iDwidgets: Parameterizing Widgets by User Identity

Kathy Ryall[1], Alan Esenther[1], Katherine Everitt[2], Clifton Forlines[1], Meredith Ringel Morris[3], Chia Shen[1], Sam Shipman[1], Frederic Vernier[4]

[1]MERL, 201 Broadway, Cambridge, MA, USA
[2]University of Washington, 185 Stevens Way, Seattle, WA, USA
[3]Stanford University, 353 Serra Mall, Stanford, CA, USA
[4]University of Paris 11, LIMSI-CNRS, BP 133, 91403 Orsay, France
{ryall, esenther, forlines, shen, shipman}@merl.com
everitt@cs.washington.edu, merrie@cs.stanford.edu,
frederic.vernier@limsi.fr

Abstract. We introduce the concept of *identity-differentiating widgets* (iDwidgets), widgets parameterized by the identity of their user. Although multi-user applications have become more common, most support only traditional "single-user" widgets. By adding user-identity information we allow interactions with today's widgets to be dynamically customized on a per-user basis in a group usage setting. The concept has inspired the design of new widgets as well. In this paper we describe example iDwidgets and define a conceptual framework based on what is being customized in the widget. iDwidgets can support novel interaction techniques in collaborative settings.

1. Introduction

WIMP interfaces provide a consistent look and feel across different computer platforms and applications. Widgets, standard reusable GUI elements, are a staple in that environment. To date most widgets have been designed to be used by one person at a time; within a single session, a widget will behave the same regardless of who is using it. Pebbles [5] and MMM [1] were two of the first systems to extend widgets, adding user identity information. Much of their focus is on the visual representation needed to distinguish people (and their actions/interactions) in shared-display settings. We further exploit user identity information, extending and generalizing the concept to encompass a larger set of functionality, and providing a conceptual framework in which the use of identity to customize interaction can be designed and analyzed.

iDwidgets (*identity-differentiating widgets*) are basic GUI building blocks for user-aware environments; the iDwidget's novelty is that its function, contents and/or appearance are parameterized by the identity of its current user amongst a group of co-present (local or remote) users. Thus an iDwidget may look or behave differently for different user identities. By identity we mean a person with particular preferences and privileges, or a tool associated with such a person (e.g., the stylus the person is using). A person may have multiple identities (e.g., Dad and Senior Engineer).

Although we focus our discussion on shared-display groupware (SDG) settings [7], iDwidgets are applicable in any multi-user environment. A number of toolkits have been developed for these spaces. Multi-rodent systems (e.g., MID [3]) give multiple people simultaneous access to traditional widgets. DiamondSpin [6] provides multiple toolbars (one for each user) and simultaneous interactions with traditional single-user widgets, and has recently added support for some iDwidgets concepts [4].

Likewise, the SDG Toolkit [8] supports group interaction, but does not introduce any new widgets. Tool-based systems may provide support for multiple people to work in parallel, but each tool is not inherently user-aware; it behaves the same regardless of who is using it. The goal of iDwidgets is to increase a widget's utility and to support widget reuse. A single widget instance serves multiple people, helping reduce clutter in shared-display groupware applications. In cases where widget replication may have advantages, iDwidgets do not preclude widget replication.

While our concept originates with the use of multi-user touch surfaces that can identify who is touching (e.g., DiamondTouch [2]), iDwidgets are applicable in any system that can provide identity information. Many ubiquitous computing environments exploit identity-differentiating technology (e.g., face recognition, biometrics, RFID) to build applications for multi-user environments; we propose embedding the identity information at a lower level, encapsulating it into reusable iDwidgets, rather than at the application or system level.

2. Customizing iDwidgets

The ability to identify the user of a widget is the key feature that enables iDwidgets to be defined; user-id becomes a parameter (or a tag on input) to the widget. iDwidget versions can be made of many traditional widgets by extending and customizing them in up to four dimensions: function, content, appearance, and group input.

Customizing Function: The widget can pass along the identity information to its event handler; the widget looks the same to all users, but behaves differently.

Multi-User Buttons: a single traditional pushbutton would perform a different function based on who is touching the button. For example, a multi-user 'Undo' button, when pushed by a particular person, causes the last action of that person to be undone. While the system is responsible for keeping track of the multiple undo stacks [5], the user identity parameter indicates which stack to access.

Semantic Interpretation: a single string (or graphic) may denote different objects for different people. For example, 'Dad' is a different alias for most people – only identifying the same person for siblings. In a photo-browsing application with a search feature, if John entered 'Dad' in the search box, the system would return pictures of his father; if Mary entered 'Dad' it would return all pictures of her father.

Differentiated Behavior: a single instance of a widget performs the same action, but behaves differently based on the user's identity. For example, a scroll bar may provide continuous scrolling for one user and discrete scrolling for another, based on their pre-specified preferences, or a paintbrush may vary in its numerical values of brush thickness range (e.g., thin, medium, broad) on a per-user basis.

Privileged Access: a particular widget may only work for some special subset of users. It may only respond when a specific individual interacts with it. A security widget, for example, might respond only to a senior member of the group touching it.

Customizing Content: When activated, the contents of the widget will look different to different people, potentially providing different options.

Custom Lists: a traditional list widget whose contents vary depending on who is accessing the list. A list of Bookmarks (web browsing), for example, varies across users. The contents of the displayed list may be generated on the fly depending on who is interacting with the widget.

Custom Menus: a traditional pull-down menu whose contents are the same across users, but access to certain menu items (or sub-menus) is determined based on who is interacting with the menu. Menu items may be reordered (e.g., most recently used items appear closer to the top as in Windows), inactive (e.g., grayed out) depending on who is using the menu, or even be removed, making it more of a custom list.

Customizing Appearance: A widget's appearance (e.g., fonts, colors) may be customized using identity information without changing its behavior or function.

Properties and Aesthetics: the colors, fonts, languages and other traditional graphical features of a widget may also be customized based on the user, while the widget functionality remains unaffected. While some might argue that label language on a widget may impact its functionality, we prefer to think of it as a user preference or a usability issue. Adapting some of the widget properties (e.g., large type) is especially important for elderly and disabled individuals.

Orientation: a widget's orientation may also be customized based on identifying information, which is especially relevant for horizontal displays. Combining user identity with location information would allow widgets to dynamically orient themselves to a particular user. Automatic handle positions may also be determined by user identity and location.

Customizing Group Input: iDwidgets may support or require group input, enabling interaction from multiple people. We distinguish these iDwidgets from previous multi-input widgets (e.g., two-handed input), which are typically used by one person. Multi-input widgets could, of course, be extended to exploit the identity information provided to an iDwidget.

Cumulative Effect: an iDwidget could require interaction from a number of people before taking some action. A voting widget, for example, may require all users to respond before tallying the result; as an alternative, it might only require a quorum to agree, without requiring everyone to respond. The number of distinct users interacting with the widget (as indicated by the identity information) would be the distinguishing factor.

Simultaneous Input: iDwidgets may also require simultaneous interaction in order to activate a widget. Two people, for example, may be needed to turn virtual silo keys in order to launch a missile. Other examples include large surface interactions, where one person cannot directly reach all the needed objects or regions.

Modal Input Sequences: iDwidgets can support parallel moded-interactions. By tracking identity information they can support the interleaving of different people's actions. One person can be in 'Delete' mode while another person is simultaneously in 'Annotate' mode. With traditional widgets once the 'Delete' mode is activated, the next object touched would be deleted. With iDwidgets only the next touch of the person in 'Delete' mode would delete an object.

Audit and Logging: iDwidgets also support easy logging and audit trail creation. Because the widget knows who is touching, the identification information can be added to the log file. Many other systems provide such multi-user audit support (such as MS Word's 'Track Changes' feature); iDwidgets incorporate it at the widget level.

In general, any attributes or behaviors that are customizable in traditional (single-user) computing environments (typically via user-profiles or resource files) can be exploited by iDwidgets. Rather than creating multiple tool bars or other controls (one per person) a single instance of an iDwidget may be used, providing flexible and customizable interaction for different people, and in some cases screen space savings.

3. Conclusions and Future Work

The power of a widget lies in encapsulating a set of behaviors and packaging them up along with graphical attributes so that it can easily be used and reused. The iDwidget concept, adding user identity as a parameter in order to customize a widget in a multi-user setting, enables interactions with widgets to be dynamically adapted on a per-user basis in a group usage setting. In addition to the benefits of personalized interactions, iDwidgets support widget reuse (or sharing). When and how it is appropriate to allow shared widgets, and which dimension(s) of customization should be used are two application-dependent questions, and should be left as a policy question for the application developer. While some fragments of the iDwidget concept are present in a number of other systems, iDwidgets promotes identity to a first-class widget parameter, providing a unifying framework for specifying the customization of shared widgets.

iDwidgets also raise new feasibility and usability issues. Which widgets lend themselves to identity differentiation, and which do not? What happens when two people simultaneously access a menu – do one user's preferences take priority, or do we temporarily replicate the menu? What effects will dynamically adapting widgets have on interactions with an application? When will customizing a widget enhance performance or hinder learnability?

iDwidgets represent a generalization of ideas already in practice today. The enabling technology is already available and in use, most notably in ubiquitous computing environments; many applications already incorporate fragments of the iDwidgets idea. By extending and generalizing the use of identity in widgets introduced in earlier work and moving the identity information out of the application or system level and into the widgets, iDwidgets provides a conceptual framework in which to think about the use of identity information to customize interaction, and opens the door for new application development and new lines of research.

References

1. Bier, E., and Freeman, S., "MMM: a user interface architecture for shared editors on a single screen," In Proc. of UIST 1991, pp. 79-86.
2. Dietz, P., and Leigh, D., "DiamondTouch: A multi-user touch technology," In Proc. of UIST 2001, pp. 219-226.
3. Hourcade, H.P., and Bederson, B.B., "Architecture and Implementation of a Java Package for Multiple Input Devices (MID)," HCIL Tech Report No. 99-08, 1999.
4. Morris, M.R., Ryall, K., Shen, C., Forlines, C., and Vernier, F., "Beyond 'Social Protocols': Multi-User Coordination Policies for Co-located Groupware," In Proc. of CSCW 2004.
5. Myers, B., Stiel, H., and Gargiulo, R. "Collaboration Using Multiple PDAs Connected to a PC," In Proc. of CSCW 1998, pp. 285-294.
6. Shen, C., Vernier, F.D., Forlines, C., and Ringel, M. "DiamondSpin: An Extensible Toolkit for Around-the-Table Interaction," In Proceedings of CHI 2004, pp. 167-174.
7. Stewart, J. Bederson, B., and Druin, A., "Single Display Groupware: A Model for Co-present Collaboration," In Proc. of CHI 1999, pp. 286-293.
8. Tse, E., and Greenberg, S. "SDGToolkit: A Toolkit for Rapidly Prototyping Single Display Groupware." Poster session. CSCW 2002.