J. D., Lewis, C., and Barnes, V. Cursor movement during text editing, *ACM Trans Office Information Systems 3* (1985), 22-34.
5. http://gsmworld.com/technology/sms.html (includes various SMS statistics), (2002).
6. James, C. L., and Reischel, K. M. Text input for mobile devices: Comparing model predictions to actual performance, *Proc CHI 2001*, 2001, 365-371.
7. Lehikoinen, J., and Röykkee, M. N-Fingers: A finger-based interaction technique for wearable computers, *Interacting with Computers 13* (2001), 601-625.
8. MacKenzie, I. S. KSPC (keystrokes per character) as a characteristic of text entry techniques, *Proc HCI Mobile 2002*, 2002, (to appear).
9. MacKenzie, I. S., Kober, H., Smith, D., Jones, T., and Skepner, E. LetterWise: Prefix-based disambiguation for mobile text input, *Proc UIST 2001*, 2001, 111-120.
10. MacKenzie, I. S., Nonnecke, R. B., Riddersma, S., McQueen, C., and Meltz, M. Alphanumeric entry on pen-based computers, *International Journal of Human-Computer Studies 41* (1994), 775-792.
11. MacKenzie, I. S., and Soukoreff, R. W. Character-level Error Analyses for Evaluating Text Entry Methods, 2002, [submitted for publication].
12. MacKenzie, I. S., and Soukoreff, R. W. Text entry for mobile computing: Models and methods, theory and practice, *Human-Computer Interaction* (2002), [to appear].
13. MacKenzie, I. S., and Zhang, S. X. The design and evaluation of a high-performance soft keyboard, *Proc CHI '99*, 1999, 25-31.
14. MacKenzie, I. S., and Zhang, S. X. An empirical investigation of the novice experience with soft keyboards, *Behaviour & Information Technology 20* (2001), 411-418.
15. Silfverberg, M., MacKenzie, I. S., and Korhonen, P. Predicting text entry speed on mobile phones, *Proc CHI 2000*, 2000, 9-16.
16. Soukoreff, R. W., and MacKenzie, I. S. Measuring errors in text entry tasks: An application of the Levenshtein string distance statistic, *Ext Abstracts CHI 2001*, 2001, 319-320.