Long-Term Study of a Software Keyboard That Places Keys at Positions of Fingers and Their Surroundings

Yuki Kuno, Buntarou Shizuki, and Jiro Tanaka

University of Tsukuba, Japan
{kuno,shizuki,jiro}@iplab.cs.tsukuba.ac.jp

Abstract. In this paper, we present a software keyboard called Leyboard that enables users to type faster. Leyboard makes typing easier by placing keys at the positions of fingers and their surroundings. To this end, Leyboard automatically adjusts its key positions and sizes to users' hands. This design allows users to type faster and more accurately than using ordinary software keyboards, the keys of which are unperceptive. We have implemented a prototype and have performed a long-term user study. The study has proved the usefulness of Leyboard and its pros and cons.

Keywords: Touch screen, text entry, software keyboard, long-term study.

1 Introduction

QWERTY software keyboards are available for text entry on devices with touch screens. Although their key layout is the same as those of physical QWERTY keyboards, we find it difficult to place our fingers on target keys since we cannot obtain any physical feedback.

On the other hand, software keyboard can easily change their key positions and sizes so that they can fit each user. Utilizing this advantage may be able to compensate for the lack of physical feedback and to improve input speed.

In this paper, we describe a software keyboard that can automatically adjust its key positions and sizes to the user's hands. Longitudinal experiments have proven the usefulness of the proposed keyboard.

The novel features of our software keyboard are as follows:

- We place keys at the touch points of all fingers and their surroundings, so that key layout suits the position of all fingers.
- We move all keys for the thumb to prevent the hand from breaking its posture, while users press a key with pressing another key (thumb based sliding).
- We place some keys around the position of thumbs and enable them to input by swiping them with thumbs (thumb swipe input).
- We combine thumb based sliding and thumb swipe input, so that multiple keys can be input without breaking the posture of the users' hand.

M. Kurosu (Ed.): Human-Computer Interaction, Part V, HCII 2013, LNCS 8008, pp. 72–81, 2013.

2 Related Work

LiquidKeyboard [1], CATKey [2], Personalized Input [3], and the study of Gunawardana et al. [4] adopt a similar approach to our research, which adjusts shapes and places of keys to users' hands. These keyboards can input letters but not some keys available on ordinary keyboards, including numbers and some symbols. In contrast, we use the combination of thumb based sliding and thumb swipe input to input such keys.

The study of McAdam et al. [5,6] and SLAP widgets [7] take an approach to provide users tactile feedback, which is different from ours. In contrast, we reduce the difficulty of input arising from the lack of feedback by fitting the place of keys to each user.

Gestyboard [8] and Bimanual Gesture Keyboard [9] use gesture input in text entry. In our method, gesture input is used only as a modifier, not for text entry itself.

3 Leyboard

We designed a prototype of the software keyboard that places keys at positions of fingers and their surroundings [10]. In this section, we describe the essential features, way to input, and design of this prototype. We named this prototype "Leyboard". We named it Leyboard by replacing 'K' with 'L', since we expect this will be a more advanced than an ordinary Keyboard.

3.1 Key Placing on Positions of Fingers and Their Surroundings

The keys of Leyboard are placed to be based on the touch points of users' fingers. Each layout of keys is determined from Voronoi diagrams. Leyboard places keys on the basis of each finger's position and QWERTY layout. A, S, D, F, Space, Enter, J, K, L, and semi-colon keys are placed at each finger's position. Hereafter, we call these keys home position keys and the rest non-home position keys.

Calibration. In this study, calibration is the name of the procedure for determining the layout of Leyboard. Leyboard checks the correspondence of touch points and users' fingers, when users place all their fingers (i.e., ten fingers) on the touch screen. Then, Leyboard calculates gradients of hands from position of the index and little finger of each hand.

Leyboard places home position keys at each finger's position, and places non-home position keys around them. Positions of non-home position keys are concyclic, where they are rotated in accordance with the gradients of hands, with the home position key at the center. We considered that keys can be input easily as making their distance from home position key the same as that of other keys. On the other hand, those distances are different on ordinary software keyboards.

The calibration is completed when users release their fingers from the touch screen. Now the keys of Leyboard are placed at the positions of users' fingers

and their surroundings. Thus the key layout suits the position of users' fingers, which enables users to input the keys they intend to input easily.

Determining Key Area. We make Leyboard to input the key the coordinates of which are the closest to the users' touch point. This involves making the area of the keys as large as possible to enable users to press keys easily. We have made Leyboard determine the area of keys by Voronoi diagrams. We used Fortune's algorithm [11] to draw lines of Voronoi diagrams.

Placing Keys Around Thumbs. On physical keyboards, thumbs are used to press the Space key. On Leyboard, not only Space but also many keys are placed around the position of thumbs. Keys that change the key set, which we describe below, are also included in those keys. Therefore, the key layout of Leyboard is strictly different from the QWERTY layout. The point is that all keys are placed at the positions of users' fingers or their surroundings in this design. Therefore, users' do not need to move their hands as widely as they do with ordinary QWERTY layout keyboards. We consider this enable users to keep their fingers at the position of home position keys while they input, thus reducing error inputs.

We made Leyboard able to provide three key sets (Fig. 1 to Fig. 3). With Leyboard, users change the key sets as necessary, while they type. We placed keys to change the key set around thumbs as described earlier. Key sets were provided because we cannot put all necessary keys for text entry in one state (i.e. one key set) in the design of Leyboard, where all keys are at the position of users' fingers or their surroundings. While users press keys to change key sets, which is shown by circle in Fig. 2 and Fig. 3, the key set changes into the one shown in these figures. When users released their finger from the key for changing key sets, the key set returns to the alphabet set (Fig. 1). As a result, Leyboard is able to input 102 keys for total in these three key sets.

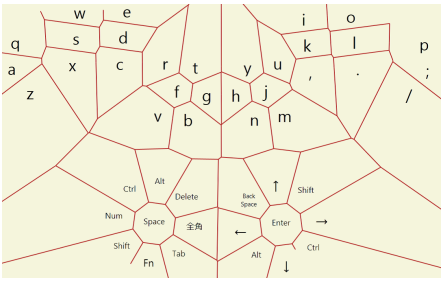

Fig. 1. Layout of alphabet set

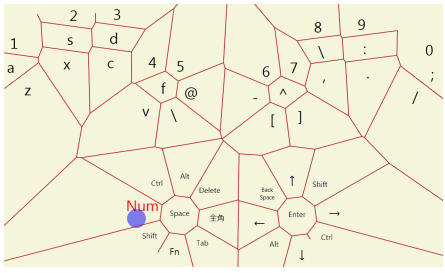

Fig. 2. Layout of numbers and symbols set

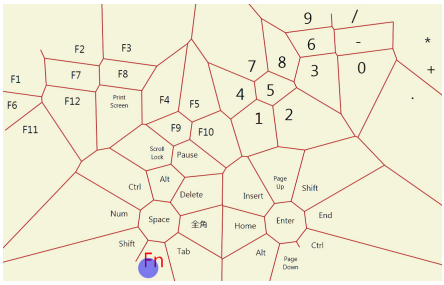

Fig. 3. Layout of functions and numerical keypad set

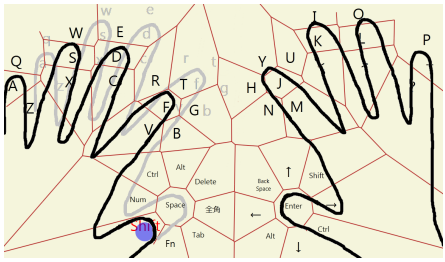

Fig. 4. Example of thumb based sliding

3.2 Thumb Based Sliding

We developed a technique called "thumb based sliding". Two keys must be pressed to input some characters (e.g., 'f' and Shift to input capital 'F'). The thumb based sliding technique makes such simultaneous input easy with our key layout. Fig. 4 shows an example of thumb based sliding. Assume that the left thumb presses Shift. Then Leyboard moves all keys for the left hand like Fig. 4. This design allows users to press keys while pressing another key without breaking the posture of the hand, which enables users to input smoothly and therefore, quickly.

3.3 Thumb Swipe Input

Keys at thumbs and their surroundings are able to be input with "thumb swipe input". Users can input these keys by swiping their fingers from one key to the next. Note that keys to change the key set are only functional while users press these keys. Thumb swipe input is used when users need to input modifier keys such as Shift while the key set is changed.

3.4 Combination of Thumb Based Sliding and Thumb Swipe Input

Leyboard enables users to input many types of keys without breaking the postures of their hands by combining thumb based sliding and thumb swipe input. Fig. 5 shows an example of the combination of thumb based sliding and thumb swipe input. Note the rectangle at the upper right of the left index finger. The key displayed in the rectangle changes while the user swipes his or her left thumb. Here, the keys are constantly at the user's fingers or their surroundings.

3.5 Sound Feedback

We give users sound feedback when they input a key or change the key set. Leyboard makes clicking sound to notify users that events described above has occurred.

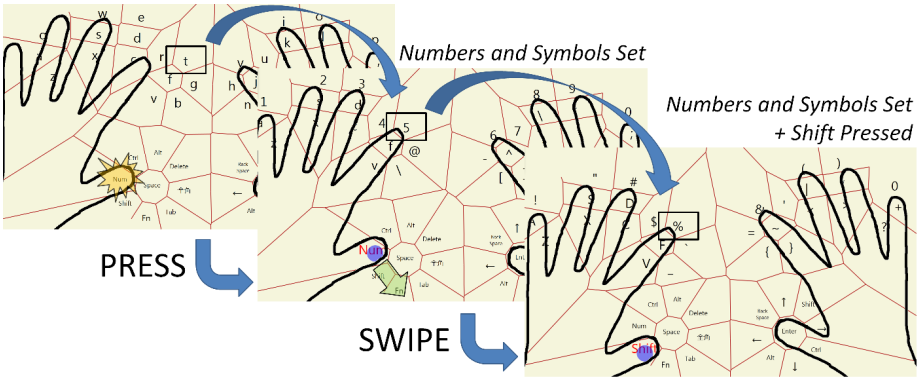

Fig. 5. Example of combination of thumb based sliding and thumb swipe input

4 Developing Environment

We chose C# as a programming language for developing Leyboard. We implemented Leyboard as a WPF application, which supports multi touch by using API of .NET Framework 4/WPF4.

5 Long-Term User Study

We conducted a long-term user study to compare an ordinary software keyboard and Leyboard. We chose the software keyboard regularly installed in Windows 7. Hereafter, we call this software keyboard the Windows 7 keyboard. The study lasted about one year, which is a considerable term.

5.1 Environment

We used Acer's ICONIA-F54E to operate Leyboard in the study. Fig. 6 shows our experiment environment where we used ICONIA-F54E. ICONIA-F54E has a 14-inch touch screen, 1366×768 pixels resolution (WXGA), and can detect up to 10 touch points.

5.2 Participant and Tasks

The participant was one of the authors. We chose tasks of inputting English pangrams. The pangrams contained capital letters and some also included symbols. Characters in one pangram range from 31 to 63. Hereafter, we call inputting 10 different pangrams a set. The participant had three sets of tasks for each software keyboard on each day. We conducted this evaluation from February 15, 2012 to February 14, 2013, for 364 days (cutting 2 days on which no task occurred). This is equivalent of 1092 sets. For the first seven days, the participant performed tasks first with the Windows 7 keyboard and then Leyboard. These orders were reversed in a seven-day cycle.

Fig. 6. Experiment environment

5.3 Results

We calculated an input rate in words per minute (wpm), since completion time itself is not a metric of performance of software keyboards. This is because pangrams inputted by the participant differed in every sets. Wpm is a unit of presenting words inputted in a minute that is defined by Gentner [12]. It is calculated as follows:

$$\frac{1}{5}\frac{\text{Incoming keystrokes except mistakes (times/set)}}{\text{Input time (minutes/set)}}$$

Fig. 7 shows the input rate in wpm on each software keyboard, with fitted curves, which are approximated to logarithmic curves.

Although the difference is not yet large, Leyboard still outperforms ordinary software keyboards according to the fitted curve. The maximum input rate of each keyboard was 54.8 wpm for the Windows 7 keyboard and 56.4 wpm for Leyboard. The average input rate of each keyboard was 39.7 for the Windows 7 keyboard and 41.6 for Leyboard. Leyboard has a higher input rate than the Windows 7 keyboard in both values.

Fig. 8 shows the error rate on each software keyboard. The average error rate of each keyboard was 6.07 % for the Windows 7 keyboard and 6.40 % for Leyboard. The value of Leyboard is slightly higher than that of the Windows 7 keyboard.

When a long-term study is conducted, values of evaluation follow the power law. The power approximation curves of values become almost straight lines in a double logarithmic chart. Fig. 9 shows the double logarithmic chart of time for inputting 100 characters on average. As the power approximation curves of Fig. 9 are nearly straight lines, values in the evaluation are considered reasonable and proper.

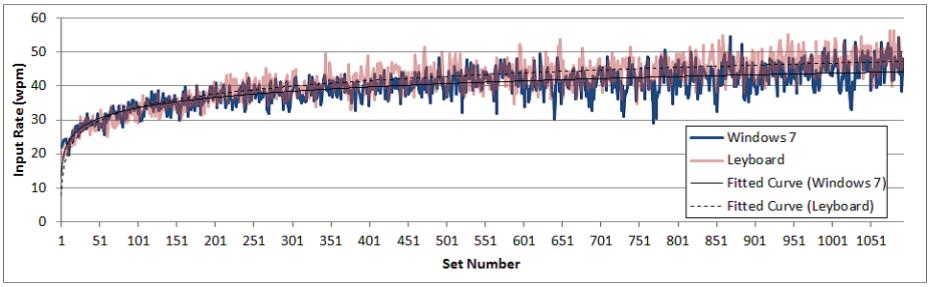

Fig. 7. Input rate (wpm) on each software keyboard

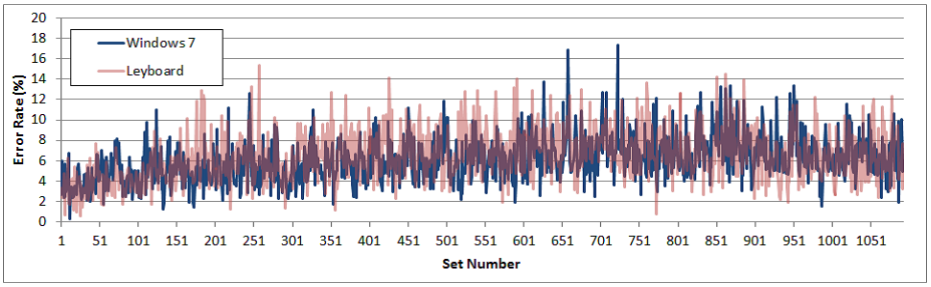

Fig. 8. Error rate on each software keyboard

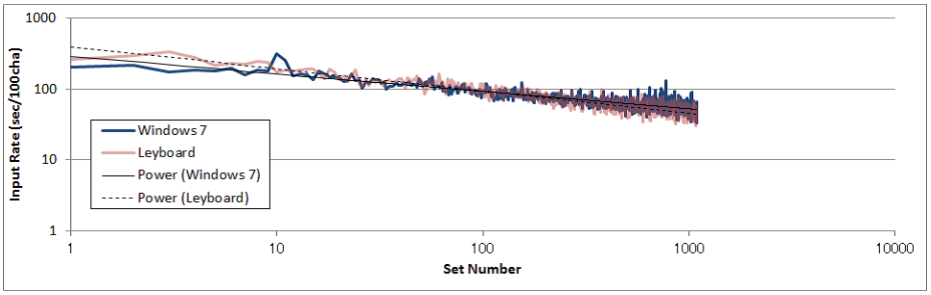

Fig. 9. Input rate (seconds per 100 characters) on each software keyboard

6 Discussion

We have conducted paired t-test to verify that wpm and error rate averages of both keyboards significantly differ. As a result, the input rate of Leyboard is significantly higher than that of the Windows 7 keyboard ($t = -14.6591, p = 2.2e-16<0.01$); but its error rate is also significantly high ($t = -3.708, p = 0.0002<0.01$).

Furthermore, we have analyzed the content of the errors. We collected logs of all key inputs during the study. Therefore, we referred to the log and determined every mistaken or omitted input using the following algorithm:

1. Errors are determined in prefix search.
2. The algorithm compares the text and input by characters. The correct input will be skipped.
3. If the character and the input were different, focus on the next character. If it matches the input, the input is considered an omission. Otherwise, it is considered a mistake.
4. If there are any errors, ignore every input from the next onwards until the input come to the correct. This is to avoid the slippage of input that arises from the omission.

For example, if the text was "puppy" and the input was "pupy", the algorithm would find the omission of 'p'. For another example, if the text was "lazy" and the input was "kazy", the algorithm would find the mistake of 'l'. Note that the participant cannot go farther on the tasks as long as he or she inputs wrong characters and eventually has to input the correct character. As a result, the actual inputs with an error would have inserted certain character strings compared with the text. Thus, the actual input of the first example becomes "pupypy" (pup(y)py), and the second becomes "klazy" ((k)lazy).

There were 17,215 errors on the Windows 7 keyboard and 17,490 on Leyboard. The Windows 7 keyboard had 12,878 mistakes and 4,337 omissions. Leyboard had 11,148 mistakes and 6,342 omissions. Even though Leyboard had more errors than the Windows 7 keyboard, Leyboard seems to have had fewer mistakes and more omissions. To describe the tendency of errors on both keyboards, we draw graphs of error frequencies. Fig. 10 shows the mistakes, and Fig. 11 shows the omissions. Here we cut the characters that have errors below a certain number (100 for mistakes and 50 for omissions) to make these graphs conspicuous.

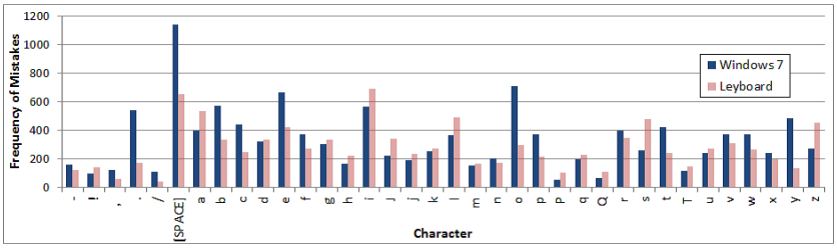

Fig. 10. Breakdown of mistakes

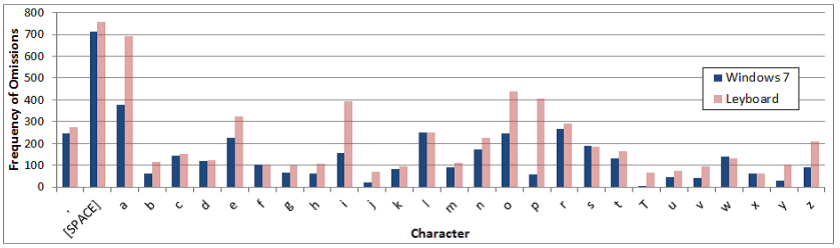

Fig. 11. Breakdown of omissions

In mistakes, the Windows 7 keyboard mistook period, space, 'o' and 'y' more frequently than Leyboard. Period was mostly mistaken as comma (316 times), Space as 'n' (605), 'o' as 'p' (368) and 'y' as 'u' (276). They are all keys placed horizontally except Space and 'n'. On the other hand, Leyboard mistook 's' and 'z' more frequently than the Windows 7 keyboard. 's' was mostly mistaken as 'x' (246) and 'z' as 'a' (221). They are both keys placed vertically.

In omissions, the Windows 7 keyboard did not exceed Leyboard for any specific key. On the other hand, Leyboard omitted 'a', 'i', 'o' and 'p' more frequently than the Windows 7 keyboard. Especially omitted were 'a' in "and" (37), 'i' in "quiz" (151), 'o' in "of" (40), and 'p' in "nymph" (66). It seems that Leyboard is poor at inputting keys with little fingers, such as 'a' and 'p'. Actually, little fingers were hardly used on the Windows 7 keyboard in the study; annular fingers were used instead. This is because keys for little fingers on the QWERTY layout are hard to reach for little fingers on a software keyboard. This is not a specific case because at the time we conducted the user study on the early version of Leyboard, there were users who did not use little fingers while inputting on the Windows 7 keyboard for the same reason. It is also difficulty to input adjacent keys in the top row continuously. This seems to be because we placed non-home position keys con-cyclically, which strains the key layout, especially of the top row.

7 Conclusion and Future Work

In this paper, we have presented Leyboard, a software keyboard that enables faster typing than ordinary software keyboards. Leyboard places home position keys of the QWERTY layout at the touch point of each finger and non-home position keys at their surroundings. Leyboard enables many keys to be pressed with a small amount of hand movement by combining thumb based sliding and thumb swipe input.

A long-term user study found that Leyboard exceeds input rate of the regularly installed Windows 7 software keyboard. However, its error rate is also high. This is because the current version of Leyboard tends to make users input wrong keys placed vertically, is poor at inputting with little fingers, and has difficulty inputting adjacent keys in the top row continuously. Therefore, our future

work is to make Leyboard input these for sure. Also, we are considering having user studies with more participants and comparing Leyboard with physical keyboards, including ergonomic products.

References

1. Sax, C., Lau, H., Lawrence, E.: LiquidKeyboard: An ergonomic, adaptive QWERTY keyboard for touchscreens and surfaces. In: Proceedings of the Fifth International Conference on Digital Society, ICDS 2011, XPS, pp. 117–122 (2011)
2. Go, K., Endo, Y.: CATKey: Customizable and adaptable touchscreen keyboard with bubble cursor-like visual feedback. In: Baranauskas, C., Abascal, J., Barbosa, S.D.J. (eds.) INTERACT 2007. LNCS, vol. 4662, pp. 493–496. Springer, Heidelberg (2007)
3. Findlater, L., Wobbrock, J.: Personalized input: improving ten-finger touchscreen typing through automatic adaptation. In: Proceedings of the 2012 ACM Annual Conference on Human Factors in Computing Systems, CHI 2012, pp. 815–824 (2012)
4. Gunawardana, A., Paek, T., Meek, C.: Usability guided key-target resizing for soft keyboards. In: Proceedings of the 15th International Conference on Intelligent User Interfaces, IUI 2010, pp. 111–118. ACM, New York (2010)
5. McAdam, C., Brewster, S.: Distal tactile feedback for text entry on tabletop computers. In: Proceedings of the 23rd British HCI Group Annual Conference on People and Computers, BCS-HCI 2009, pp. 504–511 (2009)
6. McAdam, C., Brewster, S.: Mobile phones as a tactile display for tabletop typing. In: Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces, ITS 2011, pp. 276–277 (2011)
7. Weiss, M., Wagner, J., Jansen, Y., Jennings, R., Khoshabeh, R., Hollan, J.D., Borchers, J.: Slap widgets: bridging the gap between virtual and physical controls on tabletops. In: Proceedings of the 27th International Conference on Human Factors in Computing Systems, CHI 2009, pp. 481–490 (2009)
8. Coskun, T., Artinger, E., Pirrilli, L., Korhammer, D., Benzina, A., Grill, C., Dippon, A., Klinker, G.: Gestyboard: A 10-finger-system and gesture based text input system for multi-touchscreens with no need for tactile feedback. In: Proceedings of the 10th Asia-Pacific Conference on Computer-Human Interaction, APCHI 2012, pp. 701–702 (2012)
9. Bi, X., Chelba, C., Ouyang, T., Partridge, K., Zhai, S.: Bimanual gesture keyboard. In: Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology, UIST 2012, pp. 137–146 (2012)
10. Kuno, Y., Shizuki, B., Tanaka, J.: Leyboard: A software keyboard that places keys at positions of fingers and their surroundings. In: Proceedings of the 10th Asia-Pacific Conference on Computer-Human Interaction, APCHI 2012, pp. 723–724 (2012)
11. Fortune, S.: A sweepline algorithm for voronoi diagrams. In: Proceedings of the Second Annual Symposium on Computational Geometry, SCG 1986, pp. 313–322. ACM, New York (1986)
12. Gentner, D.R.: Keystroke timing in transcription typing. In: Cooper, W.E. (ed.) Cognitive Aspects of Skilled Typewriting, pp. 95–120. Springer (1983)