GetSetPress: A Tool for Analyzing and Designing Activation Methods for Touch and Physical Keyboards

Auejin Ham
HCI Lab, KAIST
Daejeon, Republic of Korea
hamaj@kaist.ac.kr

ABSTRACT
While numerous research studies have proposed that different activation methods can impact the typing performance of touch and physical keyboards, few options have been suggested to simulate various key activation applicable to both touch and physical keyboards. Therefore, we developed a tool to design and simulate different activation methods for touch and physical keyboards. We successfully implemented activation points of touch, release, peak force, peak area, and even a programmable travel depth threshold of given mechanical switches which can be utilized in both touch and physical keyboards. Our system showed decent amount of polling rates to correctly record input streams of typing experts, and end-to-end latency to simulate off-the-shelf desktop environments. We expect this tool to be used for designing optimal key activation methods for different needs.

Author Keywords
Keyboard; Performance; Interaction devices

INTRODUCTION
From the birth of physical keyboards, numerous researchers [11, 5, 8, 14, 7, 9, 16] have been tried to apply touch modalities for activating key press events. While doing so, researchers have been compared typing performance of touch and physical keyboards over three decades. A novel work by Gould et al. [3] in 1990 showed that typing speed of novice users was 20 WPM (word per minute) in physical keyboard, while those of touch keyboards was 12 WPM. This lacking performance of touch keyboards have been consistently reported by following research studies [13, 4].

Researchers have investigated the cause of this lack of performance. One work proposed that typing performance can hugely be affected by input modalities [1], while other works tried to increase the performance by designing key activation regarding a temporal threshold [8], sensory synchronization [7], and implicit touch pressure [16]. Considering a button pressing tasks as as an open-loop problem of neuromechanical control based on the cue integration theory [10], we can assume that the neuromechanical control towards each key can affect the order of touching and pressing keys resulting order of input character misplaced.

To further investigate different key activation methods and the resulted neuromechanical control while typing with touch and physical keyboards, it is critical to compare mental model of users while typing different modalities of button but in the same activation conditions. Therefore, we developed a tool to design common key activation functions with different modalities of touch and physical keyboards. We first built a touch keyboard that simulates from given input force to travel displacement of programmable mechanical switches. We also developed a touch detectors that can be applied for each keys of analog keyboard. And finally, we build a software that provides unified timestamp recordings of high-res polling rate and the graphical interface of key activation.

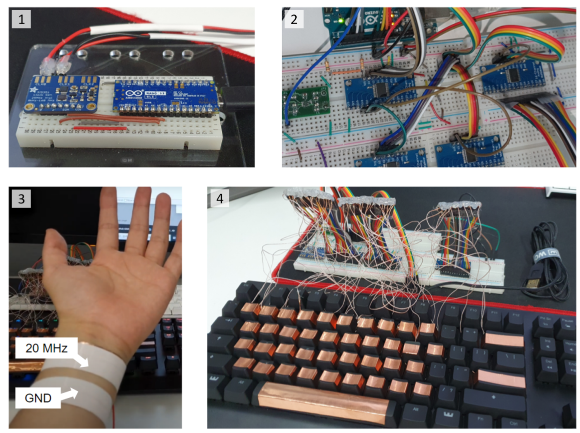

Figure 1: Our touch detecting system: Box 1 shows a clock generator sending 20 MHz square wave to each two hands. Box 2 shows a cirtuit for the multiplexer tree. Box 3 shows a demonstrator wearing a wrist band. Box 4 is the analog keyboard with touch-detecting key caps.

RELATED WORKS

Metrics of Typing Error
Several research studies have proposed metrics to measure typing accuracy (error rates) for transcribing tasks. Gentner et al. [2] proposed a method of categorizing text input errors from presented and transcribed strings. Wobbrock and Myers proposed a character-level classification methods for typing errors [15] using input streams for unconstrained text entry evaluations and it was widely adapted in HCI field. They defined typing errors into three categories (insertion, omission,

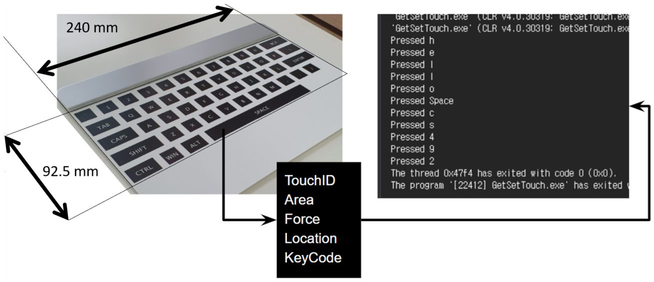

Figure 2: Our touch keyboard and the component of each key for our keyboard event handler.

```
Sensel = GetSensel();                                                    1

// predefined methods of key activation
KeyManager = new KeyActivationManager(Sensel, Activations.TOUCH);
KeyManager = new KeyActivationManager(Sensel, Activations.RELEASE);
KeyManager = new KeyActivationManager(Sensel, Activations.PEAKAREA);
KeyManager = new KeyActivationManager(Sensel, Activations.PEAKFORCE);

// Wooting One keyboards (analog linear switch)
KeyManager = new KeyActivationManager(Sensel,
        Activations.FD(FDs.FlaretechLinear55(), 3.6f));

// Wooting Two keyboards (analog linear switch)
KeyManager = new KeyActivationManager(Sensel,
        Activations.FD(FDs.LekkerLinear60(), 3.8f));

// you can customize parameters of your linear switch
KeyManager = new KeyActivationManager(Sensel,
        Activations.FD(FDs.Linear(0,50,0,4), 4));

// you can define your own switch as a ForceCurve
ForceCurve customSwitch;
KeyManager = new KeyActivationManager(Sensel,
        Activations.FD(customSwitch, 4));

// you can define your own way to activate key press from force input
Activation pressTrigger;
KeyManager = new KeyActivationManager(Sensel, pressTrigger);
```

```
WootingPoller wootingPoller;                                             2

// you can define trave depth threshold for key activation
wootingPoller = new WootingPoller(
        this.keyboardView,
        AnswerTextBox,
        keyActivationDepth: 1);

// or you can use pre-defined threshold
wootingPoller = new WootingPoller(
        this.keyboardView,
        AnswerTextBox,
        WootingPoller.KeyActivation.Tachyon);

// touch activation is possible if touch detector is connected
wootingPoller = new WootingPoller(
    this.keyboardView,
    AnswerTextBox,
    WootingPoller.KeyActivation.Touch);
}
```

Figure 3: Simulating different activation methods for the touch keyboard (left), and the physical keyboard (right).

and substitution). Under this taxonomy, a research study by Chaparro et al. [1] measured typing performances (typing speed in WPM, and error frequency of [15]) for different key modalities with the same layouts. Their work found out that physical keyboards showed higher typing speed than the touch-based one. Also, substitution errors had the largest amount and frequency compared to other kinds of errors. Kano et al. [6] expanded previous works of character level errors into word level errors and phrase level errors.

Key Activation Methods for Typing

A research study performed by Rumelhart et al. [12] developed a performative model to simulate a skilled typists by observing several features of their key input stream. Kim et al. [8] focused on the limited states of touch keyboards (released and touched) compared to physical keyboards (released, touched, and pressed) and tried to distinguish touching and pressing state of touch keyboards by utilizing temporal threshold for activating each key. Oulasvirta et al. [10] designed a neuromechanical model for pressing a single button. They proposed that button press should be considered as an open-loop problem due to cue integration theory. They also claimed that models of sensory synchronization can aid the performance of temporal control. Inspired from this model, Kim et al. [7] proposed that activating at the highest impact can improve the performance of temporal pressing tasks of a single button. Their experiment showed that impact activation resulted higher success rates, less asynchrony, and less variance both in physical key and the touch-based key. Yi et al. [16] proposed a Markov-Gaussian decorer based key activation method utilizing locations and pressures whose performance were higher in one-handed touch-typing tasks.

APPARATUS

Touch Keyboard

We first built a touch keyboard (Fig. 2) based on Sensel Morph[1], a commercially available force-sensing sensor utilized by several research studies (e.g., [16]). Sensel Morph has high range of force range of 5-5000 grams at sensitivity of 30,000 levels. It also has high speed polling rate of 500 Hz and touch resolution 6502 DPI (dot per inch) with 1.25 mm pitch between 185 × 105 sensor elements.

We set the profile of each virtual key based on standard 1u key cap with its width of 18.5 mm and margin of 2.75 mm which is the same as the physical keyboard we use to compare (Wooting One). We maintained the size of each key but minimized the number of keys into 50, remaining typing keys and a few control keys. The key layout is modified from the one used in Logitech G710, one of commonly used standard layouts. We printed our keyboard layout into a removable sticker without vinyl coating and attached it over the force-sensing surface.

Key Activation Simulator

We implemented a custom event handler (Fig. 3) to receive each frame of force array and translate into key press event. To maximizing the design space of key activation to simulate, we designed our key press event handler to fine-tune parameters of various key activation methods. Each key can be activated at the start of touch, release phase, or the activation point can be determined even with a programmable force-displacement curve. We also implemented impact activation with peak area and force which were first proposed from a previous research study [7].

Physical Keyboard

We used Wooting One[2] to measure the travel depth of each key press. It uses analog linear switches called Flaretech Linear55 Red that can measure 255 levels of pressed depth ranging from 1.5 to 3.6 mm. It has fast polling rate of 1000 Hz and matrix layout scan delay of about 20ms. We covered copper tapes on top of 42 key caps (alphanumeric keys including backspace and return keys) and soldered nylon-coated wires to connect with multiplexers.

Touch Detector

Achieving high polling rate is critical for key input system. While using capacitive sensors with Arduino STL, the polling rate for each key linearly increased by about 1 ms for each key while touching or idle state. Under this result, the expected duration of polling 42 capacitive keys were about 50 ms while pressing and 43 ms in idle state. Considering the typing speed of experts is about 58.4 WPM (about 6 characters per second) [13], these latency were too slow to receive all the input correctly. Therefore, we utilized a power detector (LTC5507) in

[1] http://guide.sensel.com/morph/

[2] https://wooting.io/comparison

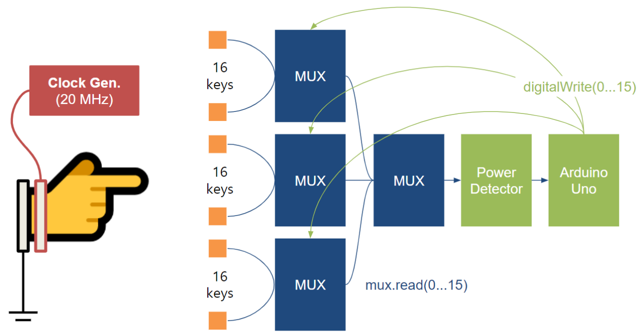

Figure 4: Schematics of our touch detector for the physical keyboard.

order to minimize the latency of touch sensing process. Power detector returns DC voltage if specific amount of frequency were detected. Detection frequency is tuned by capacities and resistors. For our project, we tuned our power detector to detect 20 MHz of square wave.

Figure 4 and 1 illustrates the schematic and the experimental setup of our touch detector. Each user wears wristbands for each hand emitting 20 MHz square wave. When user's hand touches a key cap, the cascaded multiplexers send the wave directly to power detector and determine the existence of touch. By doing so, we were able to be away from suffering the innate capacitance delay and result faster polling rate. We polled pins from each cascaded multiplexers by first fixing address to read for each child multiplexer and then reading values from each child's address by the root multiplexer. Since there were synchronization issue where too fast polling rate would fail to read the square wave signal, we had to manually decrease the speed for each Arduino loop. We used a clock generator (SI5351) and four 16-pin multiplexers (CD74HC4067) in our system.

SYSTEM VALIDATION

POLLING RATE

We set the polling rate of Sensel Morph as its maximum of 500 Hz. Polling rate of Wooting One is documented as 1000 Hz. We manually delayed polling rate of our touch detector as 1111 Hz. Our tool shows decent amount of polling rate that can correctly record input stream of expert-level typing speed of 58.4 WPM (each key press of 171.2 Hz) [13].

END-TO-END LATENCY

It is important to measure latency on our system since this system deals with high-res temporal duration. Therefore, we measured end-to-end latency of our system using internal LED of Sensel Morph. We set one of its internal LED to emit when it detects one or more contact points. We recorded the temporal duration from the LED emission to the display update by recording 960 FPS (frames per second) of high-speed video. While recording, we tried to press each key in constant frequency by listening constant beeping sounds. Numbers of recorded trials were three times for each activation method.

Several activation methods are considered to measure end-to-end latency of our system. For our touch keyboards, we measured end-to-end latency under three different activation methods (touch, peak force, and release). For our physical keyboards, we measured latency under three different travel depth activation points (touch, 2.0 mm, and 3.8 mm). We considered 2.0 mm of travel depth since is the standard travel key depth of keyboard famously used by Cherry MX switches. The maximum key depth (3.8 mm) and peak force activation are also considered for the sake of impact activation [7]. Release activation is considered since most of the touch keyboards on smartphones invoke key press event on releasing.

While using the touch keyboard, the average temporal delay were 80.21 ms (SD=1.473) for touch activation, 139.9 ms (SD=50.71) for peak force activation, and 147.9 ms (SD=10.42) for release activation. Note that since touch event invokes first than the rest of touch-keyboard events, the delay of touch event represents the minimal systemic delay of our system. However, since we could not control the key pressing speed, the latter two values do not represent the end-to-end latency of our system, but rather represents the demonstrator's mental model for neuromechanical control. Observing the deviation of delay in each conditions, we can see that peak force condition showed higher temporal variance compared to other conditions with its deviation less than 5% of its corresponding mean. It is notable since impact activation is reported to increase the performance of temporal button pressing tasks [7]. We assume this high variance is occurred due to either the visual feedback of typing interface or the lack of control in the experimental environment.

While using the physical keyboard, the average temporal delay were 146.4 ms (SD=6.771) for touch activation, 154.7 ms (SD=43.23) for peak force activation, and 156.7 ms (SD=12.50) for release activation. Also same as the measures while using the touch keyboard, the three values represents the demonstrator's mental model. Activation points with tactile feedback (touch and maximal depth of 3.8 mm) showed lower variance compared to the activation point without it (2.0 mm), which is consistent with the result of [7].

CONCLUSION

We developed a tool to design and simulate different activation methods for touch and physical keyboards. We successfully implemented activation points of touch, release, impact activation [7], and even given travel depth threshold of custom mechanical switches which can be utilized in both touch and physical keyboards. We minimized the polling rate by utilizing a power detector and multiplexers to correctly record input stream of expert typewriters. Our system showed end-to-end latency of 80.21 ms which is decent enough for simulating everyday desktop environments. We expect this tool to be used by interaction designers and researchers to optimize force curve of key press actions of their needs, and to observe how people type in different key activation functions and input modalities. We plan to aid the neromechanical control while using touch keyboards by utilizing different sensory inputs as our future work.

REFERENCES

1. Barbara S Chaparro, Mikki H Phan, Christina Siu, and Jo R Jardina. 2014. User Performance and Satisfaction of Tablet Physical Keyboards. *Journal of Usability Studies* 9, 2 (2014).
2. Donald R Gentner, Jonathan T Grudin, Serge Larochelle, Donald A Norman, and David E Rumelhart. 1983. A glossary of terms including a classification of typing errors. In *Cognitive aspects of skilled typewriting*. Springer, 39–43.
3. John D Gould, Sharon L Greene, Stephen J Boies, Antonia Meluson, and Marwan Rasamny. 1990. Using a touchscreen for simple tasks. *Interacting with computers* 2, 1 (1990), 59–74.
4. Jens Grubert, Lukas Witzani, Eyal Ofek, Michel Pahud, Matthias Kranz, and Per Ola Kristensson. 2018. Text entry in immersive head-mounted display-based virtual reality using standard keyboards. In *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE, 159–166.
5. Iman Habib, Niklas Berggren, Erik Rehn, Gustav Josefsson, Andreas Kunz, and Morten Fjeld. 2009. Dgts: Integrated typing and pointing. In *IFIP Conference on Human-Computer Interaction*. Springer, 232–235.
6. Akiyo Kano, Janet C Read, Alan Dix, and I Scott MacKenzie. 2007. ExpECT: an expanded error categorisation method for text input. In *Proceedings of HCI 2007 The 21st British HCI Group Annual Conference University of Lancaster, UK 21*. 1–11.
7. Sunjun Kim, Byungjoo Lee, and Antti Oulasvirta. 2018. Impact activation improves rapid button pressing. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–8.
8. Sunjun Kim, Jeongmin Son, Geehyuk Lee, Hwan Kim, and Woohun Lee. 2013. TapBoard: making a touch screen keyboard more touchable. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 553–562.
9. Alexander Otte, Tim Menzner, Travis Gesslein, Philipp Gagel, Daniel Schneider, and Jens Grubert. 2019. Towards utilizing touch-sensitive physical keyboards for text entry in virtual reality. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE, 1729–1732.
10. Antti Oulasvirta, Sunjun Kim, and Byungjoo Lee. 2018. Neuromechanics of a button press. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–13.
11. Jun Rekimoto, Takaaki Ishizawa, Carsten Schwesig, and Haruo Oba. 2003. PreSense: interaction techniques for finger sensing input devices. In *Proceedings of the 16th annual ACM symposium on User interface software and technology*. 203–212.
12. David E Rumelhart and Donald A Norman. 1982. Simulating a skilled typist: A study of skilled cognitive-motor performance. *Cognitive science* 6, 1 (1982), 1–36.
13. Andrew Sears. 1991. Improving touchscreen keyboards: design issues and a comparison with other devices. *Interacting with computers* 3, 3 (1991), 253–269.
14. Ying-Chao Tung, Ta Yang Cheng, Neng-Hao Yu, Chiuan Wang, and Mike Y Chen. 2015. FlickBoard: enabling trackpad interaction with automatic mode switching on a capacitive-sensing keyboard. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 1847–1850.
15. Jacob O Wobbrock and Brad A Myers. 2006. Analyzing the input stream for character-level errors in unconstrained text entry evaluations. *ACM Transactions on Computer-Human Interaction (TOCHI)* 13, 4 (2006), 458–489.
16. Xin Yi, Chen Wang, Xiaojun Bi, and Yuanchun Shi. 2020. Palmboard: Leveraging implicit touch pressure in statistical decoding for indirect text entry. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–13.