Pointer: Multiple Collocated Display Inputs Suggests New Models for Program Design and Debugging

Marcello Bastéa-Forte, Ron Yeh, Scott R. Klemmer
Stanford University HCI Group
Computer Science Department
Stanford, CA 94305-9035, USA
{graphite, ronyeh, srk}@cs.stanford.edu

ABSTRACT
With multi-touch, multi-user display devices becoming more prominent, a need for developing applications that support multiple input sources is emerging. We present Pointer, a toolkit that abstracts and aggregates multiple input sources while also providing a framework to help test and develop software that utilizes multiple input sources.

The toolkit event abstraction homogenizes multiple input sources and allows applications to differentiate them only as needed. Furthermore, the toolkit event aggregator affords a number of debugging tools such as recording and real-time input monitoring to assist developers.

ACM Classification: H.5.2. [Information Interfaces]: User Interfaces — *input devices and strategies; prototyping; user-centered design*. D.2.2 [Software Engineering]: Design Tools and Techniques — *User interfaces*.
General terms: Design, Human Factors

Keywords: Toolkits, Multi-touch displays, Debugging

INTRODUCTION
Although prior work has looked at implementation and interaction design issues with multiple input sources [1][3][5], there is little research in providing developers with tools for building such applications. As we explored integrating two existing systems (DiamondTouch multi-touch table, and Anoto digital pens) to experiment with new interaction techniques with multiple input point devices, we found ourselves constructing tools that simplified development.

We built the Pointer toolkit for Java programmers developing multi-input, multi-user applications. We abstract inputs from multiple input devices into one unified event system. The API leverages existing APIs to provide connectivity to available hardware. By integrating these systems we provide a single point of entry that streamlines debugging.

The toolkit was used to build two applications: a basic hello-world application intended as a building block for new developers, and a photo manipulation application that demonstrates integration of a multi-touch surface with a digital pen. We intend to continue development of the developer toolkit and push it out to external developers so that we can evaluate its effectiveness.

UIST'07, October 7-10, 2007, Newport, Rhode Island, USA.
ACM 978-1-59593-679-2/07/0010

RELATED WORK
Existing research looks at multi-user multi-input systems from two directions: some at the low-end technical implementation of supporting multiple input devices on a computer [3] while others look at how user interaction is regulated in a multi-user multi-input system [1]. Some examine both [5].

The Pointer Toolkit attempts to fill the middle ground by assuming that the low-level components necessary to read multiple input sources are in place. It assists developers who want to easily develop applications to experiment with new user interactions.

When designing the debugging tools for Pointer, we followed principles put forth by existing toolkit research. Exemplar shows the importance of displaying real-time information when debugging an application that works with continuous signals [2].

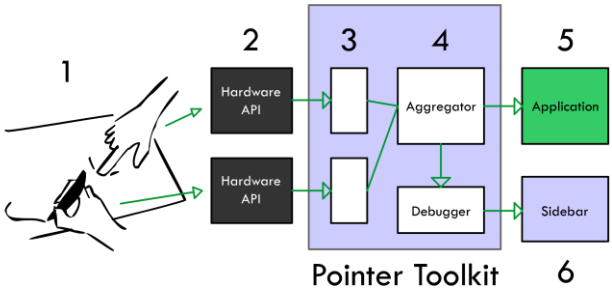

Figure 1: Input flow: user input (1) is read by an existing hardware API (2) which is sent through an input connector to the InputAggregator (4), which finally dispatches to the application (5).

THE TOOLKIT
The toolkit consists of two components: an API for aggregating and disseminating point events and a debugging tool for analyzing and recording input as the application is running (Figure 1).

Pointer API
The Pointer API is based on the standard Java Event architecture, providing a standard input listener interface for applications to listen for point events.

Point events represent hover, pressing, dragging, and releasing with coordinate, pressure, and size information. Since not all this information is available for all devices (for example, most touch displays cannot recognize hover),

a point source object associated with the event can be used to determine which values are provided. Point events also have information about their source hierarchy. For example, events from the DiamondTouch table link to the user who initiated them which links to the table from which the events are being received. Several input connectors are provided to interface with mice, DiamondTouch tables, Anoto pens, and graphics tablets.

In addition, the API provides an InputAggregator class that acts as a centralized interface for interacting with different sources by receiving and broadcasting point events.

Pointer Debugging Interface

With central input aggregation we can easily provide a number of techniques to help developers debug their applications.

The first tool we can provide is implicit logging of all events to a file. This file can then be played back at any point to simulate input and create unit tests.

The second tool is a sidebar GUI that provides real-time feedback of all input sources. This was designed to replace the common debugging practice of printing out raw numbers to the console. The sidebar makes it clear if input is coming to your application and gives access to the raw data without modifying your application (Figure 2 inset).

POINTER APPLICATIONS

We built two applications with Pointer. The first is a rudimentary hello world application. The second is a photo tagging application that inspired the Pointer project.

Hello World

The hello world application enumerates all event sources and displays each individual input as a collection of dots. Related events are connected by lines. This application demonstrates how to use the Pointer API and provides a starting point for new developers.

Photo Tagging

We used the toolkit to develop a photo tagging application for the DiamondTouch table (Figure 2). We integrated the Anoto digital pen with by printing a table-sized print with the Anoto pattern that the digital pen recognizes.

Figure 2: Photo tagging application we developed using the Pointer API. Users can move pictures with their hands and annotate them with a pen. Inset: the debug sidebar.

The DiamondTouch table allows us to independently recognize multiple input points from up to four different users while the Anoto pens allow us to accurately read stylus input across the entire table from multiple sources.

We use this to give the user two modes of interacting. They can move the photos around with their hands: scaling and rotating photos by using multiple fingers. The digital pen allows users to annotate photos by writing on them.

The application uses a pair of input handlers that receive point events from the Pointer API. The handlers include a manipulation handler and stroke handler used to orient/scale and write on the photos, respectively.

Pointer input abstraction made application design implicitly support multiple users scaling and rotating photos collaboratively even though their input comes from different sources. This allows for faster prototyping of interfaces such as collaborative gestures [4]. Furthermore, since the input sources are generic, it is simple to reconnect the various handlers to touch input, mouse input, or digital pen input for testing and exploring new interactions.

CONCLUSION AND FUTURE WORK

Our maiden venture into the world of multi-point input APIs shows that it is worth exploring new toolkits for new interaction models. Initial applications developed using the Pointer API and toolkit demonstrated the ease of rapid development of multi-touch applications. Homogeneous input events seem to encourage homogenous input handling—resulting in applications behaving more as the user expects.

We intend to expand our debugging interface from the early prototype stage to a more robust system that we can use to evaluate our system with developers and iterate on its design.

REFERENCES

1. Bier, E. A. & Freeman, S. MMM: a user interface architecture for shared editors on a single screen. *UIST: ACM Symposium on User interface Software and Technology*: pp.79-86, 1991.
2. Hartmann, B., Abdulla, L., Mittal, M., & Klemmer, S. R. 2007. Authoring sensor-based interactions by demonstration with direct manipulation and pattern recognition. *CHI: ACM Conference on Human Factors in Computing Systems*: pp. 145-154.
3. Hourcade, J.P., & Bederson, B.B. Architecture and Implementation of a Java Package for Multiple Input Devices (MID). Tech Report HCIL-99-08, CS-TR-4018, UMIACS-TR-99-26, C.S. Department, University of Maryland, MD: 1999.
4. Morris, M. R., Huang, A., Paepcke, A., & Winograd, T. Cooperative gestures: multi-user gestural interactions for co-located groupware. *CHI: ACM Conference*. R. Grinter, T. Rodden, P. Aoki, E. Cutrell, R. Jeffries, and G. Olson, Eds. 2006: pp.1201-1210.
5. Myers, B. A., Stiel, H., and Gargiulo, R. Collaboration using multiple PDAs connected to a PC. *CSCW: ACM Conference on CSCW*. 1998: pp. 285-294.