A System for Fast, Full-Text Entry for Small Electronic Devices

Saied B. Nesbat, Ph. D.

ExIdeas, Inc
Belmont, CA 94002 USA
saied@exideas.com

ABSTRACT

A novel text entry system designed based on the ubiquitous 12-button telephone keypad and its adaptation for a soft keypad are presented. This system can be used to enter full text (letters + numbers + special characters) on devices where the number of keys or the keyboard area is limited. Letter-frequency data is used for assigning letters to the positions of a 3x3 matrix on keys, enhancing the entry of the most frequent letters performed by a double-click. Less frequent letters and characters are entered based on a 3x3 adjacency matrix using an unambiguous, two-keystroke scheme. The same technique is applied to a virtual or soft keyboard layout so letters and characters are entered with taps or slides on an 11-button keypad. Based on the application of Fitts' law, this system is determined to be 67% faster than the QWERTY soft keyboard and 31% faster than the multi-tap text entry system commonly used on cell phones today. The system presented in this paper is implemented and runs on Palm OS PDAs, replacing the built-in QWERTY keyboard and Graffiti recognition systems of these PDAs.

Categories and Subject Descriptors
H.5.2 [**User Interfaces**]: Graphical user interfaces (GUI), Interaction styles.

General Terms: Human Factors.

Keywords:Text entry, pen-based, mobile systems, mobile phones, keypad input, Fitts' law, soft keyboard, stylus input..

1. INTRODUCTION

As the mobile electronic devices converge and their uses become pervasive, their text entry technologies continue to diverge. The most widely used mobile device around the world today is the cellular-phone. Most of more than 500 million cell phones in use today provide the multi-tap system for text entry, a technology that is neither efficient nor user-friendly. PDAs, still growing in popularity, use either a miniaturized QWRTY keyboard, or on-screen letter recognition systems (e.g., Jot, Graffiti). Some communication and email devices (e.g., BlackBerry, HipTop) use a miniaturized full QWERTY keyboard to be operated with two thumbs. Various point-of-sale devices used in the industry employ yet another keyboard, where the letters are arranged in alphabetical order. The emerging interactive TV systems either rely on scroll-and-pick systems invoked through the TV remote controllers, or require a full QWERTY keyboard for text entry. Car navigation systems generally depend on their own proprietary text entry technology. For these devices, portability prohibits the use of a ten-finger QWERTY keyboard and reduces the area or the number of keys devoted for text entry. While many of these devices will converge, it is not clear which of their text entry technologies will prevail.

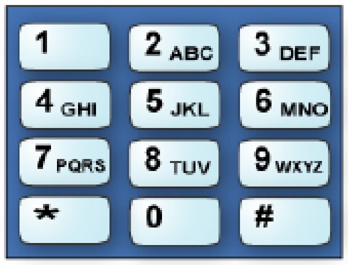


Figure 1. A keypad of a common mobile phone.

Cell phones, vastly outnumbering the other mobile devices, have an advantage in defining a de-facto text entry system for all small mobile devices. However multi-tap, the prevailing text entry technology used on the cell phones today, suffers from the legacy assignment on the keys of the telephone [Figure 1]. A relic from when human operators manually switched regional phones. Not properly designed for text entry, this heirloom "standard" leaves little room for creating an efficient text input system to be used by other mobile devices. It is also unlikely that cell phone's form-factor will change to conform to the text entry systems of other small mobile devices. The lack of an efficient text entry system applicable to both cell phones and other mobile devices impedes the convergence of all mobile devices. This paper introduces a system of efficient text entry which is applicable to the current form factor of cell phones, yet is also adaptable for use on any small mobile device.

1.1 State of the Art

Multi-tap [14] is the most widely used method of text entry on cell phones. Using this method, to enter each letter the user presses, clicks, or taps a key once or several times until the desired letter appears on the display. The number of times a key must be tapped depends on the position of the letter on its key. E.g., to enter S, key 7 needs to be tapped four times. If two consecutive letters share the same key, then after entering the first letter the user either needs to pause for about 1-2 seconds, or press another key (e.g., # key) to indicate that the next tap is for the next letter.

Besides multi-tap, some mobile phones use a so called two-key method to enter text. Using this method, first the key where the letter indicated is tapped, then one of the keys 1-4 corresponding to the position of that letter on that key is Tapped. E.G., to enter S first key 7, then key 4 is pressed.

Neither multi-tap nor two-key method is designed for efficient text entry; they both use the legacy letter assignment without attempting to reduce the number of key taps. Furthermore, using either of these methods to enter characters other than letters and numbers is an even bigger chore. For example, to enter '@' the user must switch to a different mode where all the Special characters are displayed and the desired character is tediously picked by moving the cursor and selecting it. Therefore, these methods, while barely adequate for entering names and a phone numbers, are inadequate when entering an email address, a URL, or any text with punctuation marks.

1.2 Predictive Methods

To alleviate the pains of multi-tap, several predictive and disambiguating methods have been introduced in recent years (T9 [3], iTap [12], ezText[1], Eatoni[9]). Using the same legacy lettering, most of these systems require the user to press or tap only one key per letter; the system, using a wordlist or compiled word statistics, attempts to disambiguate and figure out the true intention of the user. When more than one word map onto the same key sequence, these systems provide several words for the user to choose from. To succeed, the user is required to vigilantly watch the display, lest an unwanted word is entered by the system. This puts an unreasonable cognitive load on the user not only to spell and enter the words correctly.

The cognitive load imposed by these systems is heaviest when entering words conforming to neither the wordlist nor the word statistics. This happens when words of two or more languages are mixed (e.g., Hello amigo), if obscure abbreviations are used (e.g., your QPR affects our NSTH), or if slang or made-up words are entered (e.g., wusup, cooool maaan!). In these cases the disambiguation system often fails, forcing the user to revert to the slow multi-tap process.

1.3 Soft or Virtual keyboards

Another class of solutions is based on virtual or soft keyboards, where the size and positions of the keys can be arbitrarily defined to increase efficiency. With the popularity of Palm Pilots and the emergence of tablet and wearable computers this strategy has become of particular interest. Using virtual keyboards different keyboard configurations can be used to suit the task at hand or the language to be used. Mackenzie [11] and Zhai [16] have presented a quantitative analysis of various soft keyboards; they have presented a unified framework for measuring theoretical maximum speed of text entry based on Fitts-digraph model [16] for relative performance comparison of several designs. Later on in this paper this framework will be used to evaluate the soft key version of our design.

2. A NOVEL TEXT ENTRY SYSTEM

In this section we present our text entry system, called MessagEase™. This system is primarily designed for the cell-phones phone factor, using its common 12-button keypad. However this design is applicable to any devices sporting the 10- or 12-button keypad. Furthermore the principles behind this design are also utilized to create an efficient soft keyboard applicable to PDAs, tablet computers, watches, and other similar devices.

Our primary design objective was to use the same 12- button keypad and its numeric designators (1-9, 0) but not necessarily the same relic lettering assignment commonly found on phone's keypad. In fact, an important principle guiding our design was the notion that different letters occur at different rates in a body of text, Figure 2.

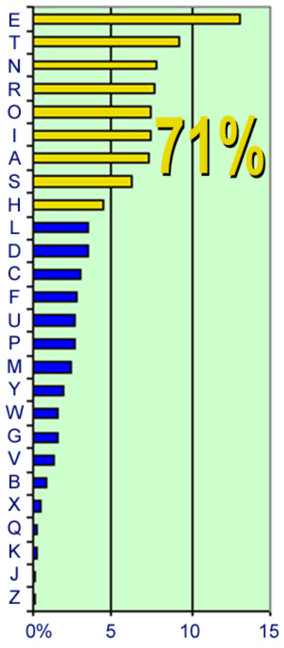


Figure 2. Letter-frequency of a body of English Text.

Mayzner [7] has published data indicating that, for example the letter E occurs about 13%, and the letter Z appears about 0.2% in a body of English text. (This information shows that using the current lettering assignment and multi tap system, in many instances more frequent letters, such as S require more taps – four taps – than infrequent letters, such as W – one tap). We used this data lettering assignment so more frequent letters would be easier to enter than the less frequent ones. Figure 2 depicts the letter frequency distribution of the 26 letters of alphabet based on the data published in [7].

Another design goal was to create a deterministic predictable entry system, without the need for any disambiguation: only the time independence of its key taps governing characters entered, avoiding cognitive load.

Obviously, given that there are 12 buttons on the keypad and 26 letters in the alphabet, at least some letters must share keys. But this sharing must be guided by some efficiency criteria, rather than just an alphabetical order. The first design decision was to assign the top 9 most frequent letters – E, T, A, O, N, I, S, and H – to the keys 1-9 of the keyboard (Figure 3.a). Given this assignment, each letter of this group is entered by a double clicking (tapping twice) the button corresponding to that letter. Since the letters in this group comprise 71% of the text [7], then to enter text most of the time one needs only to double click. (Optimization of these letter positions is presented later on in this paper.)

Next, 8 less frequent letters (V, L, X, M, F, W, Y, and K) are assigned to two-key sequences. Each of these letters are indicated on the side of one of the peripheral keys (keys 1, 2, 3, 4, 6, 7, 8, and 9) closest to the central key 5 (Figure 3.b). To enter any of these letters, first its key, then key 5 (the key that the position of that letter points to) is tapped. In other words, to enter any of these letters, first one of the peripheral keys then the central key (key 5) is tapped. For example, to enter the letter L, key 2, then key 5 is tapped.

Another group of 8 less frequent letters (U, P, B, J, D, G, C, and Q) is assigned to the peripheral positions of the center key 5, as depicted in Figure 3.c. The position of each of these letters points to one of the peripheral keys. In a similar fashion, a two-key sequence is used to enter each of these letters: first the central key 5 is pressed then the peripheral key pointed to by the position of that letter is pressed. For example, to enter the letter J, key 5 then key 9 is pressed.

The last remaining letter, Z is assigned to a two-key sequence 8-9. Therefore, the letter z is indicated on the side of key 8, pointing to key 9.

Out of the remaining three keys (*, 0, #), key 0 is used to enter the SPACE character, using only one key press. Key '*' is used as a possible toggle between *numeric* or *alphabet mode*, and key '#' is used for BACK SPACE character, using a single key press.

2.1 Measuring the performance

To measure the efficiency and performance of MessagEase's design, we used the methodology presented in MacKenzie, [8], Silfverberg [13], and Soukoref [14]. We used Fitts' law to model the time to move from one key to another to tap the latter, as described in [13]:

$$MT = a + b \log_2 (A/W+1) \qquad (1)$$

Since MessagEase has no timeout or time dependency, and every letter is entered with two taps, the time to enter each letter is the sum of the time it takes to tap the first key (MT_1) and the time it takes to tap the second (MT_2):

$$CT = MT_1 + MT_2 \qquad (2)$$

For the letters requiring double clicks, there is no movement for the second key tap:

$$CT_{DC} = 2a + b \log_2 (A/W+1) \qquad (3)$$

To compare our results with those published in [13], the parameters of the same keypad of Nokia 5100 series phone

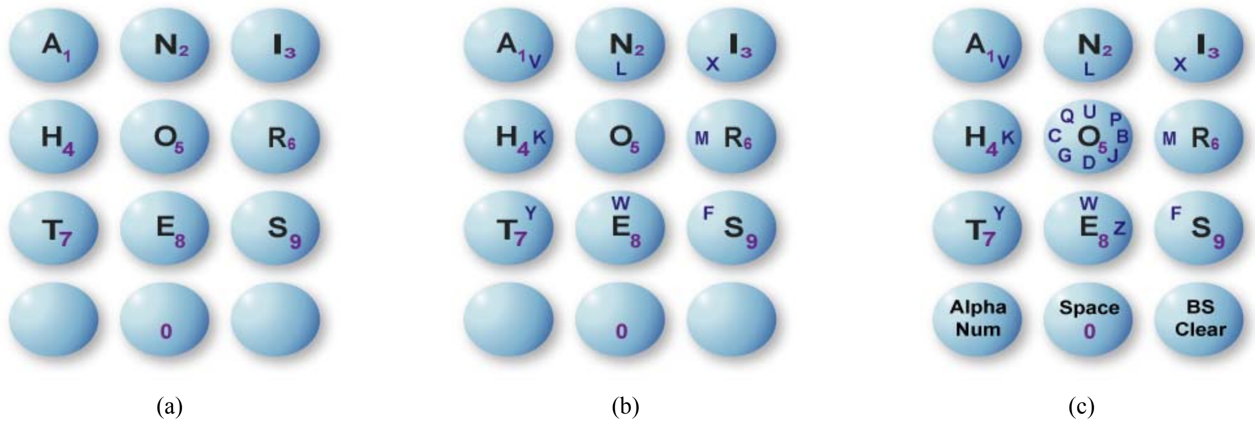


Figure 3. MessagEase's letter assignment.

were used (W = 6 mm; Column pitch = 13.25 mm; Row pitch = 8.7 mm). The same empirical results found in [13] were used to further validate our comparison: (a = 165 and b= 52 for index finger and a = 176 and b= 64 for thumb). As described in [13], the average character entry time is:

$$CT_{av} = \sum\sum(P_{ij} \times CT_{ij}) \qquad (4)$$

P_{ij} is derived from the same 27 × 27 digraph published in [7] and used in [13], [11], and [16]. From the above and assuming five characters per word, the theoretical upper limit of the text entry speed is calculated as:

$$WPM = (1/\, CT_{av}) \times (60/5) \qquad (5)$$

An Excel worksheet was developed for calculating the speed based on the above formula. our initial configurations resulted in 28.5 WPM when index finger factors were used and 25.9 WPM when thumb factors where used. These numbers compared favorably with multi-tap's 22.5 WPM, and 20.8 WPM, respectively [13].

With these encouraging results, and with favorable preliminary feedback, we set out to improve and optimize MessagEase™.

2.2 Simulating and Optimizing the Results

In order to optimize MessagEase™'s keypad letter assignment, we developed a computer program written is C and run on a 2.4 MHz Pentium IV PC. In its first version, given a lettering configuration, the program computed text entry speed by applying Fitts' law and using the same letter frequency di-grams[7]. Then the program was modified to generate assignments and to select the one with maximum speed. Rather than attempting to simulate all possible combinations of assignments— O(n!), or about 10^{28}, a computationally impossible task [16]— we set out to simulate all possible combinations of assignment within the fields of the most frequent and least frequent letters. First, keeping the assignment of least frequent letters constant, we simulated 9! assignment possibilities of the most frequent letters. Doing so improved the speed by about 3%, raising it to 29.4 WPM. Further, we simulated the less frequent letters in groups of 8, manually grouped which marginally improved the speed up to 29.53 WPM.

This speed insensitivity to the rearrangements of the less frequent letters was expected, as by definition these letters have relatively small probability factors P_{ij} (see equation 4), contributing relatively little to the overall speed. However, we took advantage of this relative speed insensitivity and employed further manual grouping of the less frequent characters to enhance the visual appeal and to facilitate keyboard pattern recall. We grouped all "curved", less frequent letters around the central letter "O" ("O" was placed in the center by the simulation results of step 1). Then we placed the rest of the letters — letters without curves — on the edges of the peripheral keys. The positions of letter Z, and the SPACE character remained unchanged. This left the overall speed essentially unchanged (from the high speed of 29.53 WPM down to 29.51 WPM), making the visual enhancements thus achieved worthwhile.

This speed of 29.5 WPM thus reached is 31% faster than the speed measured for multi-tap, as reported in [13].

2.3 Extension to Special Characters

Figure 3(b), shows how the central key 5 is used as a pointer for 8 peripheral keys, with 8 letters pointing to their corresponding second keys. Assuming rollover, *every* key from the set of keys 1-9 has 8 positions pointing to 8 other keys of the same set (i.e., each has top, top-right, right, right-bottom, bottom, bottom-left, left, and left-top positions). When a key position points to no physical key (e.g., there is no key to the right of key 3) rollover points to a key on the other side of the layout of the keys (i.e., the right side of key 3 points to key 1).

By allocating each of these positions to a symbol or special character, they can be entered using the same logic and a similar two-key sequence. With this simple extension, the matrix of the first 9 keys is capable of entering 81 characters, each only with a two-key sequence. 26 of these two-key sequences were used for the letters of the alphabet. two sequences (6-3, and 6-9) were used for cap-shift and lower-case; this allowed entering both uppercase and lowercase letters using 28 out of the possible 81 two-key sequences.

Some of the available two-key sequences were used to enter 36 special characters found on a standard keyboard, as shown in Figure 4. For example, key sequence 6-8 is used to enter "@" and, using rollover, key sequence 6-4 is used to enter ")". Applying the same logic, key sequence 3-7 is used to enter a carriage return (marked CR).

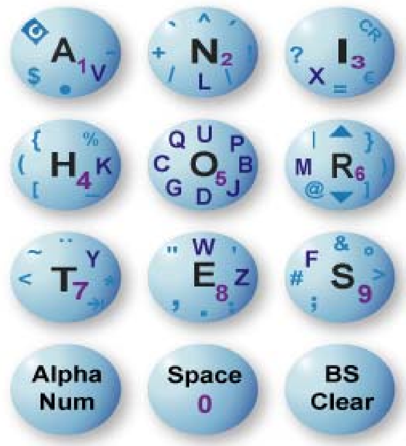


Figure 4. Letters and special characters assignment on MessagEase's keypad.

2.4 Accented and Other Special Characters

To accommodate even more characters, especially the many accented characters used in the European languages, a combining sequence is used. Key sequence 1-9 (marked by an inverted C**)** is used to combines the last two characters entered into a new one, if such a combination is applicable (i.e., defined in the code). For example, to enter 'â' first 'a' is entered (double click 1), then '^' is entered (key sequence 2-8), finally **combine** command is entered (key sequence 1-9). The last key sequence combines 'a^' to produce the desired character 'â'. Using the **combine** command, MessagEase is capable of entering many special characters not directly available on an ordinary QWERTY keyboard, including £, €, §, ©, µ, ½, ¿, Ö, ã, ç, ê, ÷, ø, and ÿ. With this extension, more than 6000 additional symbols may be entered, each with six key strokes.

3. SOFT KEYBOARD DESIGN

The same layout designed for the 12-button keypad is used for a soft-key implementation (Figure 5). This implementation has a distinct advantage over the hard-key version: the two-key sequence required to enter a letter or character is simplified to a single tap or drag (slide) on the screen. For the 9 most frequent letters only a single tap on its corresponding key area enters that letter. For a less frequent letter, a drag or slide, starting from its first key, *in the direction of* its second key enters that letter. In general, any character or symbol that required two-key taps is entered by its corresponding single drag on the soft key implementation. For example, touching the stylus anywhere on the top-middle square on MessagEase's soft keypad (Figure 5) and dragging it down (for about ¼ to ½ of the key width) enters the letter L. Likewise dragging from anywhere on that square toward the left side enters the plus symbol '+'.

3.1 Soft Key Performance

The framework presented in [10] and [16] is used to model MessagEase's soft keyboard and quantify its theoretical maximum speed. Again to enable us to compare our results with those of [16], we used their published parameters: (**a** = 0 and **b** = 1/4.9 sec.). But as suggested in [16] for the special cases where the same key is tapped twice (i.e., no movement for the second tap) **a** = 0.127 sec. was assumed for the second tap. Unit size is assumed for all keys. Therefore, Time to tap Letter L requiring one single tap on Key i is:

$$TL_i = (1/4.9)\log_2[(D_{0\text{-}i}/W)+1]; \qquad \text{if } D_{0\text{-}i} > 0,$$
$$TL_i = \mathbf{a}; \qquad \text{if } D_{0\text{-}i} = 0 \qquad (6)$$

Where $D_{0\text{-}i}$ is the distance from the stylus' previous position to the center of key i. If the stylus is already over that key, then **a** is taken as the time to tap the stylus on key i; this is the time it takes to bring the stylus down and then up.

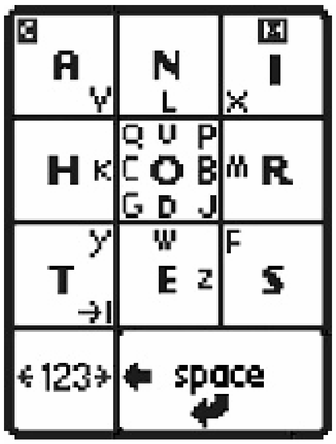


Figure 5. MessagEase's soft keyboard layout (Palm)

For modeling the time taken to enter a less frequent letter L requiring dragging from key j toward key k, we observe:

$$TL_{jk} = t_{0\text{-}j} + t_{down} + t_{j\text{-}k} + t_{up} \qquad (7)$$

Where $t_{0\text{-}k}$: the time to move to key j,

t_{down} : the time to bring the stylus down,

$t_{j\text{-}k}$: the time to move from key j to key k, and

t_{up} : the time to bring the stylus up.

Rewriting (7):

$$TL_{jk} = (t_{0\text{-}j} + t_{down} + t_{up}) + (t_{j\text{-}k} + t_{up} + t_{down}) - (t_{down} + t_{up}) \qquad (8)$$

Which results in:

$$TL_{jk} = TL_j + TL_k - a \qquad (9)$$

Where TL_j and TL_k are computed according to equation (6). In other words, a drag from key j to key k is modeled as tapping key j followed by tapping key k, less the time it would take to tap key k again.

We also observe that unlike the physical keyboard model, with a soft key model it is unnecessary to travel the whole distance from the geometrical center of one key to the geometrical center of another while dragging. In practice tapping anywhere on a key or dragging from anywhere on a key *toward* the target key and traveling about ¼ to ½ of the key width works conveniently. This is particularly applicable to MessagEase's 9-key design which has most of the letters entered by dragging away or toward the center key. Figure 6.a depicts the effective area of the keyboard for entering letters and space (the shaded area).

To account for this shorter travel time (while keeping W=1) for each of the peripheral keys and the SPACE key (keys 1, The position of the EC of a key **k** is defined by moving the geometrical center (GC) of key **k** closer to that key's edge which is closest to the center key (Figure 6.b), such that:

$$D_{EC\text{ to edge}} = (1/2)^{½} * D_{GC\text{ to edge}} \qquad (10)$$

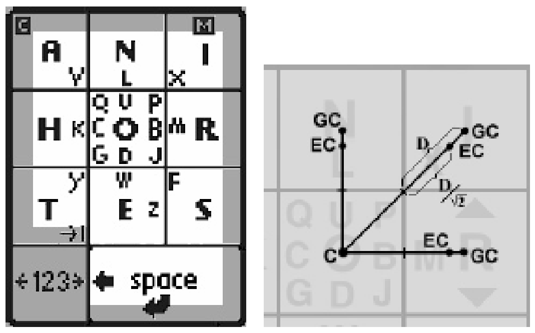


Figure 6. (a) Effective area of the keyboard for typing letters + space, (b) Finding the effective center (EC).

In computing the distances traveled instead of the GC, the EC of these keys are used to reflect the shorter drags and "hub" configuration of this method.

In computing the distances traveled instead of the GC, the EC of these keys are used to reflect the shorter drags and "hub" configuration of this method. Our design was further optimized by enlarging the soft-key assigned to the "space" character since Space is the most frequent character found on a body of text (19%)

Based on the above formulation, considering the larger, more tragetable spacebar, using the framework presented in [16], and accordingly modifying the spread sheet developed for the physical key MessagEase the resulting speed for our soft keyboard design is 50.1 WPM. Compared to the QWERTY soft keyboard [16], our design is measured to be 67 % faster.

3.2 Comparison with Other Soft Key Designs

Reference [16] compiles a set of performance numbers computed with the same framework, and using the same parameters and assumptions. This compilation makes it possible for us to compare the performance of MessagEase soft keyboard with four other soft key layouts.

First, as the benchmark, the soft key implementation of QWERTY keyboard of [10] and [16] has 27 keys (letters + space bar), with only uppercase keys. No mechanism for entering punctuation marks is included. Its speed is measured to be 30 WPM[16].

Fitality keyboard of Textware Solutions [15] is also presented with its letter-only keyboard. It consists of 28 keys (two space keys) manually arranged to enhance speed by grouping the most frequent letters in the center. Fitaly's computed speed [16] is 36 WPM. In its basic form there is no provision for entering numbers or punctuation marks. But on its website, Texware Solutions has added an extended soft keyboard with 10 additional keys for the basic punctuation marks and 17 additional keys for the mode switches, cursor control, and other functions. With 55 keys, the extended Fitaly keyboard measures twice as large as the letters-only keyboard.

The improved Opti soft keyboard of [10] has 30 keys (four space keys) and is designed for uppercase letters only. It is also a design based on letter frequency di-grams. Reference [16] computed its speed at 38 WPM.

Metropolis soft keyboard of [16] is designed based on letter frequency di-garms also. It sports novel hexagonal keys to improve proximity of keys. Metropolis' keyboard has 33 keys (26 letters, space, shift, return, and four basic punctuation marks. It provides no apparent facility to enter numbers or other punctuation marks. Its speed is calculated at 43 WPM.

In Comparison, the soft keyboard implementation of MessagEase has 12 keys. It enables the full text entry of 52 lowercase and uppercase letters, numbers and up to 53 symbols or special characters each with a single drag or slide of the stylus. Using the same framework presented in [16], the speed of the soft-key implementation of MessagEase is measured to be at 50.1 WPM.

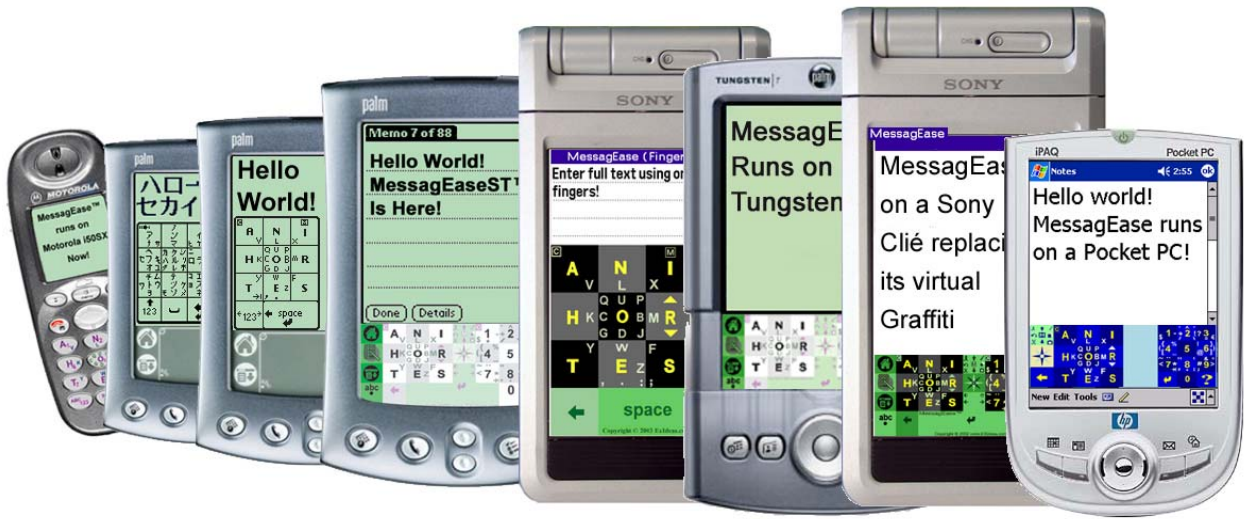
With the same total area, the area of each key of MessagEase is 2-5 times bigger than the area of a key of other soft keyboards mentioned here. Relatively bigger keys can be expected to ease targeting and reduce error rate. Also, the bigger keys may allow the users to enter text with one's finger, rather than the stylus.

According to [16] the 50.1WPM speed thus computed for MessagEase soft keyboard is based on a very conservative assumption of 4.9 bits/s Fitts' law IP. Plugging in IP = 6, then the performance of the MessagEase soft keyboard is calculated as 62.7 WPM.

4. IMPLEMENTATION

We implemented both MessagEase Soft-Key and hard-Key in order to assess their ease of use and user acceptability. We implemented four software tools running on Palm OS PDAs (Fig. 5: b-e). One of these implementations replaces Palm's on-screen QWERTY keyboard with that of MessagEase. Another transforms the Palm's Graffiti area into a MessagEase keyboard. A third implementation of MessagEase sports a large (5 cm $\times$ 5 cm) keyboard which is efficiently operated by a single finger. We implemented MessagEase on a Motorola i50sx cell-phone (Fig. 5:a) demonstrating its hard-key capability. We also applied the MessagEase's principles and created a Palm OS soft keyboard for entering Japanese (Katakana) text. This Katakana implementation demonstrates that MessagEase can be applied to different languages —even to those with radically different alphabets— to optimize and simplify text entry.

The above implementations running on stand-alone devices are available for downloads from our website. Additionally we have several on-screen simulators of hard key and soft-key implementations of MessagEase available on our website



MessagEase™ runs on Motorola i50SX Now!
ハロー セカイ
Hello World!
Memo 7 of 88
Hello World! MessagEaseST Is Here!
Enter full text using on fingers!
MessagE Runs on Tungsten
MessagEas on a Sony Clié replaci its virtual Graffiti
Hello world! MessagEase runs on a Pocket PC!
space

(a) (b) (c) (d) (e) (f) (g) (h)

Figure 7. Current implementations of MessagEase: (a) Hard-key implementation for a cell phone, (b) Japanese (Katakana), (c) On-screen keyboard for Palm, (d) Graffiti replacement for Palm, (e) large keyboard operable with a single finger on a Sony Clié, (f) replacing the keyboard for Tungsten T, (g) with virtual Graffiti on a Sony Clié, and (h) MessagEase keyboard for Pocket PC.

5. DISCUSSION

This paper has employed text-entry speed, calculated by applying the Fitts' law, to measure the performance of MessgEase's text entry system and to compare it with other existing text entry systems. However another important factor to be considered is user acceptability. To ascertain that, we made our earlier MessagEase software tools available for download to the Palm users' community. The response was overwhelming: so far we have had more than 210,000 downloads. Based on this favorable response, we continued developing various tools for Palm OS using on our technology, the latest of which (MessagEaseST) is being sold online as a shareware product. More than 1500 of our users have joined our users' group (http://groups.yahoo.com/group/messagease) urging us to continue implementing this technology as new tools and for other devices. While we have not conducted a formal longitudinal study for MessagEase, many unprompted responses from our users who have converted to using MessagEase after having tried other available methods attest to the ease-of-use and short-learning-curve of our technology.

5.1 Disadvantages:

- MessagEase calls for a new letter assignment on the keys and therefore requires learning.
- Its keys, although bigger, require a more "cluttered" key legend, especially on its central key.

5.2 Advantages:

- It unifies both the hard-key and soft-key devices by providing an efficient text entry system applicable to both.
- It provides *full* text entry (i.e., letters, numbers, special characters, and even extended ASCII set) with the same consistent overall rule set.
- It is size agnostic: since given the overall keyboard area, the size of MessagEase's keys are bigger than the comparable keyboard, MessagEase can be implemented in a very small area. Given a larger keyboard area (but not large enough to suit a full, ten-finger QWERTY), MessagEase can be operated with a single finger, much more efficiently than other keyboards.
- It is deterministic and unambiguous (e.g., in comparing with multi-tap).
- With its limited number of keys, it can be used one-handed, for touch typing, or by the blind, tasks that are not easy to achieve with the miniaturized QWERTY keys.
- It is applicable to any language, using any alphabet, including Kanji, Korean, Arabic, or Hebrew.

In our opinion the disadvantages mentioned above are far outweighed the advantages of this novel text entry system.

6. SUMMARY AND CONCLUSIONS

This paper has presented a novel text entry system for small electronic devices with a unique keyboard based on letter frequency and positional matrix. This keyboard is applicable to hard-key devices with limited number of keys (e.g., a cell phone, or a TV remote controller) as well as soft-key devices where only a limited area is available for keyboard implementation (e.g., a PDA, or a tablet computer). The text entry system presented here provides full text entry (full ASCII 220) and is adaptable for any language. Comparative analysis of this system using Fitts' law has resulted in speeds between 50 to 63 WPM for the soft-key version and 30 to 37 WPM for the hard-key version. This system requires a relatively small physical and memory footprint. Currently several tools implementing this text entry technology are available for Palm OS PDAs and our user experience has indicated that it has a relatively short learning curve.

7. ACKNOWLEDGMENTS

The author wishes to thank Professor Scott MacKenzie of York University, Dr. Shumin Zhai of IBM Almaden Research Center, Dr. Bahram Afshari of Agilent Technology, and Dr. Hamid Najafi of Wireless Link for their insightful and stimulating, comments.

8. REFERENCES

[1] Ackermans, P., Using Predictive Text Input In MMI by ECN, October1, 2000.

[2] Blickensorfer, C. H., Instant Text and the One-Finger Keyboard, A very efficient method for entering text with a keyboard or a pen. *Pen Computing*, December 1995.

[3] Grover, D. L., King, M. T., and Kuschler, C. A. Patent No. US5818437, *Reduced keyboard disambiguating computer*. Tegic Communications, Inc., Seattle, WA (1998).

[4] Guernsey, L. Playing taps on the cell phone, *New York Times* (2000, October 12), D9.

[5] http://www.digitwireless.net

[6] http://www.handykey.com

[7] Mayzner, M.S. and M.E. Tresselt, Tables of single-letter and digram frequency counts for various word-length and letter-position combinations. *Psychonomic Monograph Supplements*, 1965. 1(2): p. 13-32.

[8] MacKenzie, I. S., Fitts' law as a research and design tool in human computer interaction. *Human Computer Interaction*, 1992. 7: p. 91-139.

[9] MacKenzie, I. S., Kober, H., Smith, D., Jones, T., & Skepner, E. (in press). LetterWise: Prefix-based disambiguation for mobile text input. To appear in *Proceedings of the ACM Symposium on User Interface Software and Technology – UIST 2001*. New York: ACM.

[10] MacKenzie, I. S., Zhang, S. X. (1999) The design and evaluation of a high-performance soft keyboard. *Proceedings of the ACM Conference on Human Factors in Computing Systems - CHI '99*, pp. 25-31. New York: ACM.

[11] MacKenzie, I. S., & Zhang, S. X. (in press). An empirical investigation of the novice experience with soft keyboards. To appear in *Behaviour & Information Technology*..

[12] Motorola Lexicon Division, http://www.motorola.com/lexicus/index.html.

[13] Silfverberg, M., MacKenzie, I. S., & Korhonen, P. (2000). Predicting text entry speeds on mobile phones. *Proceedings of the ACM Conference on Human Factors in Computing Systems – CHI 2000*, pp. 9-16. New York: ACM.

[14] Soukoreff, W., and MacKenzie, I. S. Theoretical upper and lower bounds on typing speeds using a stylus and keyboard, *Behaviour & Information Technology 14* (1995),370-379.

[15] TextwareSolutions: http:// http://www.textwaresolutions.com .

[16] Zhai, S., Hunter, M., Smith, A. S., The Metropolis Keyboard – An Exploration of Qualitative Techniques for Virtual Keyboard Design. *Proceedings of the ACM Symposium on User Interface Software and Technology – UIST 2000*. San Diego, California pp 119-128.