An On-Screen Keyboard for Users with Poor Pointer Control

Rick Kjeldsen

IBM T.J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598
fcmk@us.ibm.com

Abstract. This paper describes a novel design for an on-screen keyboard that makes it easier for some users with relatively poor pointer control to type. The keyboard uses the edge of the screen to reduce the dimensionality of the pointing task required to identify a character, and uses gesture, rather than click or dwell to select it for typing. A prototype keyboard is used in a study involving people with and without physical impairments using several types of pointing devices. The results show that this approach has some significant advantages compared to a conventional two dimensional on-screen keyboard, especially with some types of pointing devices.

1 Introduction

Users who have Cerebral Palsy, Multiple Sclerosis, spinal cord injuries or similar motor impairments often have difficulty using physical computer input devices like a keyboard or mouse. For many, moving the pointer is possible, using devices including head trackers, joysticks, trackballs, touchpads or even standard mice, but typing on a physical keyboard is very difficult. These users often rely on on-screen keyboards for textual input.

While the design of on-screen keyboards for mobile devices has benefited from a great deal of innovation recently, the design of keyboards for users with physical impairments seems to be dominated by older designs that arrange characters in a two dimensional grid patterned after their physical counterparts. In our experience, this approach is not ideal for many users.

Standard on-screen keyboards assume that the user has sufficient accuracy of pointer control to easily identify a small two dimensional region of the screen, and the ability to click while hovering at one spot. Users with physical impairments often can control a pointer only coarsely. It may be difficult for them to keep the pointer still, and fine motor coordination may be insufficient to move it accurately over small distances. Once the pointer is on a character, a user may have trouble clicking to type it without moving the pointer. Often users will click inadvertently as they move about the screen, causing unexpected and undesirable actions. As an alternative to point&click typing, point&dwell is often provided. Here the user must keep the pointer still for a dwell period, after which a click is automatically generated. If a user has trouble keeping the pointer still, as might be the case with a head tracker for example, this pause can be a problem. Waiting for each character in this way also makes typing very slow.

We have developed an on-screen keyboard that attempts to address these issues. The characters are placed along the edges of the display and normally hidden. A user types a character by moving the pointer to the edge of the display to pop-up the keyboard, moving along the edge to select a character, then moving away from the edge to type that character.

By placing the characters along the edge of the display, we remove one dimension along which the user needs to accurately position the pointer. This can be thought of as "bracing against" the side of the display. Just as it is by bracing the heal of your hand against a table you can position a pencil to write far more accurately than by holding your hand floating above the paper, so we propose that by bracing against the edge of the display a user can position the pointer in one dimension easier and more accurately then by positioning in two dimensions simultaneously. Of course, this technique is based on the assumption that the pointer will only move to the edge of the display, and not wrap or move to another screen if the user moves the mouse further in the same direction.

Second, the user need not perform a separate action (e.g. pressing a mouse button or other switch) to type the character. By using the same modality for typing as for selection, we reduce the number of different inputs a user must provide, hopefully reducing the complexity of the interaction. At the same time, moving away from the screen edge to type is quicker than waiting for a dwell time to elapse, allowing users with better dexterity or more experience to type faster.

This keyboard was originally developed in conjunction with a head tracking pointer. In that context this style of typing is a natural fit. It is easy to “overdrive” the pointer against the edge of the screen so that it does not pull away (assuming the system does not readjust when pushed against the side of the screen, as some do), and with no inherent mechanism to generate a click, typing by pulling away from the edge is more natural than dwelling over a key. If the keyboard is on the top edge, the typing motion is to tip the head back to open the keyboard, aim left or right to select a character, then tip the head down again to type. In short order this sequence feels very natural and fluid. We have since begun to explore the suitability of this approach for other input devices, and this paper reflects some of that work. We will describe the keyboard, and the results of a study involving users with and without physical impairments using various pointing devices.

2 Related Work

A number of commercial on-screen keyboards are intended for users with physical impairments, for example .the Grid (www.zygo-usa.com/grid.html), Madentec’s Discover Screen (www.madentec.com) or WiViK (www.wivik.com) to name just a few. While these focus on the needs of users with physical impairments, most do it without straying far from the traditional key layout (a 2D grid in QWERTY or alphabetic order). These systems generally support various typing styles including point&click, point&dwell, and scanning, but the design of the keyboard itself does not take into account the issues faced by a user who has difficulty pointing accurately.

Work that does address the design of the keyboard itself is usually focused on configuring the layout of the keys for efficient typing, generally in consideration of Fitts' Law [8]. By altering the spacing and layout of the keys, the time to move between them can be minimized, improving typing speed [7]. Unfortunately there is some evidence that such an analysis may not apply well to users with physical impairments such as Cerebral Palsy [1].

Interesting work has been done on the design of on-screen keyboards for mobile devices. Most of these are designed for stylus input, but some are relevant in this context. In [3] Mankoff and Abowd present a keyboard that types as the pointer enters keys laid out in a circular pattern. This allows users to type using gestures similar to cursive writing, as they move from letter to letter, but it requires a high degree of precision and so is not well suited to users who have difficulty accurately manipulating the pointer.

Wobbrock has designed the EdgeWrite system which he has adapted for PDAs, and more recently Trackballs [6]. This system allows a user to "type" by sketching approximations of letters around the edges of a square, or in the case of the trackball implementation by "pulsing" the pointer in the appropriate directions to simulate the same interaction. The system is well suited to people with poor pointer control using physical input devices, because the interaction can occur within a physical rectangle which helps constrain the user's motions. It is not clear how well this approach would work with a non-contact pointer control method such as a head tracker.

Dasher[5] is an interesting alternative to a standard keyboard where users select letters by moving the pointer into expanding colored regions which travel across the screen. This makes typing a continuous gesture, however the visual presentation is always changing, which can make it intimidating to novices and cognitively impaired users. It is also not clear how the system will perform for a user with poor hand-eye (pointer-eye ?)coordination.

3 Keyboard Description

The ScreenEdge Keyboard is simple in concept. Whenever the system pointer moves to the top edge of the screen, the characters in the alphabet appears along the top edge (Figure 1). The user moves the pointer left and right to highlight the desired character, then down from the top of the screen to type it. Just as with a traditional keyboard, the active window receives the character.

In the prototype, this primary keyboard area holds the characters of the English alphabet (A-Z), plus BackSpace. By default these characters fill the top edge of the screen. The size of the keys can be enlarged horizontally to make them easier to select. Since only a portion of the keyboard is now visible, the keyboard scrolls when the pointer reaches the edge of the screen (Figure 2).

In order to provide more space for characters, when the pointer reaches the left or right edge of the keyboard (i.e. enters the "<" key on the left edge or the ">" key on the right edge – the user may have to scroll to get there) another keyboard opens along the edge of the screen (Figure 3). This keyboard is navigated by traveling up and down along the edge of the screen, and characters are typed by moving away from the edge. These side keyboards can also scroll up and down, though in the current versions they need not. The keys on the bottom of these side menus are Space and Enter so that the user only need move the pointer to the top corner, then straight down for these common operations.

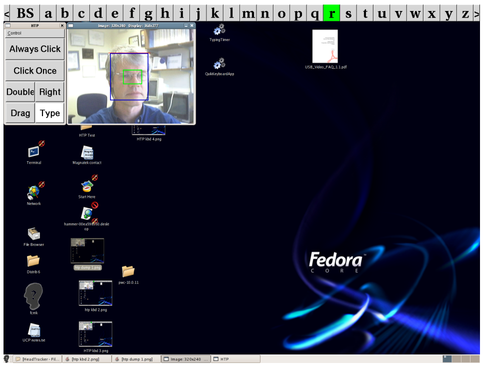

Fig. 1. Keyboard in use with a head tracking pointer

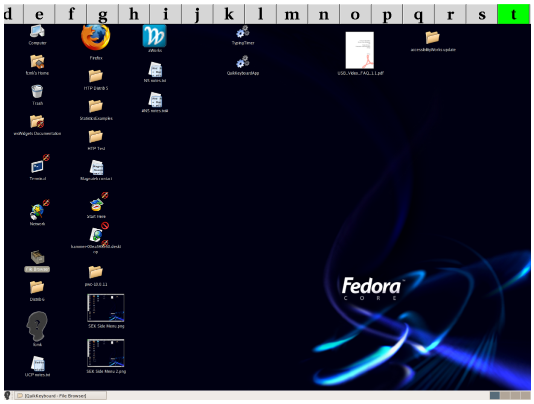

Fig. 2. Screen Edge Keyboard, enlarged for better readability, scrolls when the pointer reaches the edge

The current prototype has been optimized to support web browsing, especially the typing of URLs. Toward this end, the right menu contains several shortcuts, such as single keys for " www." and ".com". There is currently no provision for capitalization, and not all the keys on a traditional keyboard are supported. These shortcomings will be addressed over time.

This basic algorithm worked very well with a head tracking pointer, for which it was originally designed, but when using the keyboard with a mouse or trackball, new users would often type characters inadvertently. With a head tracker or joystick, it is easy to pin the pointer against the top edge of the screen as it is moved back and forth

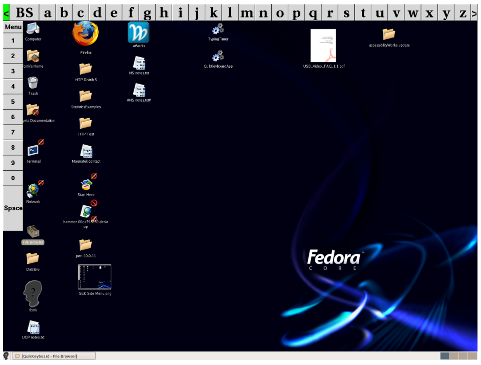

Fig. 3. When the pointer reaches the left or right of the keyboard, a menu appears down the side of the screen

to select a character. With a mouse or trackball, however, no matter how far up the pointer is pushed, a very small down movement will take it away from the screen edge again – typing a character. To complicate matters, with a mouse, moving the pointer straight left or right without moving it down at all is difficult. As a result, users have to continually move the pointer up as they moved left and right, to keep it against the top edge, which causes the mouse to move further and further from the user as they work. This is an inconvenience for unimpaired users, but it is a real problem for users with a limited range of hand movement. Early trials showed that because extending their arm is difficult, many such users move a mouse left/right by swinging their arm left and right about the pivot of their elbow. As the mouse moved in this arc, it would pull away from the top edge, typing a character.

To improve performance with mouse pointers, we modified the behavior of the keyboard so that it captures the pointer within it till a character is typed. Instead of simply looking for any motion away from the screen edge, the system now types only when a pointer movement is above 55 degrees relative to the edge. If the motion is less than this angle, the pointer is captured within the keyboard as it moves, meaning that the component of movement parallel to the screen edge is retained, but the component perpendicular to the edge is forced to remain within the key board. The net effect is that typing requires a more precise movement directly away from the screen edge, and until this occurs the user can only move left and right within the keyboard. To allow users to exit the keyboard without typing, if the pointer pauses for more than 3 seconds the keyboard will disappear. A movement directly away from the edge will then exit, and restore normal pointer movement. A movement along the edge will re-open the keyboard and re-capture the pointer.

These changes greatly decreased the number of falsely typed characters. Users with poor pointer control are now captured within the keyboard till they type or pause long enough to exit.

To provide visual feedback we use color. When a key is selected, as the pointer moves back and forth along the edge, it turns green. After a character is typed, it changes to blue. This helps confirm the typing to the user and avoids the need for them to glance down at the window to see the character. After typing, the keyboard remains visible for 2 seconds, both so the user can see the highlight, and to allow them to begin targeting the next character if there will be one. Some users have requested that this time be increased so that the keyboard remains visible longer.

4 Experiment

To evaluate the performance of the ScreenEdge Keyboard (SEK) it was compared to a traditional on-screen keyboard during several days of testing. Because the keyboard has been developed on Linux, the comparison keyboard was the Gnome On-screen Keyboard (GOK) version 1.0.3. Users were asked to type 3 sentences on each keyboard and the average character typing time was used to compare the keyboards.

Twenty three users participated in 29 evaluation sessions. Of these, we were able to get sufficient data from 17 users to make valid within-user comparisons between the keyboards. Only data from these users are included in the following discussion.

Eleven of the 17 were clients at United Cerebral Palsy of Suffolk, NY. All had previous computer experience using a mouse or trackball as a pointing device. All had limited hand and arm movement and used the pointing device with various levels of difficulty. All were capable of using a standard (physical) keyboard, and none had prior experience with on-screen keyboards. Because we did not have time to train them in the use of a head tracker, they used the input device they were most used to. One chose to use a trackball, the remainder used a standard computer mouse.

Six participants had no physical impairment. These were either staff at UCP, peers at IBM Research, or various family members. All had extensive computer experience. All of these participants were asked to use a head tracker [2] as an input device, three had sessions with other devices as well. None had previous experience with head tracking pointers. Two had prior experience with on-screen keyboards.

Each user was given a demonstration of the two keyboards, and then given as much time as they liked to practice with them before timing data was recorded. When they felt ready, they were then asked to type three sentences with each keyboard (6 different sentences per user). The sentences were chosen from a pool of 15 sentences such as "My favorite color is blue" and "It is almost time to go home". The keyboards were used in semi-random order, either taking all data first from one keyboard then the other, or alternating keyboards with each sentence.

Typing in SEK was as described earlier in this paper. SEK was configured to fill the top of the screen without scrolling. Typing in GOK involved clicking on the keys of the Composition window (Figure 4). It can be argued that using GOK with a point&dwell strategy may be a better comparison, as SEK does not require clicking. We decided, however, that the dwell time required to type a character (500-1000 ms per character) would unfairly bias the data toward SEK. Since our subjects were generally able to click well, we chose to use GOK with point&click typing. The keys in GOK were set to be the same width and height as the width of the keys of the SEK keyboard. The characters were arranged alphabetically.

Timing data was recorded by a custom program which recorded the time each keystroke event was received with millisecond accuracy. The users were able to see the characters as they typed them into this program's window.

The results of the sessions are summarized in Table 1. Over all sessions, SEK was about 1 second faster in terms of time-per-character (TPC). The fastest average TPC for one user was 1.9 seconds for a user with no impairments using a trackball on SEK. The slowest average TPC was 23.4 seconds for a user with CP using a mouse on GOK, who had great difficulty keeping the mouse still while clicking. Excluding this user, the highest average TPC was 11.9 seconds for a user with CP using a mouse on GOK.

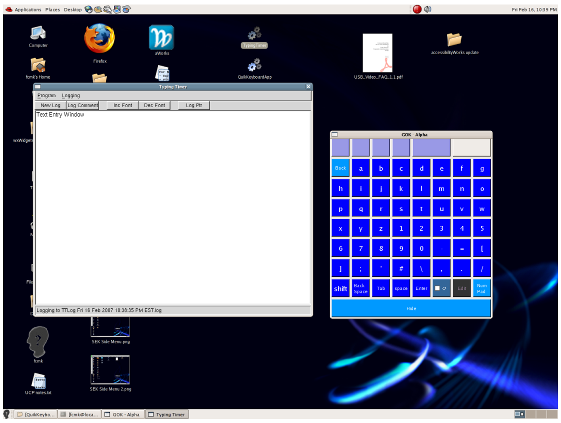

Fig. 4. The Gnome On-screen Keyboard (GOK) Composition window

Table 1. Results of SEK field trial. Times are in seconds / character typed. Note that the absolute times between lines can not be compared because of different sets of users on each line, and the large variability in typing ability between users. Relative times on each line are important.

Line	Group	SEK time	GOK time	Gain w/ SEK
1	Everyone	4.9	6.0	17%
2	User with CP	6.0	7.3	18%
3	Unimpaired users	3.4	4.1	16%
4	CP users except J	5.9	5.5	-7%
5	CP users – 1st session	5.7	4.8	-20%
6	CP users – 2nd session	6.7	8.3	19%
7	All Mouse users	5.4	5.2	-4%
8	All Head Tracker users	4.4	6.2	28%

To really understand the merits of SEK, it is informative to examine the data more closely. The first cut is to separate the data for users with and without motion impairments. Lines 2&3 of Table 1 show the TPC for users with CP versus non-impaired users. In both groups, SEK proved to be about 15% faster than GOK. Looking at the individual session data, however, many of the users with CP were slower with SEK than GOK. This observation fits with comments we received from many of these users that they felt awkward using SEK.

The data for the users with CP was greatly influenced by one user, "J", mentioned above, who had a very large variation between his times with the two keyboards (SEK: 6.9 seconds, GOK: 23.4 seconds). The cause of this was that when he attempted to click on the GOK key, the mouse would often move inadvertently. This is a common issue with some types of disabilities [4], and highlights one of the advantages of the gesture-based typing action used in SEK. This large discrepancy, however, does hide the general trend in favor of GOK. In Line 4 we see that without J, the users with CP were about 7% slower with SEK.

Three of the users with CP participated a second time after using SEK occasionally for one week. To examine the effect of this practice, Line 5 shows users with CP on their first exposure to SEK, and Line 6 shows the average TCP for users on their second session. Here we clearly see that the first time users with CP are about 20% slower using SEK, but after practice the affect reverses. All the repeat users also reported liking SEK better than GOK in their second session. (Note that the longer absolute times on line 6 are because only a subset of the users had a second session)

Users with no physical impairments were universally faster with SEK, regardless of the device they used (mouse, trackball, touchpad or head tracker). They also reported preferring SEK in most cases.

Finally, we examined how various input devices performed with SEK. Lines 7&8 show the average for all mouse users, and the average for all head tracker users (note the mouse users include impaired and non-impaired users, the head tracker users were all non-impaired). The mouse users show about equal performance between the two keyboards. With the head tracker, however, SEK is significantly faster than GOK. This corresponds to the comments received from our participants. All 6 users who used the head tracker reported a preference for SEK. Interestingly, two users who used other devices with SEK for the first time reported preferring GOK, even though their typing times were faster with SEK.

5 Discussion

The numeric results from the study show clearly that the ScreenEdge Keyboard is both faster and preferred when used with a head tracking pointer. Results for other devices are not so clear cut. Users with no physical impairments, and experienced users universally typed faster with SEK using a mouse. First time users with CP were universally slower using a mouse with SEK, and reported liking it less. We suspect this is due to the difficulty these users had learning the gesture-based interaction. Many of them have worked very hard to achieve some skill with point&click interaction. To compensate for their poor pointer control, they have often developed idiosyncratic pointing styles. While a well coordinated user will generally move directly to a target in a ballistic movement, then home in with a short feedback driven phase, many users in our study take less direct paths to the desired target. One good example was S, who would uses a series of curving paths shaped almost like a fish hook, that would get progressively closer to the target. Once inside she would carefully click the mouse button. This suggests that in spite of being able to point and click successfully, these users had not learned to accurately control the path of the pointer in the process. Because typing in SEK depends on the path of the pointer, it was initially more difficult for them to use. There is evidence that given sufficient time these users can perform well with SEK, but it remains to be seen how many users would have trouble learning to control the pointer path sufficiently to type well.

The experience of J (see above) suggests that the gesture-based interaction of SEK is a big advantage for users who have trouble clicking. Because he would move the pointer when clicking, his time per character with GOK's point&click interactions was almost 4 times longer than with SEK. We expect there would be a similar advantage for users who can not click at all, for if dwell times were added to the results, SEK would have bested GOK far more often. Some users, however, reported difficulty NOT clicking when using SEK, and in one case this clearly slowed her down.

Several users commented on the simplicity and lack of clutter with SEK. Having the keyboard appear on the edge of the screen when needed made it far less obtrusive than the large 2D keyboard window.

The most praise for SEK came from those who used it with a head tracker. They liked the nod-to-type motion used for typing characters on the top of the screen. One user reported she felt she was able to anticipate the pointer movement across the top of the screen, and begin the downward "typing" movement as the pointer moved into the desired character, making for a very fluid typing rhythm. Typing on GOK with a head tracker required a dwell, and that pause tended to break her train of thought.

6 Conclusion

We have described a new style of typing for on-screen keyboards where the characters are arranged on the edge of the screen, and the user types by moving to the edge of the screen, sliding along it to the desired character, and moving way again. For users with physical disabilities, this approach has some advantages. Users who have difficulty clicking can type without incurring the overhead of dwelling on each key. The character selection and typing motions form a gesture which users reported as fluid and natural. Users who have difficulty keeping the pointer stable can force it against the edge of the screen, and only need to precisely position the pointer in one dimension during character selection, potentially making character selection easier

Experimental data suggest that while the method works best with head tracking pointers, even with other devices, typing with the screen edge keyboard is faster than with 2D on-screen keyboards once the user has become comfortable with the movement pattern. Users with some types of motor impairments, however, may take some time to get comfortable with it.

Acknowlegements. I would like to thank the staff and clients of United Cerebral Palsy in Suffolk County Long Island for their gracious hospitality and help in testing SEK, especially on such a tight schedule. I would also like to thank Shari Trewin for her thoughtful comments, suggestions, and hours typing - and watching others type - simple sentences.

References

1. Gump, A., LeGare, M., Hunt, D.L.: Application of Fitts' law to individuals with cerebral palsy. Perceptual and Motor Skills 94(3), 883–895 (2002)
2. Kjeldsen, R.: Improvements in Vision-based Pointer Control. In: Proceedings of ASSETS 2006: 8th International ACM SIGACCESS Conference on Computers and Accessibility, Portland, USA, ACM Press, New York (2006)
3. Mankoff, J., Abowd, G.D.: Cirrin: a word-level unistroke keyboard for pen input. In: Proceedings of the 11th Annual ACM Symposium on User interface Software and Technology UIST '98, pp. 213–214. ACM Press, New York (1998)
4. Trewin, S., Keates, S., Moffatt, K.: Developing Steady Clicks: A Method of Cursor Assistance for People with Motor Impairments. In: Proceedings of ASSETS 2006: 8th International ACM SIGACCESS Conference on Computers and Accessibility, Portland, USA, pp. 26–33. ACM Press, New York (2006)
5. Ward, D.J., Blackwell, A.F., MacKay, D.J.C.: Dasher –A data entry interface using continuous gestures and language models. In: Proc. UIST '00, pp. 129–137. ACM Press, New York (2000)
6. Wobbrock, J.O., Myers, B.A.: Trackball Text Entry for People with Motor Impairments. In: Proceedings CHI'2006: Human Factors in Computing Systems. Montreal, Canada pp. 479–488 (2006)
7. Zhai, S., Hunter, M., Smith, B.A.: Performance Optimization of Virtual Keyboards. Human-Computer Interaction 17(2&3), 229–269 (2002)
8. Zhai, S., Kong, J., Ren, X.: Speed-accuracy trade-off in Fitts' law tasks — On the equivalency of actual and nominal pointing precision. International Journal of Human-Computer Studies 61(6), 823–856 (2004)