Bridging the Gap Between Desktop Computers and TableTop Displays

Peter Hutterer and Bruce H. Thomas
HxI Initiative – Project [braccetto]
Wearable Computer Laboratory
School of Computer and Information Science
University of South Australia
Mawson Lakes, SA 5095, Australia
Tel: 61-8-8302-3669
{peter | thomas}@cs.unisa.edu.au

Abstract

Traditional desktop environments are designed on the assumption of having a single keyboard focus and a single mouse cursor. Tabletop displays on the other hand are often used as multi-user interactive displays. User interface software built for one of these environments cannot easily be run in the other environment. This paper presents Multi-Pointer X (MPX), a modified X Window System to accommodate for multiple input devices. MPX allows traditional single-user applications to run simultaneously with novel multi-user applications. Input devices can be hot-plugged at any time and allow users to interact with multiple applications simultaneously. Ambiguity in legacy APIs is resolved with the novel "Client-Pointer" principle. MPX also provides new APIs for collaborative applications and thus bridges the gap between traditional desktop applications and tabletop applications.

1. Introduction

A common use for tabletop displays is as a collaborative workspace. Traditional PCs on the other hand are used almost exclusively as single-user workstations. This leads to a strong disparity between the software used on one's workstation and the software used to collaborate across a tabletop display.

Our Multi-Pointer X (MPX) [3] turns the windowing system into a single display groupware environment. MPX allows any single-user application to be run in a multi-user context. Novel multi-user applications can utilise all the features of a groupware environment. MPX thus reduces the difference between a traditional single-user PC and a collaboration environment such as a tabletop display. Users can simultaneously interact with standard single-user applications as well as collaborative applications.

MPX is not a collaboration application, but the underlying technology for collaboration that does not depend on application support. The collaboration features are integrated in the windowing system and available to any application, that is both single-user and groupware applications. A tabletop display can thus be used as a single-user workstation and transition ad-hoc to multi-user mode as people naturally walk up to a table to collaborate. MPX is an important step towards broadening the single-pointer single-keyboard paradigm that dominates current user interfaces.

2. Related Work

Stewart et al. found that users prefer individual input devices when working with Single Display Groupware (SDG) [4] but multi-user toolkits (e.g. [5]) require applications to be modified to make use of multi-user capabilities. At the same time, they restrict the use of traditional single-user applications. On the other hand, multi-touch hardware for tabletop displays is becoming more popular (e.g. [2]). Applications have to use custom-built toolkits to leverage the multi-user capabilities of these tables. This again restricts the use of these applications on a traditional desktop computer.

The adoption of groupware depends on its support for everyday applications [1], yet few groupware toolkits focus on collaboration in legacy applications on a single display. Instant transition between single-user and multi-user mode while multiple applications are running is not supported by any current technology other than MPX.

3. MPX

MPX is the first GroupWare Windowing System (GWWS) [3] and supports up to 128 independent input devices. MPX changes the input subsystem of the X.org X server. MPX keeps state information for all devices and delivers the event based on a device's focus. These different event streams allow interaction with multiple applications simultaneously.

MPX also provides events with device IDs and extends the X Input Extension (XI). XI can thus be employed by future SDG applications to provide true multi-user functionality within the same application. MPX delivers core events and XI events based on the application's support and thus it is possible to run groupware applications and legacy single-user applications simultaneously.

4. Ad-hoc Collaboration with MPX

MPX creates additional cursors and keyboard foci for hot-plugged devices. MPX automatically pairs a new keyboard with the first available unpaired pointer. To enable an additional pair of input devices, a user only needs to plug in a mouse and a keyboard. A new cursor and a new keyboard focus become available and the user can start working immediately, without any additional setup. MPX also exposes an Xlib API to change the device pairing at any time if the default behaviour does not suffice. Whenever a device pairing is changed, applications are notified about the new pairing and the list of pointer-keyboard pairings can be queried at any time. Applications can thus adjust their interfaces or based on the current device pairing.

5. Resolving API Ambiguity

Virtually all applications are designed for an infrastructure that supports a single pointer and a single keyboard. As a result, single-user APIs can be ambiguous when multiple devices are availalable.

We introduced the notion of a *ClientPointer* to resolve these ambiguities. Each application can be assigned a distinct pointer device (paired with a keyboard), and this ClientPointer is chosen as the default device for all ambiguous requests. If an application issues an ambiguous keyboard request, the keyboard that is paired with the ClientPointer is chosen to provide data to the application. The ClientPointer setting is application-specific and an input device can interact with the application even if it is not set as ClientPointer. Only ambiguous requests from the application result in the preference for one cursor. If an application never issues an ambiguous request, all devices are equal.

The ClientPointer principle is designed for by legacy applications only. MPX provides APIs for multi-user applications that allow applications to avoid ambiguities. A multi-user application should never need to issue an ambiguous request.

6. Conclusion

MPX supports fluid transition between a single-user and a multi-user environment on a single display by simply adding additional input devices. Collaboration features are available in any application and any application that runs under X can be utilised immediately. Additional cursors and keyboard foci are added as new devices are plugged into the host computer. Automatic device pairing allows users to collaborate instantly after connecting new devices.

The ClientPointer principle resolves ambiguity in legacy APIs by assigning a single pointer to legacy applications. Ambiguous requests will default to the ClientPointer but any device can interact with the application. MPX broadens the single-pointer single-keyboard paradigm that is prevalent in traditional desktop environments. As a result, we can run single-user applications on a collaborative display and collaborative applications on a single-user workstation.

7. References

[1] Grudin, J. *Groupware and social dynamics: eight challenges for developers.* Commun. ACM, Vol. 37, No. 1, pp 92-105, 1994.

[2] Han, J. Y. Low-cost multi-touch sensing through frustrated total internal reflection. In *UIST '05: Proc's of the 18th annual ACM Symposium on User Interface Software and Technology,* pp 115-118, Seattle, WA, USA, 2005.

[3] Hutterer, P. and Thomas, B. H. Groupware Support in the Windowing System. In *AUIC '07: Proceedings of the eighth conference on Australasian user interfaces,* pp 39-46, Balarat, Australia, 2007.

[4] Stewart, J., Raybourn, E. M., Bederson, B., and Druin, A. When two hands are better than one: enhancing collaboration using single display groupware. In *CHI '98: CHI 98 conference summary on Human factors in computing systems,* pp 287-288, Los Angeles, California, United States, 1998.

[5] Tse, E. and Greenberg, S. Rapidly prototyping Single Display Groupware through the SDGToolkit. In *AUIC '04: Proceedings of the fifth conference on Australasian user interfaces,* pp 101-110, Dunedin, New Zealand, 2004.