Touch-Display Keyboards and their Integration with Graphical User Interfaces

Florian Block[1], *Hans Gellersen*[1], *Matthew Oppenheim*[1] *and Nicolas Villar*[2]
[1]Lancaster University, [2]Microsoft Research Cambridge
[1]{block, hwg, oppenhei}@comp.lancs.ac.uk, [2]nvillar@microsoft.com

ABSTRACT
We introduce Touch-Display Keyboards (TDK) that retain the traditional physical key layout and in addition provide dynamic display and touch-sensing capability for each key. We demonstrate how TDKs can be seamlessly integrated with graphical user interfaces by extending the graphical output as well as three-state input across the keyboard's surface. TDKs allow the graphical interface to be dynamically distributed across keyboard and display, exploiting both the benefits of physical controls and the flexibility of graphical widgets.

ACM Classification: H.5.2 [Information interfaces and presentation]: User Interfaces. - Input devices and strategies; Graphical user interfaces.

General terms: Design, Human Factors

Keywords: Touch-Display Keyboards, GUI, Interface Customization, Multi-Touch Input

Introduction

In this paper, we introduce *Touch-Display Keyboards* (TDKs) that are seamlessly integrated with the interaction space of graphical user interfaces. TDKs are based on conventional keyboards and integrate novel display and touch-sensing capabilities. Each key has the ability to display dynamic graphical output, as also demonstrated in commercial adaptive keyboards [1]. The keys further integrate touch-sensing, allowing each key to sense fingers touching, in addition to pressing and releasing [3]. The integration of both display and touch provides the keyboard with properties of graphical interfaces while retaining the physical characteristics of the conventional keyboard.

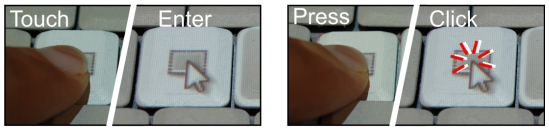
TDKs are integrated with the GUI in a novel way, by conceptualizing the surface of the keys as coherent display area that seamlessly extends from the primary screen (cf. Figure 1). In this way, graphical output can be seamlessly extended to the keyboard. In the same manner, mouse interaction is supported across primary screen and keyboard. The Touch-Display Keyboard further provides three-state input, allowing it to consistently support interaction techniques known from mouse input, such as pointing and selecting [2] (cf. Figure 2.

Figure 1: Touch-Display Keyboards extend mouse input and output space of existing graphical user interfaces, allowing the seamless distribution of interface elements across screen and keyboard.

TDKs can be operated like conventional keyboards by simply pressing keys. However, instead of producing conventional key events in the operating system (such as key pressed / and released), TDKs convert interaction with its keys as mouse events within the graphical space they display:

1. *Touching* a key is mapped to the same event that a mouse would cause by moving into the screen area of the key. From an interaction perspective, this corresponds to mouse pointing, while in the domain of GUI programming this could be described as a *hover* or *MouseEnter* event.

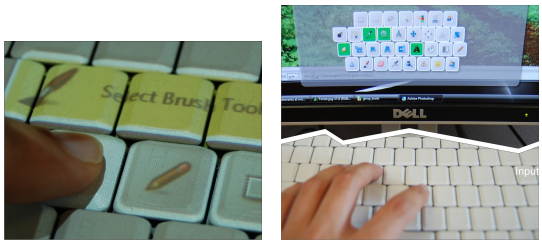
2. *Pressing* a key is remapped to the same event that a mouse would cause when clicking at the screen area that covers the key. To the user, this constitutes the selection of a control, while for developers it corresponds to a *MouseClick* event. Note that this event actually consists of two events,



a) Point b) Select

Figure 2: TDKs support three-state input, allowing the implementation of fundamental input techniques across mouse and keyboard input, such as point and select.

UIST'09, October 4-7, 2009, Victoria, British Columbia, Canada.



(a) Tooltips on the keyboard's surface (b) Preview of finger positions on the primary screen

Figure 3: Touch-sensing is used to provide pro-active preview.

namely *MouseDown* and *MouseUp* and pressing a key down and releasing a key are mapped accordingly.

This model for input and output on TDKs allows for consistent interaction across keyboard and display. The demonstration will focus on how this allows users and developers to apply existing practices to working with keyboard interfaces and how TDKs can facilitate novel ways of interacting with physical controls and graphical widgets.

Interacting with Touch-Display Keyboards

Based on their new hardware capabilities and their integration with graphical user interfaces, TDKs facilitate several new interaction styles:

Interface Exploration. Novice users can explore TDKs via the mouse in the same way they explore graphical interfaces. Moving the mouse over controls on keys invokes tooltips that are directly displayed on the keyboard's surface. Since TDKs also provide three-state input that maps on identical mouse events, tooltips can also be inherently invoked by touch, before the finger actually triggers the control (cf. Figure 3(a)).

Pro-Active Preview. TDK's also facilitate pro-active preview for supporting scenarios, in which it is beneficial operate keyboard interfaces by touch, while visually focusing on the primary screen (cf. [3]). Using the built-in touch-sensors, TDKs assist the tactile acquisition of keys, by previewing the fingers' current position on a virtual on-screen keyboard close to the task on the primary screen (cf. Figure 3(b)).

Multi-Touch Gestures. Since each key produces individual touch-events, TDKs can detect a variety of multi-touch gestures on the keyboards surface, such as swiping or pinching. Those gestures can serve as triggers that can be mapped to actions, in addition to key input.

Sensing User Context. TDKs can detect the users activity by analyzing the finger's presence (or absence) on the keyboard. This can be used to automatically adapt keyboard interfaces to different activities. For instance, the TDK can sense the intention to write (the fingers of both hands rest on the home row) or to trigger commands (dominant hand is absent from the keyboard) and switch between text input and a command interface, accordingly.

Drag&Drop Interface Customization. TDKs can enable new ways of dynamically customizing graphical interfaces across

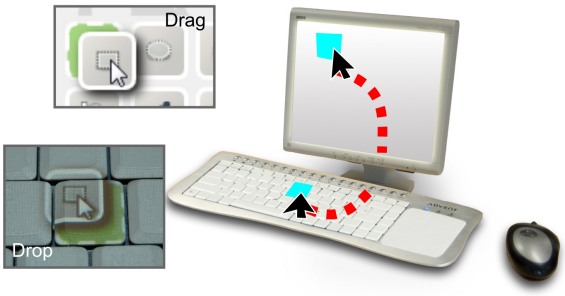


Figure 4: TDKs allow the flexible distribution of graphical interface elements between keyboard and display via the mouse using drag-and-drop.

keyboard and display. Because of the consistent input and output layers that are underlying TDKs, toolbar buttons can simply be dragged in between display to the keyboard using the mouse (cf. Figure 4). This way, users can dynamically customize their interface according to their needs, accessing both the flexibility of graphical widgets as well as the benefits of physical controls.

Implementation

The demonstration prototype is a fully functional TDK that supports the described interactions (cf. Figure 5). Overhead projection is used to augment a blank keyboard with dynamic graphical output (Figure 5(c)). Capacitive touch-sensing is embedded into each key and the case of the keyboard (Figure 5(a,b,d)). Gestures are supported by interpolating across multiple binary touch-events. The implemented TDK is compatible with existing applications through its generic input layer that injects mouse events into running applications. Furthermore, a custom interface toolkit, implemented using WPF and .NET, is demonstrated, allowing advanced interaction techniques, such as the drag-and-drop customization of keyboard interfaces (cf. Figure 4).

REFERENCES

1. Optimus Maximus Keyboard, ArtLebedev Studios. http://www.artlebedev.com/everything/optimus/.

2. W. Buxton. A three-state model of graphical input. In *Proc. INTERACT '90*, pages 449–456, Amsterdam, The Netherlands, The Netherlands, 1990. North-Holland Publishing Co.

3. J. Rekimoto, T. Ishizawa, C. Schwesig, and H. Oba. Presense: interaction techniques for finger sensing input devices. In *Proc. UIST '03*, pages 203–212, New York, NY, USA, 2003. ACM.

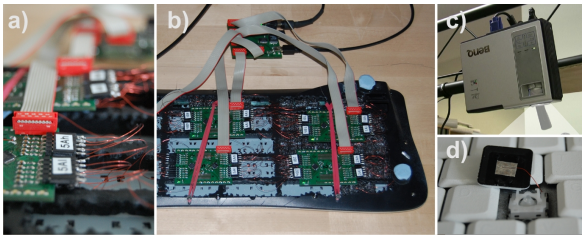


Figure 5: The TDK-demonstrator is implemented using capacitive touch-sensing and overhead projection.