A Comparison of Soft Keyboards on Circular Smartwatches

Reinhard Pointner
National Taiwan University
r02922148@ntu.edu.tw

Mike Chen
National Taiwan University
mikechen@csie.ntu.edu.tw

Abstract

Smartwatches, wearable electronics and other miniature devices are becoming increasingly popular. However, text input on such small devices remains a challenge. We implement and compare 3 fully functional soft keyboards with word prediction and basic error correction on the latest generation of circular smartwatch devices. We introduce KeyDial, a novel circular QWERTY keyboard, improve upon SwipeKey [9] and compare these two novel input methods against an implementation of the standard QWERTY keyboard optimized for a tiny screen. Our evaluation is based on a user study with 12 participants and shows input speeds of around 12 WPM for KeyDial, 10 WPM for SwipeKey and 16 WPM for a tiny QWERTY keyboard after 20 minutes of training. While many participants ultimately preferred the familiar QWERTY keyboard in our experiment, unfamiliar input methods such as KeyDial and SwipeKey did show merit in qualities such as character-by-character accuracy.

Author Keywords

text entry; smartwatch; soft keyboard; circular keyboard

ACM Classification Keywords

H.5.2 [User Interfaces (D.2.2, H.1.2, I.3.6)]: Input devices and strategies (e.g., mouse, touchscreen)

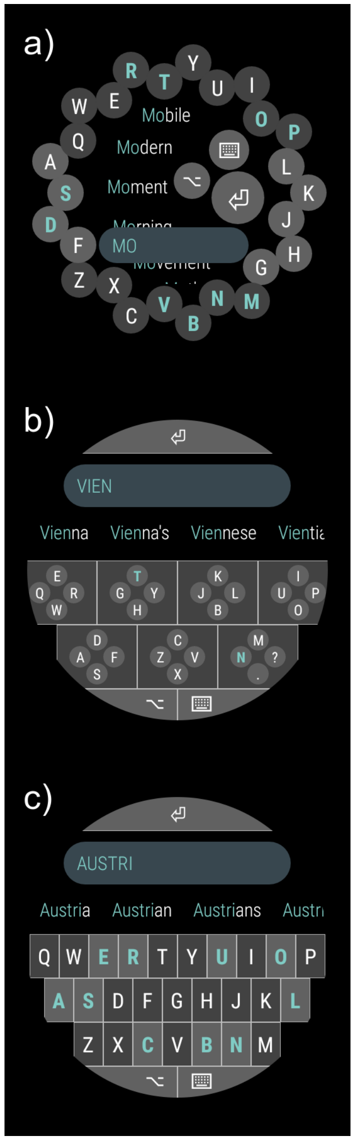

Figure 1: Keyboards for circular smartwatches used in our study: (a) KeyDial, (b) SwipeKey and (c) Standard QWERTY.

Introduction

As miniaturization and battery technology improve, major electronics manufacturers have created smartwatches, which contain the processing power and memory of a computer or a smartphone and feature 4G connectivity, GPS, accelerometers, heart beat sensors and so on. These smartwatches can provide interactions with the digital world without one having to carry a phone, for example when doing outdoor sports.

Even though smartwatch devices have become comparable to smartphones in terms of computing power, user-interaction can be tricky due to the limited screen size. Voice input, while reliable and fast in most cases, is not always socially acceptable and may not be a viable option in noisy environments or when entering out-of-vocabulary text such as addresses or passwords. As such, easy-to-use and accurate text input methods are an important aspect of smartwatch devices, and platform developers such as Google now include touch-based text input methods by default and allow users to add 3rd party input methods.

In this paper, we design and implement 3 fully functional soft keyboards for circular smartwatches. We propose a circular keyboard layout where 26 keys are aligned on the circumference of the circular screen, a variation of SwipeKey with 7 keys and 4 swipe directions each and a standard QWERTY keyboard with 26 keys optimized for the small screen size. All 3 keyboard implementations use the same word prediction and error correction algorithms to allow for a fair comparison in our user study.

Related Work

Text entry on small touch screen devices is a well explored topic. To overcome the limitations of a small screen, ZoomBoard [8] iteratively zooms in on a QWERTY keyboard at

Figure 2: The BACKSPACE and SPACE keys are mapped to the text input field.

the touch point to increase the button size until buttons can be pressed reliably and uses simple swipe gestures for common keys like space and delete. Swipeboard [1] uses swipes in different directions to iteratively select from a two-level hierarchical set of keys.

Other methods have been proposed specifically for wristwatch-sized touch screens. SplitBoard [4] uses a QWERTY keyboard that is larger than the screen and uses swipe gestures to move the desired keys into view. Dunlop *et al.* [2] proposed a system that uses an ambiguous keyboard, mapping multiple keys to a single button, and then using a predictive model to select the most likely option. SwipeKey [9] uses swipe direction so that a single button can unambiguously represent multiple characters and reports faster text entry speeds than previous works at high accuracy. Luis *et al.* [5] explore the feasibility of a tiny QWERTY keyboard on small screens of various sizes.

More recent techniques such as VelociTap [11] and WatchWriter [3] focus on using statistical decoding of words, phrases, and even sentences to guess the correct input based on noisy touchscreen typing data. Notably, WatchWriter is now available in the latest version of Android Wear as default text input method.

Three Circular Soft Keyboards

We have designed and implemented 3 keyboards for circular smartwatches. Figure 1 shows our three prototypes. Each prototype has been implemented natively for Android

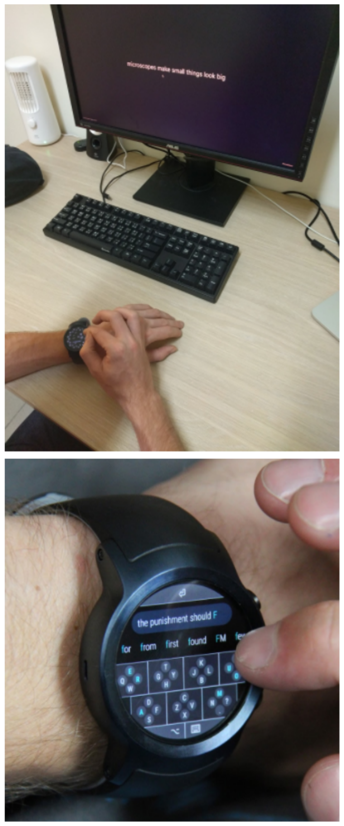
Figure 3: The setup of our user study.

Wear 2.0 and tested on an LG Watch Sport. This smartwatch has a round 1.38" (35 mm) screen. Our implementation is open-source software, so that other researchers or developers can reuse and build upon our code.[1]

KeyDial
The KeyDial keyboard arranges 26 keys along the circumference of the circular screen (Figure 1a). We leverage familiarity of the QWERTY layout by arranging keys in similar spacial locations as far as possible. Keys are aligned in a zigzag pattern to introduce additional padding between each key to make it easier to accurately press individual keys. Each keyboard button has a diameter of about 4 mm.

SwipeKey
We implement SwipeKey [9] with 7 buttons and 4 swipe directions for each button for a total of 28 keys (Figure 1b). Similar to KeyDial, keys are arranged and clustered in a way that mimics the QWERTY layout as far as possible. Each swipe key button has a size of 7 mm $\times$ 9 mm.

Standard QWERTY
Our implementation of the standard QWERTY keyboard is designed to fit into the lower half of the circular screen and uses the available screen space as best as possible (Figure 1c). The keyboard button width is primarily limited by the number of keys in the top row which allows for a button size of 3 mm $\times$ 5 mm in our prototype.

Common Features
Each keyboard has a text field that shows the text that has been entered so far. Figure 2 shows how the BACKSPACE and SPACE keys are mapped to the left-hand side and the right-hand side of the text input field. Our implementation does not support a cursor or cursor movements so pressing BACKSPACE will always delete the last character. We use OpenAdaptxt [7] to provide word prediction and error correction. Keys that are likely to be pressed next are highlighted to aid users in finding keys on unfamiliar keyboard layouts. The rotating hardware button can be used to scroll through the list of word suggestions without occluding the screen. When selecting a suggestion, the current word at the end of the current input will be replace by the suggestion and a SPACE will be added automatically. Each keyboard has 2 modifier keys which allow users to toggle between 4 different keyboard modes (Figure 4).

[1] https://github.com/rednoah/android-wear-keydial

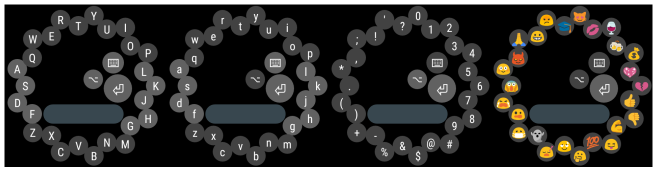
Figure 4: Left to right. Keyboard modes: upper-case letters, lower-case letters, numbers and punctuation, Emoji.

Evaluation
The aim of this study is to understand the characteristics and user preferences for each of our three keyboard prototypes. We conducted a user study consisting primarily of text-copy tasks using sentences randomly selected from the MacKenzie and Soukoreff phrase set [6] and an additional smaller task with randomly generated passwords to evaluate performance on difficult-to-predict out-of-vocabulary input.

Participants
We recruited 12 participants between the ages of 23 to 31 (M = 27.4, SD = 2.5) from various departments at our university. 7 are native or highly proficient English speakers. All of them were right-handed and 11 chose to wear the

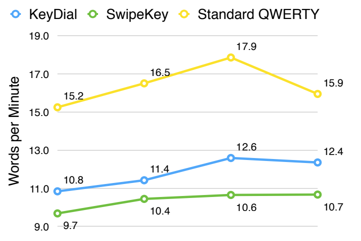

Figure 5: WPM for each keyboard in each block of 5 sentences.

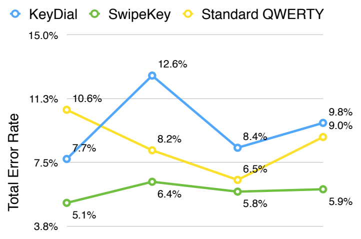

Figure 6: TER for each keyboard in each block of 5 sentences.

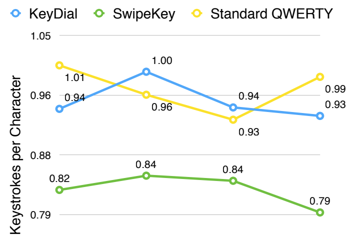

Figure 7: KSPC for each keyboard in each block of 5 sentences.

smartwatch on their left wrist and typed with their right index finger. 2 had used a smartwatch before. All participants were very familiar with the QWERTY keyboard layout. Each participant was paid the equivalent of USD 10 in compensation for their time.

Procedure
Participants were seated in a quiet environment and were asked to rest their arm on the table in front of them to avoid fatigue. A computer screen in front of the participant instructed the participant which keyboard to use and what sentences to type. Figure 3 shows our user study setup. Participants were given a chance to experiment with each keyboard and had to complete a training session which included every letter of the alphabet before starting the trials.

For each of the three keyboards, participants had to enter 20 sentences and 5 passwords. The total of $3 \times 20 = 60$ sentences for all keyboards was randomly sampled from the MacKenzie and Soukoreff phrase set [6]. These sentences contain neither numbers nor punctuation symbols. All participants entered the same list of 60 sentences in the same order but used different keyboards for each block of 20 sentences. We used Latin square design to counterbalance the order of keyboards. The total of $3 \times 5 = 15$ passwords was randomly generated and included upper-case and lower-case letters as well as numbers and symbols. In sum, each participant completed 3×20 sentences and 3×5 passwords for a total of 75 trials.

Participants were asked to type as fast and as accurately as possible and not to worry about upper-case and lower-case letters (except when entering passwords). Before typing the first letter of each sentence, participants were allowed to rest and memorize the sentence displayed on the computer screen in front of them for as long as they wanted. After entering each sentence, participants would press the ENTER key to continue to the next sentence.

After completing all trials, users were asked to rate each keyboard on a 5 point Likert scale and comment on positive and negative aspects of each keyboard.

Results
During the trial, some participants accidentally tapped the ENTER button which allowed them to submit incomplete input, resulting in high error rates due to missing characters, so for our evaluation we only consider sentences that are at least 50% complete. This gives us 717 samples of sentence input representing 6.1 hours of typing data and 176 samples of password input representing 1.4 hours of typing data.

Text Entry Performance
We calculate the text entry speed in words per minute (WPM) as described by Soukoreff *et al.* [10]. 5 characters are equal to one word. Time is measured starting at the first key press so the first character typed does not count towards the total number of characters typed. Figure 5 shows the average WPM for each block of 5 sentences over time because WPM per individual sentence can differ significantly due to typing errors and corrections. The standard QWERTY keyboard consistently outperforms other input methods in terms of speed on both sentence input and password input (Table 1), with an average WPM of 16.4 (*SD* = 4.9) with our fastest participant reaching 19.5 WPM. In comparison, KeyDial and SwipeKey only allowed for average input speeds of 11.8 and 10.4 WPM respectively. The unfamiliar nature of KeyDial and SwipeKey made it difficult for participants to type quickly. Our fastest participant was able to achieve 14.5 WPM with both KeyDial and SwipeKey.

Error Rate
We calculate the total error rate (TER) as described by Soukoreff *et al.* [10] which accounts for corrected and uncorrected errors. In our experiment, errors were not only caused by pressing the wrong key, but also by unintentionally pressing a word prediction resulting in multiple character errors at once. Figure 6 shows the average TER for each 5 sentence block. As expected, SwipeKey has the lowest TER because it has the largest buttons and accidentally swiping in the wrong direction was not an issue. Surprisingly, KeyDial performed worse than the QWERTY keyboard in this metric despite the keys being more spaced out. It turned out that many participants accidentally pressed a word prediction when trying to press a character on the left-hand side of the circular keyboard. As shown in Table 1 (II), the standard QWERTY keyboard was able to achieve high levels of accuracy once users were forced to type slowly and carefully during our password input tasks.

Keystrokes per Character
Typically, a single keystroke represents one character resulting in a keystrokes per character (KSPC) rate of 1.0. Function keys such as DELETE or SHIFT increase the KSPC, while the use of word prediction allows users to enter multiple characters in one tap and thus decrease the KSPC. Figure 7 shows the average KSPC for each 5 sentence block. SwipeKey has a low error rate and the learning curve lead many participants to rely on word prediction more heavily, resulting in a low KSPC. For KeyDial and the standard QWERTY keyboard, the keystrokes saved by word prediction roughly balance out the extra keystrokes caused by correcting errors. Table 1 (II) shows unusually high KSPC when entering passwords caused by shifting between different keyboard modes.

Table 1: Mean WPM, TER, and KSPC for sentence input and password input for each keyboard. SDs are in parenthesis.

	KeyDial	SwipeKey	Standard QWERTY
(I) Sentence Input			
WPM	11.80 (3.28)	10.36 (2.95)	16.39 (4.93)
TER	9.62 (12.32)	5.80 (8.76)	8.56 (8.90)
KSPC	0.95 (0.29)	0.82 (0.23)	0.97 (0.28)
(II) Password Input			
WPM	3.69 (1.00)	3.53 (0.92)	4.66 (1.17)
TER	7.21 (14.12)	7.81 (13.48)	5.67 (7.60)
KSPC	2.21 (0.55)	2.29 (0.70)	2.31 (0.57)
(III) User Preference			
Speed	3.2 (0.8)	2.3 (0.6)	4.6 (0.5)
Accuracy	2.8 (1.1)	4.5 (0.7)	3.6 (1.0)
Overall Rating	3.0 (0.9)	3.1 (0.9)	3.9 (0.7)

User Preference
Based on questionnaires and interviews, we found that participants strongly favoured the standard QWERTY keyboard due to the familiarity of the keyboard. Table 1 (III) indicates that the fat finger problem was not a major issue for most of our participants. However, the unfamiliar nature of KeyDial and SwipeKey made it difficult to find letters and type quickly. Nevertheless, participants did appreciate the high level of accuracy of SwipeKey despite the additional cognitive load of searching for letters and swiping correctly.

Limitation and Future Work
There are several limitations in both implementation and evaluation of our keyboard prototypes.

Firstly, we used a single device with a 1.38" screen for the entire study. The results may vary on smaller devices. More research is needed on how these techniques fare on different smartwatches of various form factors.

Secondly, each participant was only able to use each keyboard for about 20 minutes which puts unfamiliar input methods such as SwipeKey at a distinct disadvantage. In addition, some of our participants were non-native speakers and it turns out that they had trouble memorizing and transcribing each sentence correctly due to the increased cognitive load of using an unfamiliar keyboard.

Thirdly, there are many unexplored ways of improving our prototypes to arrive at better results. KeyDial may greatly benefit from statistical decoding of touch events similar to WatchWriter [3] allowing for smaller keys and giving more screen space to word prediction. SwipeKey may benefit from allowing ambiguous key taps in addition to unambiguous key swipes. This would support faster input of words and phrases with 7 ambiguous keys as well slower character-by-character input with the 28 unambiguous swipe keys in a single design.

Conclusion

In this paper, we compare three keyboard layouts for text entry on circular smartwatches. We implement a standard QWERTY keyboard and compare it to two keyboard layouts specifically designed for smartwatches, one that uses a circular arrangement of keys, and another that uses swipe gestures to map multiple keys to a single larger button.

The results of our user study show that the standard QWERTY keyboard outperforms other techniques even on a smartwatch screen in terms of text input speed at around 16 WPM and user preference due to the familiarity of they keyboard layout. However, users did achieve reasonable text speeds of 10-12 WPM with KeyDial and SwipeKey and did appreciate other characteristics such as novelty and accuracy.

REFERENCES

1. Xiang 'Anthony' Chen, Tovi Grossman, and George Fitzmaurice. 2014. Swipeboard: A Text Entry Technique for Ultra-small Interfaces That Supports Novice to Expert Transitions. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology (UIST '14)*. ACM, New York, NY, USA, 615–620.
2. Mark D. Dunlop, Andreas Komninos, and Naveen Durga. 2014. Towards High Quality Text Entry on Smartwatches. In *CHI '14 Extended Abstracts on Human Factors in Computing Systems (CHI EA '14)*. ACM, New York, NY, USA, 2365–2370.
3. Mitchell Gordon, Tom Ouyang, and Shumin Zhai. 2016. WatchWriter: Tap and Gesture Typing on a Smartwatch Miniature Keyboard with Statistical Decoding. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 3817–3821.
4. Jonggi Hong, Seongkook Heo, Poika Isokoski, and Geehyuk Lee. 2015. SplitBoard: A Simple Split Soft Keyboard for Wristwatch-sized Touch Screens. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. 1233–1236.
5. Luis A. Leiva, Alireza Sahami, Alejandro Catala, Niels Henze, and Albrecht Schmidt. 2015. Text Entry on Tiny QWERTY Soft Keyboards. In *Proc. of CHI '15*. ACM, New York, NY, USA, 669–678.
6. I. Scott MacKenzie and R. William Soukoreff. 2003. Phrase Sets for Evaluating Text Entry Techniques. In *CHI '03 Extended Abstracts on Human Factors in Computing Systems (CHI EA '03)*. ACM, New York, NY, USA, 754–755.

7. Dona P. Durga N. Montaparti, S. and R. D. Meo. 2012. OpenAdaptxt: An Open Source Enabling Technology for High Quality Text Entry.. In *In Proc. CHI Workshop on Designing Evaluating Text Entry (CHI EA '03)*. ACM, New York, NY, USA, 754–755.

8. Stephen Oney, Chris Harrison, Amy Ogan, and Jason Wiese. 2013. ZoomBoard: A Diminutive Qwerty Soft Keyboard Using Iterative Zooming for Ultra-small Devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 2799–2802.

9. Yuan-Fu Shao, Masatoshi Chang-Ogimoto, Reinhard Pointner, Yu-Chih Lin, Chen-Ting Wu, and Mike Chen. 2016. SwipeKey: A Swipe-based Keyboard Design for Smartwatches. In *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI '16)*. ACM, New York, NY, USA, 60–71.

10. R. William Soukoreff and I. Scott MacKenzie. 2003. Metrics for Text Entry Research: An Evaluation of MSD and KSPC, and a New Unified Error Metric. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '03)*. ACM, New York, NY, USA, 113–120.

11. Keith Vertanen, Haythem Memmi, Justin Emge, Shyam Reyal, and Per Ola Kristensson. 2015. VelociTap: Investigating Fast Mobile Text Entry Using Sentence-Based Decoding of Touchscreen Keyboard Input. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. ACM, New York, NY, USA, 659–668.