GestAKey: Get More Done with Just-a-Key on a Keyboard

Yilei Shi, Tomás Vega Gálvez, Haimo Zhang, Suranga Nanayakkara
Singapore University of Technology and Design
8 Somapah Road, Singapore
{yilei, tomas, haimo, suranga}@ahlab.org

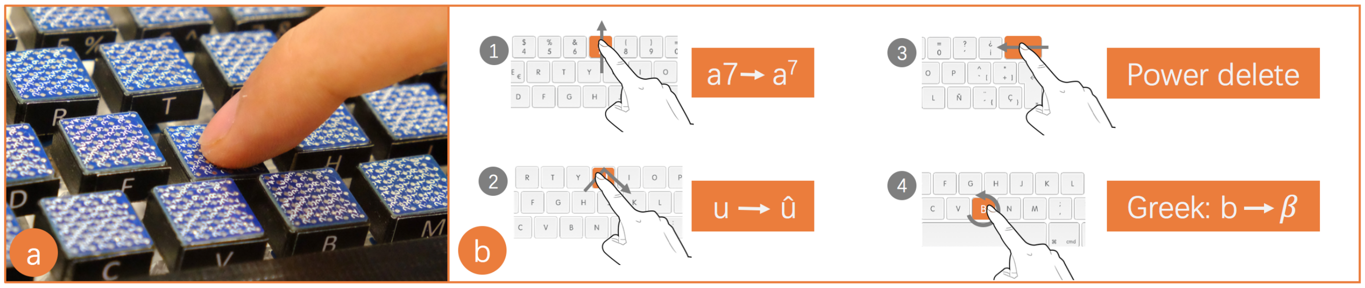


Figure 1. GestAKey, enabling novel keyboard interactions by detecting micro-gestures on individual keycaps. (a) Custom-made capacitive sensing keycaps mounted on a regular keyboard. (b) Microgestures on keycaps open up new possibilities: b1) Superscript with swipe up; b2) Accented letters by drawing the accent on the letter, b3) Deleting contents on the left or right with swipe left/right, b4) Greek letters by circling.

ABSTRACT
The computer keyboard is a widely used input device to operate computers, such as text entry and command execution. Typically, keystrokes are detected as binary states (e.g. "pressed" vs. "not pressed"). Due to this, more complex input commands need multiple key presses that could go up to pressing four keys at the same time, such as pressing "Cmd + Shift + Opt + 4" to take a screenshot to the clipboard on MacOS. We present GestAKey, a technique to enable multifunctional keystrokes on a single key, providing new interaction possibilities on the familiar keyboards. The system consists of touch sensitive keycaps and a software backend that recognizes micro-gestures performed on individual keys to perform system commands or input special characters. In this demo, attendees will get the chance to interact with several GestAKey-enabled proof-of-concept applications.

Author Keywords
Keyboard; Microgesture; Multifunctional Keystroke.

ACM Classification Keywords
H.5.2 User Interface: Input devices and strategies

INTRODUCTION
Since their invention in the middle of the 19th century, computer keyboards remain an important input device for text entry and command execution. However, the binary nature of the keystroke action creates a bottleneck for efficient interaction. For example, to input symbols or commands, users need to either memorize and use hotkeys (e.g., Ctrl+Alt+Del), or click through button menus (e.g., to insert a special symbol using Microsoft Word's insert symbol dialog) in the GUI. These operations are unintuitive and hard to learn [3], and interrupt users from ongoing text entry tasks, leading to lower efficiency. Previous researches have explored diverse methods to enhance the expressiveness of physical keyboards. Taylor [4] and Zhang[5] enable gesture-based interactions on a physical keyboard. Zheng [6] and Dietz [2] augmented keystrokes on the entire keyboard with finger detection and pressure sensing. We believe it is also possible to unlock keystroke expressiveness within the area of a single keycap. In this paper, we present GestAKey, a technique to detect micro-gestures[1] on individual keycaps. This opens up intuitive and efficient keyboard interactions such as inputting accented letters and Greek letters with a simple swipe on the keycap. The contributions of this paper include 1) a hardware prototype of a touch-sensitive keycap that can be used to augment a regular keyboard; 2) a software architecture that classifies multiple gestures and translates them into system commands; 3) proof-of-concept applications that enable efficient text input and system operations.

IMPLEMENTATION
GestAKey consists of two interacting systems: 1) The hardware that reads touch data from keycaps when pressed; 2) The software that listens for key events, recognizes gestures, and performs the corresponding functions.

Hardware Design
The hardware system has three components: 1) 3D-printed keycaps equipped with capacitive sensing matrices (5x4 points) and custom-made PCBs (14x14mm) (Figure 2); 2)

UIST'17 Adjunct, October 22–25, 2017, Québec City, QC, Canada.
©2017 Copyright is held by the owner/author(s).
ACM ISBN 978-1-4503-5419-6/17/10.
http://dx.doi.org/10.1145/3131785.3131786

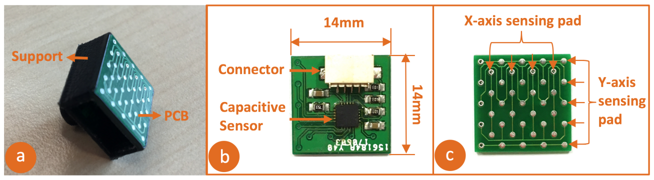


Figure 2. Structure of a single GestAKey keycap: (a) Each keycap consists of two parts: 3D-printed support and PCB. (b) Bottom layer: capacitive sensing IC and connector; (c) Top layer: sensing pad matrix.

Multiplexers connecting to capacitive sensing keycaps; 3) Microcontroller Unit (MCU) that reads data from the capacitive sensing keycaps via the multiplexers and transmits to the computer. The top layer of the PCB has a 5x4 matrix of capacitive sensing pads and bottom layer has a capacitive sensing IC (MPR121) and a connector to the multiplexer. When a key is pressed, the MCU receives the corresponding keycode from the operating system, reads touch data from the capacitive sensing matrix of the respective keycap, and sends it back to the software system (for classification). Given that MPR121 can only be configured to 4 different I2C addresses, we used multiplexers to read from 27 different keys.

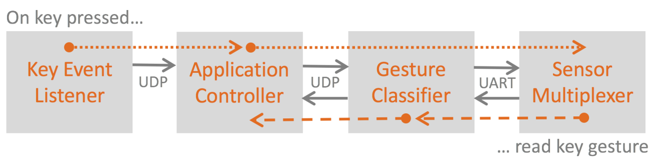


Figure 3. Data flow in the software system. Gray arrows between the components indicate the actual communication protocols used for data transmission. Red lines indicates two directions of data flow: keystroke events (orange dotted line), and gesture data (orange dashed line).

Software System

The software system consists of the following four parts. **Key Event Listener**: a C program that listens for OS-level key presses/releases; **App Controller**: a Node.js server that orchestrates data flow and triggers functions; **Gesture Recognizer**: a Python script that classifies micro-gestures; **Sensor Multiplexer**: an Arduino program that reads data from multiple capacitive touch matrices. In the current implementation, there are two directions of data flow, for keystroke events (orange dotted line) and gesture data (orange dashed line) respectively (Figure 3). **Keystroke:** The Key Event Listener listens for key press/release events and sends them to the App Controller, which in turn relays them to the Sensor Multiplexer. The MCU then multiplexes into the touch sensing matrix of the corresponding keycap and extracts raw data. **Gesture:** The Sensor Multiplexer sends touch data from the selected keycap to the Gesture Classifier. Classified gestures are then sent to the App Controller, where application-specific functions are triggered using Apple's JavaScript for Automation framework.

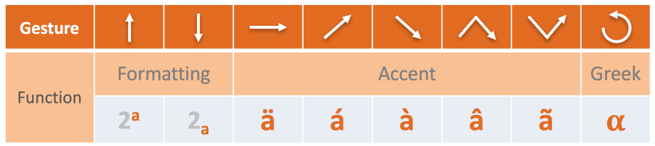
APPLICATIONS

Special function keys

We augmented two function keys **Fn** and **Backspace** for two special operations: ”Mode Change” and ”Power Delete”.

Mode Change: A downward swiping gesture while pressing the **Fn** key switches between different GestAKey modes. Currently, two modes are supported: 1) a Shortcut Mode that allows single-key interactions with applications; 2) a Symbol Mode that allows inputting special characters.

Power Delete: Typically, the backspace/delete key erases a single character to the left/right of the current cursor position. To delete text by words or sentences, the user needs to either select the words or sentences before deletion, or use modifier keys in combination with the backspace/delete keys, to change the granularity of deletion. GestAKey allows the user to delete by words and sentences in both directions from the cursor, by combining the keystroke on the backspace key with swipe gestures on its keycap. For example, a left/right swipe deletes a word to the left/right of the cursor, while an up/down swipe deletes a sentence to the left/right. This simplifies the different modes of text deletion into simple presses and swipes.



Gesture	↑	↓	→	↗	↘	∧	∨	↻
Function	Formatting		Accent					Greek
	2^a	2_a	ä	á	à	â	ã	α

Figure 4. In Symbol Mode, GestAKey enables users to input three kinds of formatting text with micro-gestures: (1) Superscript and subscript, (2) Accented and (3) Greek letter.

Alphanumerical keys

Shortcut Mode: GestAKey enables easy access to commonly used functions. Users can open an application or perform app-specific functions, by pressing a key and performing gestures on that key. For example, open a YouTube tab in safari by doing circling gesture on “Y”. When listening to music, pressing “m” and swiping up on its keycap can display the information of the song playing, and pressing “m” and swiping down can add a song to the library.

Symbol Mode: Styling text, such as adding accents and transforming to superscript/subscript, is often non-intuitive, requiring users to memorize special hotkey combinations (e.g., pressing “option+e, a” for “á”), or a sequence of interactions with GUI elements (e.g., selecting the text, then apply superscript/subscript from the text format toolbar). GestAKey allows for simultaneous inputting and styling of text, which is as easy as pressing and swiping. For example, pressing “a” and swiping top-right inputs “á”, pressing and swiping up or down inputs a super/subscript, and pressing and performing a circular gesture inputs Greek letters (e.g. pressing and circling on “a” input α). The mapping between gestures and text style is shown in Figure 4.

CONCLUSION

In this work, we introduced GestAKey, a technique that enables multifunctional keystrokes on a keycap via micro-gesture recognition. We believe that this opens up new design spaces for interactions on a keyboard. The interaction design of GestAKey technology focuses on intuitiveness, learnability, and interaction efficiency, which will be the subject of our future investigation.

REFERENCES

1. Chan, L., Chen, M. Y., and Chen, W.-h. C. B.-y. FingerPad : Private and Subtle Interaction Using Fingertips. 255–260.
2. Dietz, P. H., Eidelson, B., Westhues, J., and Bathiche, S. A practical pressure sensitive computer keyboard. *Uist* (2009), 55.
3. Hammond, J. M., Harvey, C. M., Koubek, R. J., Compton, W. D., and Darisipudi, A. Distributed Collaborative Design Teams : Media Effects on Design Processes Distributed Collaborative Design Teams : Media Effects on Design Processes. *International Journal of Human-Computer Interaction 7318*, June 2015 (2005), 37–41.
4. Taylor, S., Keskin, C., Hilliges, O., Izadi, S., and Helmes, J. Type-hover-swipe in 96 bytes. *Proceedings of the 32nd annual ACM conference on Human factors in computing systems - CHI '14* (2014), 1695–1704.
5. Zhang, H., and Li, Y. GestKeyboard. *Proceedings of the 32nd annual ACM conference on Human factors in computing systems - CHI '14* (2014), 1675–1684.
6. Zheng, J., and Vogel, D. Finger-Aware Shortcuts. *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems - CHI '16* (2016), 4274–4285.