LetterScroll: Text Entry Using a Wheel for Visually Impaired Users

Hussain Tinwala
Dept. of Computer Science and Engineering, York University
4700 Keele Street
Toronto, ON, CANADA M3J 1P3
hussain@cse.yorku.ca

I. Scott MacKenzie
Dept. of Computer Science and Engineering, York University
4700 Keele Street
Toronto, ON, CANADA M3J 1P3
mack@cse.yorku.ca

CHI 2008, April 5 – April 10, 2008, Florence, Italy
ACM 978-1-60558-012-8/08/04

Abstract
Four text entry techniques for visually impaired users are presented. *LetterScroll* uses a mouse wheel to maneuver a cursor across a sequence of characters, and a button for character selection. Keystrokes per character (*KSPC*) vary from 6.97 to 2.68. After extensive analyses and pilot testing, two variations were chosen for initial evaluation. Method 1 (M1) uses the mouse alone to enter text. Method 4 (M4) also uses the keyboard to access vowels. In a study with seven blindfolded participants, entry rates averaged 2.9 wpm for M1 and 4.4 wpm for M4. Error rates for both methods were about 3.4%.

Keywords
Visually impaired, text entry, auditory feedback, voice synthesis, mouse wheel

ACM Classification Keywords
H5.2. Information Interfaces and Presentation, User Interfaces: Input Devices and Strategies

Introduction
Increasingly, text-based tasks are shifting from physical channels, such as paper, to electronic channels, such as mobile phones and PDAs. A key question is, are these electronic devices usable by visually impaired users and, if so, to what extent? In general, text entry is highly reliant on vision, leaving visually impaired users at a disadvantage. Small

devices such as mobile phones rarely include Text-To-Speech and this makes the simple task of sending a text message (SMS) difficult for visually impaired users.

LetterScroll: Entering Text by Scrolling

LetterScroll is a text entry technique that extends an earlier three-key technique [2] to include visually impaired users. The underlying concept is the date stamp method [1] where increment and decrement operations maneuver a cursor through a character set. A select operation inputs the character. Our character set uses the 26 letters in the English alphabet. The set can be expanded to include numbers, diacritics, and special symbols such as punctuation.

LetterScroll uses a mouse scroll wheel to move the cursor through the character set (Figure 1). *LetterScroll* allows jumping to vowels using additional keys on the device (such as a mobile keypad or a keyboard).

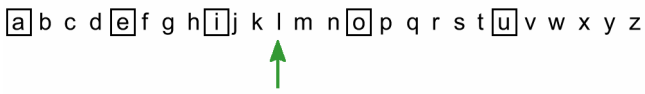

Figure 1. A model for conceptualizing the character set

Since vowels are easily recognized and recalled, they are given special treatment to decrease the cognitive load on the user and increase throughput.

To select a character, users "click" the primary mouse button. The cursor persists at the last position with subsequent navigation proceeding from that point. Upon completing a word, the user clicks the secondary mouse button to insert a space.

FEEDBACK

LetterScroll makes extensive use of speech feedback. Each character is mapped to an auditory sound. A speech synthesizer speaks the character at the current cursor position. Since the speech synthesizer is slow, the current character in the speech buffer is aborted if the cursor quickly moves to a new character. Upon selection, the letter is spoken (again). When a SPACE is inserted, the speech synthesizer speaks "space".

To review entered text, the user presses ESC (character by character) or CTRL-ESC (entire phrase). These keys are selected for their boundary positions, allowing access using the tactile sense. For experimentation, the SPACE key is employed to review the presented text.

Modeling the Interaction

We built a model of the interaction to explore design scenarios. The main statistic for this is keystrokes per character (*KSPC*) – a weighted average of the number of keystrokes required to generate a character in a given language using a given text entry technique [1]. For instance, *KSPC* for QWERTY is 1 since each character has a dedicated key. For *LetterScroll*, each increment or decrement is counted as a keystroke.

There are many ways to implement *LetterScroll* and each variation can strongly influence *KSPC;* this makes a priori analyses worthwhile. Four variations are shown in Table 1.

Table 1. Four variations of *LetterScroll* (see text for details)

Method	Character Navigation	KSPC
#1	Scroll both ways	6.97
#2	Scroll and accelerate	4.94
#3	Scroll and jump [`,1]	3.21
#4	Scroll and jump [`, 1, 2, 3, 4]	2.68

The basic mode is Method #1 (M1). Our language model is based on a word-frequency reduction of the British National Corpus [4]. Below is an example of

three of the most frequent words in the list appended with their frequency and M1 keystrokes:

```
the_  5776384 ZZZZZZZZCBBBBBBBBBBBBCBBBS
of_   2789403 ZZZZZZZCBBBBBBBBBCS
and_  2421302 ZZZZZZZZCFFFFFFFFFFFFFCBBBBBBBBBBCS
```

Initial navigation appears as "Z". These keystrokes are based on a weighted average using trigram frequencies crossing word boundaries. "C" is the select keystroke. "B" and "F" are backward and forward keystrokes, respectively. "S" is the final SPACE to select a word. *KSPC* is the overall weighted average of the keystrokes per character. For M1,

$$KSPC = 6.97 \quad \text{(M1)}$$

There are two paths to reach the desired character (backwards or forwards). Our analysis uses the shortest path and therefore yields a lower bound for *KSPC*.

To improve interaction, M2 adds acceleration by holding CTRL to scroll two characters at a time. Thus, scrolling from *a* to *c* requires one increment instead of two. An additional keystroke is added for pressing the accelerator key. *KSPC* is 29% lower:

$$KSPC = 4.94 \quad \text{(M2)}$$

Although M2 is promising, a potential improvement is to jump – to use keys to navigate a subset of the character set. For this we again consider the tactile sense and choose the two top-left keys. On our keyboard, these are the *back quote* (`) and *1* keys. They are associated with backward and forward navigation, respectively. Consecutive presses of *1* traverse *a, e, i, o, u* and back to *a*. The *back quote* key exhibits reverse behavior (*u, o, i, e, a,* and back to *u*).

A similar technique is a 1:1 mapping of vowels to keys. Five keys are dedicated to the vowels. This is M4. The keys selected are *back quote*, *1*, *2*, *3*, and *4* (see Figure 2). The index finger is shared between *o* and *u*.

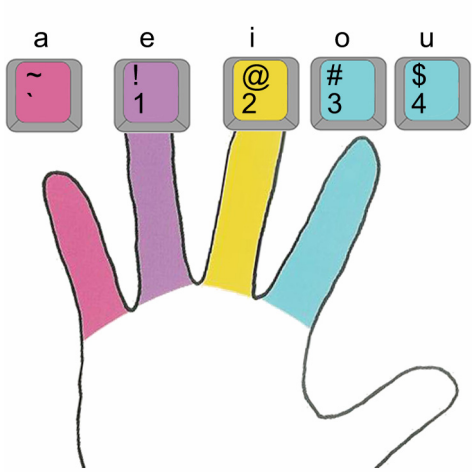

Figure 2. Mapping of fingers to keys

In M3 and M4, there are multiple ways to navigate the character set. For example, navigating from *a* to *t* can be done at least four ways:

1. Scrolling forward to *t* (*a b c d … s t*).
2. Scrolling backward (*a z y x w v u t*).
3. Jumping to the vowel *o* followed by scrolling forward (*o p q r s t*).
4. Jumping to the vowel *u* followed by scrolling backwards (*u t).*

The last technique requires fewer keystrokes. For both M3 and M4, *KSPC* is calculated using the shortest path possible. The results are

$$KSPC = 3.21 \quad \text{(M3)}$$

$$KSPC = 2.68 \quad \text{(M4)}$$

Although *KSPC* suggests that M4 requires 62% fewer keystrokes than M1, it is not certain that M4 will result in higher throughput, as our analysis does not account for cognitive load and attention demands of jumping.

All four variations were implemented and tested with two users. Overall, it was determined that M1 was easy to use and M4 provided substantial benefit in terms of keystroke savings. At this work-in-progress stage, we present an initial empirical evaluation comparing M1 and M4.

Method

Participants, Apparatus, Procedure, and Design

We recruited seven volunteer participants from the local university. The apparatus was a Windows XP notebook running software written in Java and a Java speech synthesizer – *FreeTTS*. At the top of Figure 3 are the stimulus (presented text) and result (transcribed text).

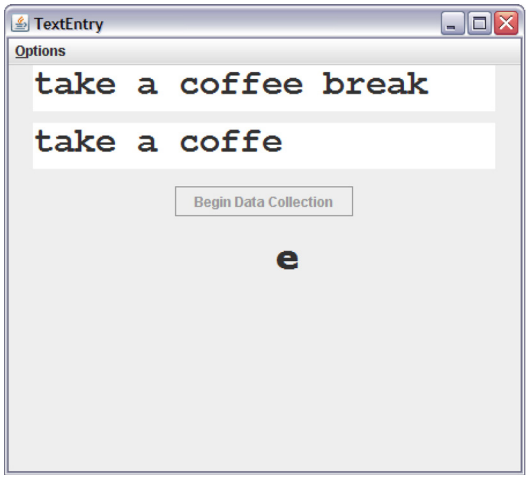

Figure 3. The user interface in the experiment

Phrases were drawn randomly from a standard 500-phrase set [3]. Six phrases (P1 to P6) were entered for each condition. The letter in the middle displays the current location of the cursor. Data collection begins by clicking the *Begin Data Collection* button.

Participants were blindfolded with a sleep shield and wore earphones for speech feedback (Figure 4).

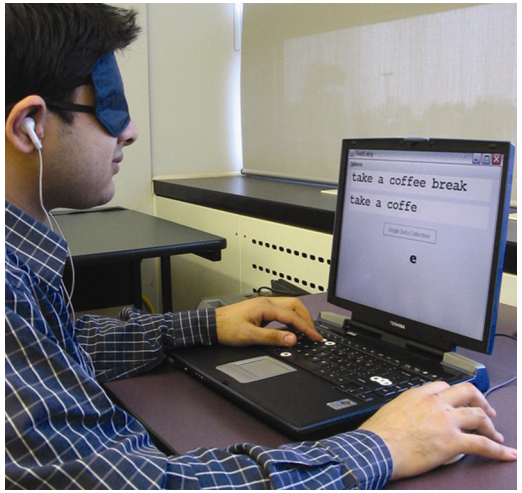

Figure 4. Experimental apparatus used for *LetterScroll*

Participants were instructed to proceed at a pace comfortable to them and to do so as accurately as possible. If an error occurred, they were to continue without correcting it. The software and jumping features were demonstrated before testing. Two minutes of practice were allowed without the blindfold before beginning.

Results and Discussion

Speed and Accuracy

The overall mean for entry speed was 3.6 wpm. M4, at 4.4 wpm, was 33% faster than M1 at 2.9 wpm (Figure

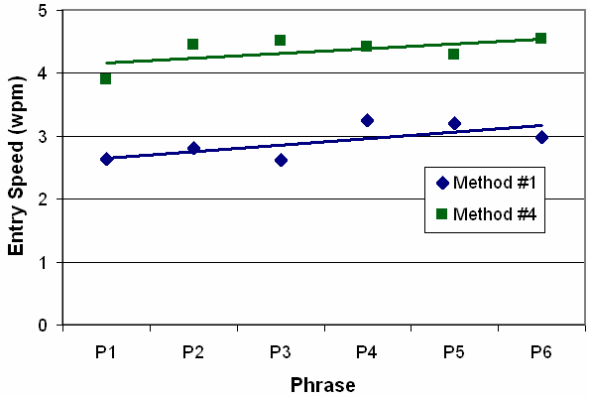

Figure 5. Results for text entry speed (wpm) by method

5). The difference was statistically significant ($F_{1,6}$ = 16.6, p < .01).

Initially, participants listened to every letter, scrolling forward or backward one letter at a time. By the second phrase, they began to scroll across multiple letters and then inspect the current position of the cursor with the feedback. This strategy allowed faster navigation wherein participants would "accelerate" by scrolling to the vicinity of the desired character.

Overall, error rates were 3.5% for M1 and 3.9% for M4. The difference was not statistically significant ($F_{1,6}$ = 0.169, ns).

KEYSTROKE DISTRIBUTION

A deeper analysis involves investigating keystroke behaviour for the two methods. See Figure 6.

In M1, participants used scroll forward 49% of the time. This decreased significantly to 28% when using M4. A similar decrease was found for scroll backward. As a result of introducing the vowels in M4, 28% of the keystrokes were attributed to *a, e, i, o,* and, *u.* The

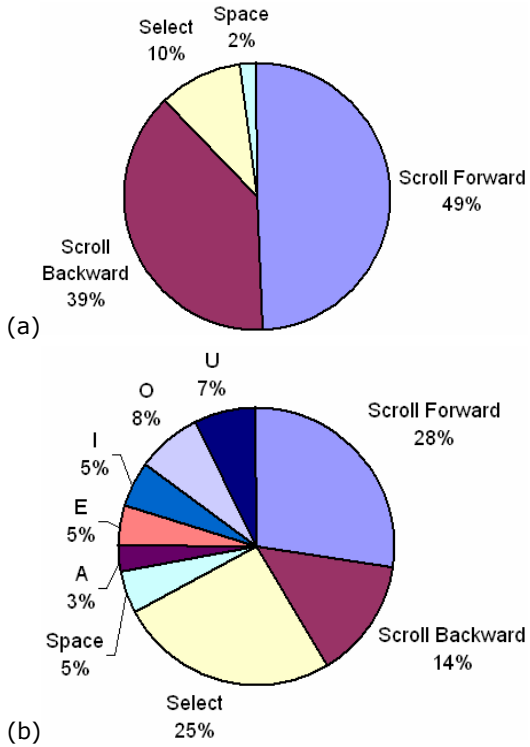

Figure 6. Keystroke distribution by method. (a) M1 (b) M4

ratio of scroll forward to scroll backward in M1 was approximately 5:3. In M4, this ratio changed to 2:1. Evidently, participants found the vowels convenient and made extensive use of them to reach the desired character. This was verified in interviews with participants after testing.

Stimulus and Input Review

Since the feedback was auditory, participants could inspect the stimulus and result as needed. Overall, they reviewed the stimulus 1.5 times per phrase for M1, and 0.9 times per phrase for M4. They reviewed the result

0.8 times per phrase for M1, and 0.45 times per phrase for M4.

Interestingly, the overall review frequency for M4 was less than for M1. It is likely that higher entry rates for M4 allowed participants to retain the stimulus and result longer, thus requiring fewer reviews of the text.

Limitations of the Apparatus
Although the apparatus was functional, it had some limitations. The voice synthesizer was of moderate quality. Perhaps replacing the alphabets and phrases with clearer, pre-recorded audio files would be better.

Approximately 50% of the participants used the ability to review input to identify the last few characters entered. Participants could potentially be less distracted if the interface allowed them to simply inspect the last word or last few words instead of the entire phrase.

Participant Questionnaire
A post-test participant questionnaire collected subjective preferences of participants. Five suggested that the quality of the speech feedback could be better. All participants found M1 "very easy" and M4 "moderately easy". However, M4 was the preferred method due to the fatigue induced by M1.

Conclusion

Four variations of text entry using a wheel mouse, keyboard, and spoken text were explored. The scroll wheel on the mouse navigated a cursor along a linear sequence of characters. Characters were selected by clicking the primary mouse button and a space was inserted by clicking the secondary button. An experiment compared two methods (M1, M4) with seven blindfolded participants who entered a total of 84 phrases of text. Overall, text entry rates were 2.9 wpm for M1 and 4.4 wpm for M4.

M4 engaged the keyboard to provide random access to vowels. There was a significant improvement in text entry rate, but it was not as large as expected. This suggests that there are other factors at play, such as the attention demands in M4 to determine which vowel is closest to the desired character.

The visual channel was completely blocked. As a result, certain behaviors were exhibited that challenge conventional concepts. For instance, accuracy was defined as the degree to which the transcribed text matches the presented text; but this does not capture the true errors. There are other cognitive processes that resulted in participants entering phrases that were different from the presented phrase, yet similar or reasonable to the presented phrase. Further research is needed to explore how to define accuracy in such circumstances.

References

[1] MacKenzie, I. S. KSPC (keystrokes per character) as a characteristic of text entry techniques. *Proc MobileHCI 2002*, Springer-Verlag, 195-210.

[2] MacKenzie, I. S. Mobile text entry using three keys *Proc NordiCHI 2002*, ACM Press, 27-34.

[3] MacKenzie, I. S. and Soukoreff, R. W. Phrase sets for evaluating text entry techniques. *Proc CHI 2003*, ACM Press, 754-755.

[4] Silfverberg, M., MacKenzie, I. S. and Korhonen, P. Predicting text entry speed on mobile phones *Proc CHI 2000*, ACM Press, 9-16.