The Large-Display User Experience

George Robertson, Mary Czerwinski, Patrick Baudisch, Brian Meyers, Daniel Robbins, Greg Smith, and Desney Tan
Microsoft Research

Large displays offer users significant benefits and usability challenges. The authors discuss those challenges along with novel techniques to address these issues.

Despite the increasing affordability and availability of large displays, most users continue to have display space that represents less than 10 percent of their physical workspace area. Similarly, many current interfaces assume that users have a relatively small display to access the much larger virtual world. Several factors indicate that this is now changing. The PC's increasing graphical-processing power is fueling a demand for larger and more capable display devices. Several operating systems have supported work with multiple displays for some time. This fact, coupled with graphic-card advancements over the past 10 years, has led to an increase in multiple monitor (multimon) use. Microsoft commissioned a Harris poll of 1,197 Windows OS users, and found that up to 20 percent of the information worker users ran multimons from a PC or laptop. Many other users are becoming aware that running multimons is an option.

There are two basic approaches to providing a large-display experience:

- Wall-sized displays typically offer seamless display surfaces that are created using multiple tiled projectors. These systems commonly use touch or pen input.
- Large desktop displays are typically multimon configurations with seams between the monitors caused by the monitor bezels. These systems commonly use keyboard and mouse for input.

In our Harris poll, the two top reasons participants cited for not running multimons were: limited physical desktop space and price concerns. However, display manufacturers are widely predicting that the price of liquid crystal displays (LCDs)—which have smaller desktop footprints—will drop dramatically over the next four years. In fact, this price drop has already begun: Most consumers can now get 25 percent more pixels by buying two 17-inch LCDs for about the same price as one 21-inch LCD. Also, all laptop manufacturers now sell their products with built-in multimon support, which could further increase the momentum for multimon use.

To take advantage of this possibility, we must address a fundamental question: How might users cope with and benefit from display devices that occupy 25 to 35 percent of their physical desk area—or perhaps one day, cover an entire office wall? As the "Related Work in Large-Display Usability" sidebar describes, many researchers have proposed answers. Our own approach to the question was to first evaluate large-display usability issues, then develop a series of research prototypes to resolve them. Here, we describe those issues and our proposed solutions.

Large-display usability issues

To adequately design for large displays and multimon systems, we must understand both the differences and similarities between multimon and single monitor use. We therefore ran formal laboratory studies of large-display productivity.[1] We also analyzed product support calls. In addition, we observed a group of single and multimon users over time by developing and deploying a windowing system logging tool called VibeLog.[2] Doing so let us discern higher-level activity patterns—such as the number of opened windows, and window activation and movement frequency—for different-sized displays. Our data analysis showed that:

- as the number of monitors increase, the number of visible windows increases, and
- window visibility is a useful measure of display-space management activity, especially for multimon users.

On the basis of our research, we identified six broad categories of large-display usability issues:

- *Losing the cursor.* As screen size increases, users accelerate mouse movement to compensate and it becomes harder to keep track of the cursor.
- *Bezel problems.* Bezels introduce visual distortion when windows cross them and interaction distortion when the cursor crosses them.
- *Distal information access problems.* As screen size increases, accessing icons, windows, and the start menu across large distances is increasingly difficult and time consuming.
- *Window management problems.* Large displays lead to notification and window-creation problems because windows and dialog boxes pop up in unex-

Published by the IEEE Computer Society 0272-1716/05/$20.00 © 2005 IEEE

Related Work in Large-Display Usability

Many researchers have studied the benefits and usability issues of large displays. Among the early observations were those by Guimbretiere and his colleagues,[1] and MacIntyre and his colleagues.[2]

Grudin[3] documents the usage patterns of CAD/CAM programmers and designers running multimons. Despite the limitations observed in current OS support, multimon users clearly like the extra screen real estate, and they adapt their windows and application layouts optimally for the number, size, orientation, and resolution of their displays. According to Grudin, most current multimon users claim they would never go back to a single monitor.

Czerwinski and colleagues[4] document a series of user studies demonstrating productivity benefits from multimon or large display use. One of their goals was to identify novel software applications that might better support multitasking between projects and applications. The studies showed a 12 percent significant performance benefit; that is, study participants accomplished a mix of typical office productivity tasks 12 percent faster with a large display. They also showed that users were more satisfied with large displays than small displays. In addition to productivity benefits, the studies showed that larger displays improve users' recognition memory and peripheral awareness.

Large display use might also reduce or eliminate gender bias, at least in some tasks related to 3D navigation of virtual worlds. Desney Tan and colleagues[5] report a series of studies demonstrating the advantages of using large displays in this domain. Many researchers have observed that male users are significantly more effective than female users at navigating 3D virtual spaces.[6,7] The Tan studies show that while large displays typically increase performance for all users, females improved so much that males and females performed equally well in virtual 3D navigation on large displays. In these studies, the large display's wider field of view increased users' ability to process optical flow cues during navigation—cues that females are more reliant upon than males. An example of optical flow cues are the scene changes that occur as a virtual camera moves from one location to another over time. That movement causes continuous changes in the images that fall on the eye. These optical flow cues are more effective on larger displays. Tan and colleagues ran their studies on DSharp, a seamless wide screen multiprojector display (see Figure A). They found that the optimal field of view for the tasks tested was about 100 degrees, the equivalent of a triple monitor display.

A **DSharp multiprojector display.**

References

1 F. Guimbretiere, M. Stone, and T. Winograd, "Fluid Interaction with High-Resolution Wall-Size Displays," *Proc. Symp. User Interface and Software Tech.*, ACM Press, 2001, pp. 21-30.

2. B. MacIntyre et al., "Support for Multitasking and Background Awareness Using Interactive Peripheral Displays," *Proc. Symp. User Interface and Software Tech.*, ACM Press, 2001, pp. 41-50.

3 J. Grudin, "Partitioning Digital Worlds: Focal and Peripheral Awareness in Multiple Monitor Use," *Proc. Computer–Human Interaction*, ACM Press, 2002, pp. 458-465.

4. M. Czerwinski et al., "Toward Characterizing the Productivity Benefits of Very Large Displays," *Proc. Interact 2003*, IOS Press, pp. 9-16.

5. D. Tan, M. Czerwinski, and G. Robertson, "Women Go with the (Optical) Flow," *Proc. Computer–Human Interaction* (CHI), 2003, pp. 209-215.

6. D. Kimura, *Sex and Cognition*, MIT Press, 1999, pp. 1-66.

7. T.R.H. Cutmore et al., "Cognitive and Gender Factors Influencing Navigation in a Virtual Environment," Int'l J. Human-Computer Studies, vol. 53, no. 2, 2000, pp. 223-249.

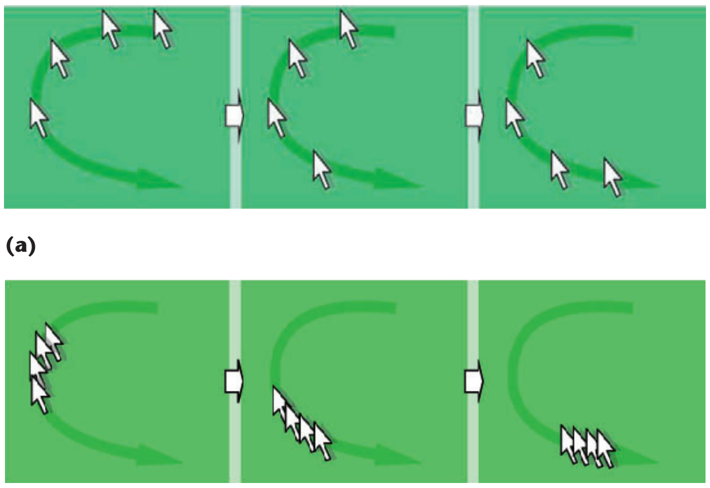
pected places. Window management is made more complex on multimon displays, because many users try to avoid having windows that cross bezels.

- *Task management problems*. As screen size increases, so too does the number of open windows. As a result, users engage in more complex multitasking behaviors and require better task management mechanisms.
- *Configuration problems*. The user interface for configuring multimon displays is overly complex and difficult to use. When users remove a monitor from the display configuration, they can lose windows as well. Also, different monitors might have different pixel densities; currently support is poor for dealing with such heterogeneity.

Our analysis results have begun to fill a research gap in real-world window management practices. The results have also influenced design choices for many prototypes and products in development. Our research prototypes address the first five of the six categories. The configuration problems are left for future research.

Losing the cursor

As screen size increases, people move their mouses faster and select higher mouse acceleration settings to quickly traverse the screen. The faster the mouse cursor moves, however, the more likely users are to lose track of it. In addition, as screen size increases, it becomes increasingly difficult to locate a stationary cursor.


(a)

(b)

1 High-density cursor versus mouse trail. Unlike the (a) similar mouse trail technology, the (b) high-density cursor uses temporal supersampling to bridge gaps in the cursor's position without lag.

High-density cursor

One key reason that users lose cursors during movement is that a cursor is rendered only once per frame, which makes it visually jump from one rendering position to the next, with the distance increasing with the cursor's speed. Our high-density cursor[3] addresses this using temporal supersampling. By filling the space between the cursor's current and previous position with additional fill-in cursor images, the high-density cursor bridges gaps between cursor positions. The effect is similar to increasing the display frame rate. Since all cursor images exist only for a single frame, the high-density cursor has no lag. As Figure 1 shows, this fact makes it different from similar techniques such as the Microsoft Windows mouse trail.

Auto-locator cursor

To help users find stationary cursors, Windows provides a locator function as part of the standard mouse properties. When enabled, this function shows the cursor location whenever users press the control key. When activated, an animated circle appears over the cursor, growing smaller over the course of a one second animation. This works well for a stationary cursor, but is less effective when the user has simply moved the cursor too fast and lost track of it. In that case, the user's hand is on the mouse rather than the keyboard, requiring an extra action.

To solve this problem, we developed an auto-locator cursor. Whenever the cursor quickly moves a long distance, it automatically invokes a similar, animated location indicator. Because this happens automatically only for fast, long-distance moves, it appears only when needed and requires no additional user action.

Distal access

As display size grows, it becomes increasingly difficult for users to access distant information. Consider a wall-sized display: If users stand near the lower right corner, for example, they must physically move to access a window or icon on the left, and might not be able to reach the upper display sections. This is an extreme case, but the problem also exists even with a more modest three-monitor configuration (triplemon). Accessing an icon or window at a distance requires moving the cursor a long distance, which takes time and raises cursor-tracking problems. Mixing monitor types exacerbates the problem. Consider a large touch-sensitive display like the SmartBoard, for example, combined with a smaller display with no touch input. It would be difficult—if not impossible—to drag an object from the touch surface onto the nontouch surface.

To solve these various problems, we developed four prototypes. Missile Mouse and Target Chooser let users move the cursor a long distance and select a distant window, respectively, using only small hand motions. Drag-and-Pop lets users drag-and-drop at a distance. Finally, Tablecloth lets users temporarily move and interact with desktop portions, then return them to their previous location.

Missile Mouse

To move the mouse across a large display takes time, much hand movement, and often requires users to clutch the mouse. As mentioned previously, when users compensate for distance using accelerated mouse settings, they often lose the cursor. Using Missile Mouse, users can "launch a missile" (the cursor) across the screen with a small movement. The cursor continues moving in the direction and at the speed of the original motion until the user moves the mouse again, reacquiring control. To activate Missile Mouse, users hold down the shift key while initiating mouse movement. With a little practice, users can probably move to any screen location with minimal hand motion (we have yet to formally test this).

A Missile Mouse variation is a wire-guided missile mouse. Users launch it in the same way, but direct the cursor's flight with small mouse motions (analogous to a wire-guided missile's directional guidance).

Target Chooser

Because large display users have more windows opened and unminimized, it's more difficult for them to select a window for interaction. The desired window might be obscured by other windows or residing at a distance (requiring time to acquire). Because of the number of windows involved, the standard Windows Alt-Tab solution no longer works well.

Target Chooser addresses this problem. Like Missile Mouse, it's initiated by holding down a modifier key (such as the shift key) while holding down the left mouse button and moving the mouse. In this case, the user's mouse movement casts a ray across the screen and tentatively selects the window centered closest to the ray. To highlight the target, Target Chooser draws visual cues on top of the window, even if it's obscured behind other windows. It also highlights the window's title. The user can then alter the selection using small mouse movements. Moving the mouse up or down selects other windows above or below the initial selection. Moving the mouse right or left will select windows to the right or left of the initial selection. Once the desired window is tentatively selected, the user releases the mouse button; this selects the window and places the cursor in it.

Drag-and-Pop

We designed Drag-and-Pop[4] to accelerate large-screen drag-and-drop interactions. By animating potential targets and bringing them to the dragged object, Drag-and-Pop reduces the user effort required to drag an object across the screen. Rather than moving target objects from their original location and inhibiting users' spatial memory, Drag-and-Pop stretches the object to the target location using a rubber band-like visualization (see Figure 2).

Tablecloth

Tablecloth gives users access to distant screen content by adapting the window scrolling notion to the entire desktop (see Figure 3). Users interact only with the area directly in front of them, which is conveniently accessible. To interact with all other content, users drag the desktop until the target content appears in the focus area—similar to pulling a Tablecloth to move a salt shaker across a table. To pan the screen, Tablecloth offers various incremental methods (dragging the screen, scrollbar, and hyperpanning) and absolute methods (target buttons with corresponding shortcuts). It also lets users drag objects between screen areas by invoking panning methods during drag interactions.

Bezel problems

Multimon display bezels present both opportunities and problems. The bezels' primary benefit is in letting users organize their work into different activities partitioned onto different monitors.[5] In fact, bezels can actually help users differentiate between different activities.

Problems occur when users need to cross bezels. If a window is too big to fit on a single monitor or is otherwise not carefully placed, it might span one or more bezels, creating a visual discontinuity at the bezel that makes reading text and perceiving image patterns difficult. In addition, when the cursor moves across a bezel, its path often appears deflected because there is no virtual space corresponding to the physical space that the bezel occupies.

To address these problems, we developed snapping and bumping, Mouse Ether, and OneSpace.

Snapping and bumping

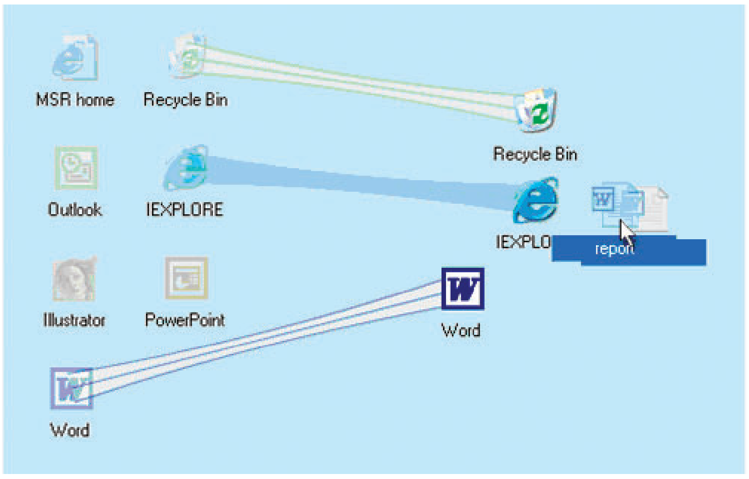
Larger displays offer opportunities for more complex window layout arrangements. With multimon displays, users try to avoid placing windows across bezels, but this effort is both time consuming and error prone. In addition, users can easily waste screen real estate in window layout. Two techniques—snapping and bumping—make these tasks easier.

Snapping lets users easily snap windows to bezels, other windows, or any display edge. Snapping is enabled whenever users move a window. As they drag a window edge near a bezel, window, or display edge, it snaps to that target. To unsnap it, users drag the window beyond the target. With snapping, users are much less likely to leave a window across a bezel.

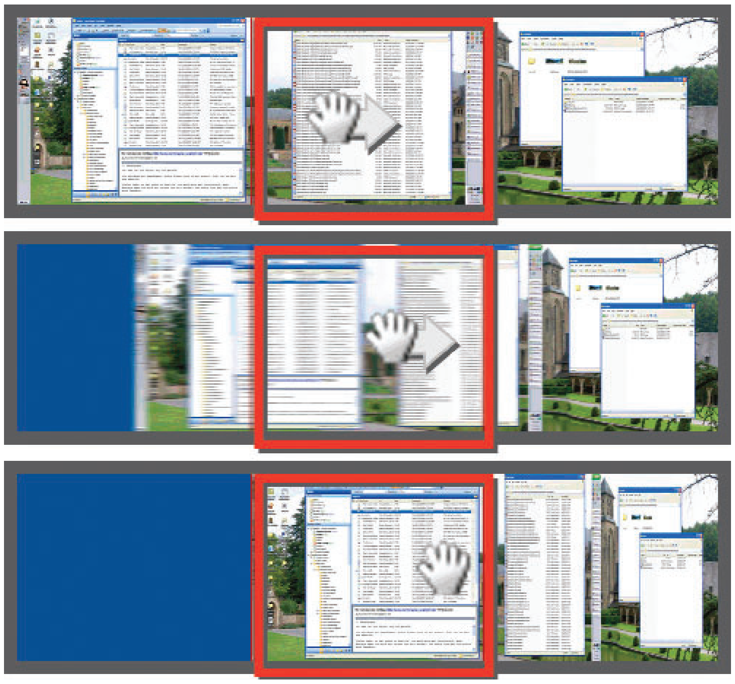
Bumping lets users move windows to other displays or to nearby empty spaces. Users first indicate the direction and type of bump, then the system finds the appropriate place to move the window. Because the user is not manually moving the window, it's easy to avoid leaving a window across a bezel.

Mouse Ether

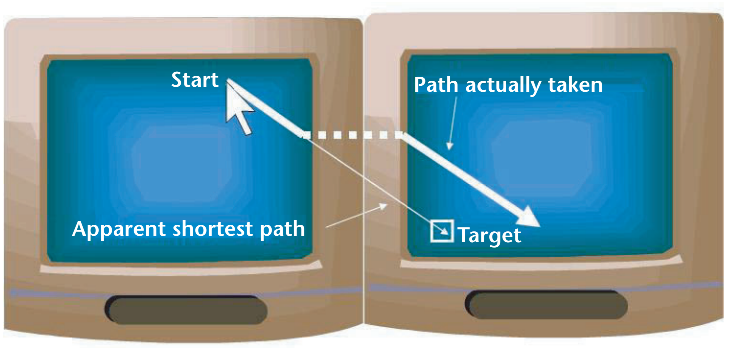
When acquiring a target located on a different screen, multimon users face a challenge: Differences in screen resolution, vertical offsets, and horizontal offsets warp the mouse pointer, making target acquisition difficult. Mouse Ether eliminates warping effects by applying appropriate transformations to all mouse move events.



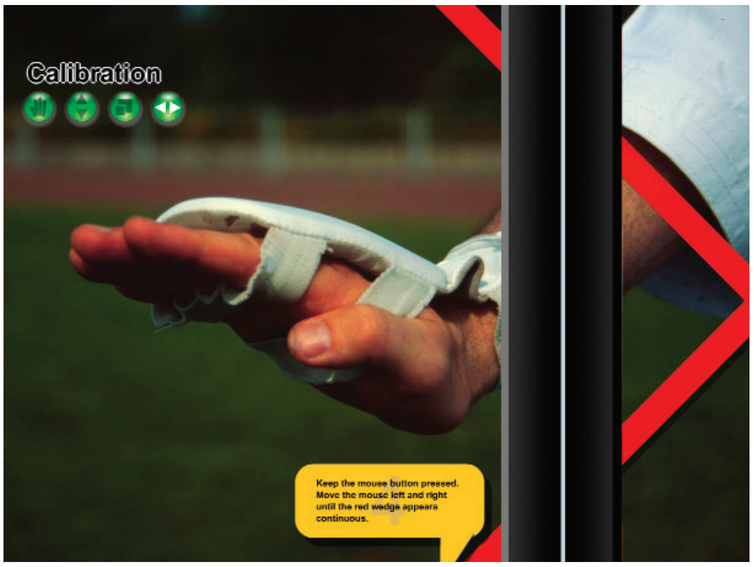
2 Drag-and-Pop interaction. To preserve users' spatial memory, the technique stretches the potential targets toward the object being dragged.



3 Tablecloth on a wall-sized display. The user accesses the displays left side while standing in the center.



4 **Mouse Ether showing the desired apparent shortest path.**


Calibration
Keep the mouse button pressed. Move the mouse left and right until the red wedge appears continuous.

5 **OneSpace adjusts the computer's geometric model to reflect the actual distance between monitors, eliminating image distortion across bezels.**

That is, as the cursor crosses a bezel, Mouse Ether moves the cursor so that it follows the perceived trajectory it was on in the first monitor (see Figure 4). In a user study, Mouse Ether improved participants' performance by up to 28 percent on a target acquisition task across two screens running at different resolutions. Seven of the eight participants also strongly preferred using Mouse Ether to the standard mouse.

OneSpace

Because a computer's geometrical model of the monitor setup doesn't account for monitor bezels, images displayed across multimons look distorted. OneSpace adjusts the computer's geometric model to reflect the actual physical distance between monitors on the user's desk. While this approach requires hiding image material located behind the bezels, it lets users view distortion-free images, as Figure 5 shows. We developed OneSpace for application on the content windows of image viewers, image-processing tools, background images, CAD tools, map viewers, or 3D games; control elements and other desktop applications would continue to run in the traditional clipping-free space.

Both Mouse Ether and OneSpace require a geometric calibration step to inform the system of the exact bezel placement. To do this, users adjust the red arrow in Figure 5 so that it appears aligned across the bezels. They need do this only once for a given multimon configuration.

Window management

Large displays pose various window management issues. When the user invokes an action that creates a dialog box or a new window, where should that window appear? If it's placed in an unexpected location far afield of the user's current focus, the user might not notice it and might assume that the system is broken. However, if the window appears directly in front of the current window or cursor position, it might obscure content that the user needs to see. While research continues on better context awareness for initial window placement, tools such as snapping and bumping, and GroupBar and Scalable Fabric (described later) can help users more efficiently manipulate windows and more easily reposition incorrectly placed windows.

Another problem multimon displays pose is the ambiguous nature of the maximize window operation. Does it mean maximize to fill the full screen or just the monitor where the window currently appears? If the user maximizes a window on the current monitor, it's difficult to move that window to another monitor. Currently, users must unmaximize the window, move it, and then remaximize it. Techniques like bumping or GroupBar maximize can also help here, as can our tristate maximize button: it first maximizes to the current monitor, then to the whole screen, and finally returns to the original size.

Start Anywhere

Task bar and start menu location also poses a large-display window-management problem, particularly on multimon displays. If there is a single task bar, the start menu might be far away from the user, making it difficult to invoke the menu's various actions.

To deal with this, we developed Start Anywhere, which lets users invoke the start menu from wherever the cursor is currently located. To invoke it, they can use the Windows key or a designated mouse or keyboard key; no mouse movement is required.

WinCuts

Each window on a computer desktop provides a view into some information. Although users can now manipulate multiple windows, being able to spatially arrange smaller window regions might help them perform certain tasks more efficiently. This is particularly important on large displays, where the target windows might be far apart. WinCuts[6] is a novel interaction technique that lets users replicate arbitrary regions of existing windows into independent windows. Each WinCut is a live view of a source window region that users can interact with. As Figure 6 shows, users can also share WinCuts across multiple devices.

To create a new WinCut, users hold down a keyboard modifier combination that brings up a semitransparent tint over the entire desktop. They then specify a rectangular region of interest (ROI) by clicking and dragging the mouse over the target window area. When they are satisfied with the ROI, they release the keyboard keys. The tint then disappears and a new WinCut appears over the source window, slightly offset from the ROI's location. The source window itself is unaffected. The WinCut is differentiated from regular windows by a green dotted line around its content region.

Users can create as many WinCuts as they wish, from one or more source windows. Each WinCut is a separate window and is managed much like a regular window. Unlike other windows, however, maximizing or resizing a WinCut preserves the relevant information, rescaling it within the WinCut. The user can thus make the information fill as much or as little space as they like.

WinCuts contain live representations of the source window's ROI content. Users can therefore view content updates through the WinCut and directly interact with WinCut content, just as they would in the original window. Also, because the WinCut view is tethered to the source window and is not an independent screen region, users can move and even hide the source window without affecting the WinCut. This flexibility lets users arrange ROIs near each other, regardless of the distance between the source windows in the larger display.

WinCuts bears some similarity to Virtual Network Computing.[7] VNC lets users share whole windows or the entire desktop using frame buffer contents, which makes sharing overlapped windows problematic. Because WinCuts lets users share window regions using off-screen rendering, it works correctly for overlapped windows.

Task management

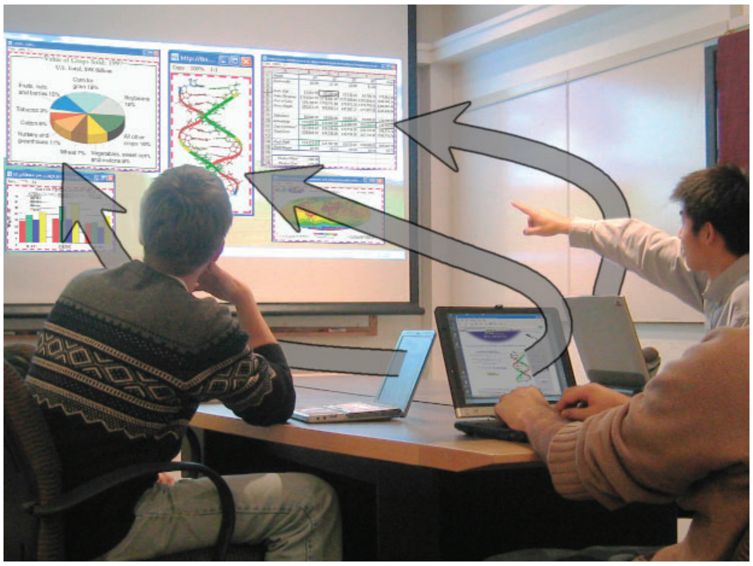
While working on high-level tasks, users need easy access to particular sets of windows and applications. An effective task management system should therefore provide convenient mechanisms that let users

- group relevant sets of windows,
- organize the groups and windows within the groups,
- switch between groups, and
- lay out the groups and windows on the screen.

On large displays, effective multiwindow task management is critical to a successful user experience.

Apple Exposé (http://www.apple.com/macosx/features/expose) is a window management system that assists users in dealing with multiple open windows. However, because Exposé is window- and application-based, rather than task-based, it fails to address the deeper question of how users can organize and manage tasks on large displays.

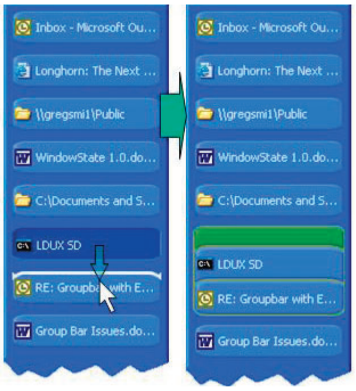
We have developed two systems that explore different facets of task management for large displays: GroupBar and Scalable Fabric. GroupBar adds new semantics for task organization and management to the existing Microsoft Windows task bar. Scalable Fabric uses scaling and a focus-in-context metaphor to visualize groups of related windows. In this system, all tasks are scaled and located in the periphery so they are simultaneously visible.



6 WinCuts at work. The technology sends portions of two personal laptops to a shared wall display.

GroupBar

We designed GroupBar[8] to provide task management features by extending the current Windows task bar metaphor. GroupBar preserves basic task bar tile functionality, presenting one tile for each open window in the system, and showing the currently active window tile in a darker, depressed-button state. Users can click on any tile to activate the corresponding window or minimize an active window. Further, GroupBar offers task management support by letting users drag and drop tiles representing open windows into high-level tasks called *groups* (see the green group tab in the lower right of Figure 7). The group control lets users switch between tasks with a single mouse click and perform window operations (minimize, maximize, close) on the entire group at once.

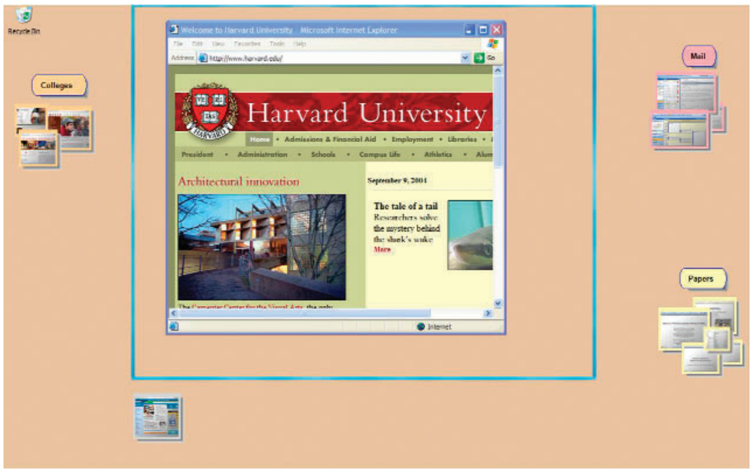


7 GroupBar technology. Dragging a window tile onto another tile combines both into a group.

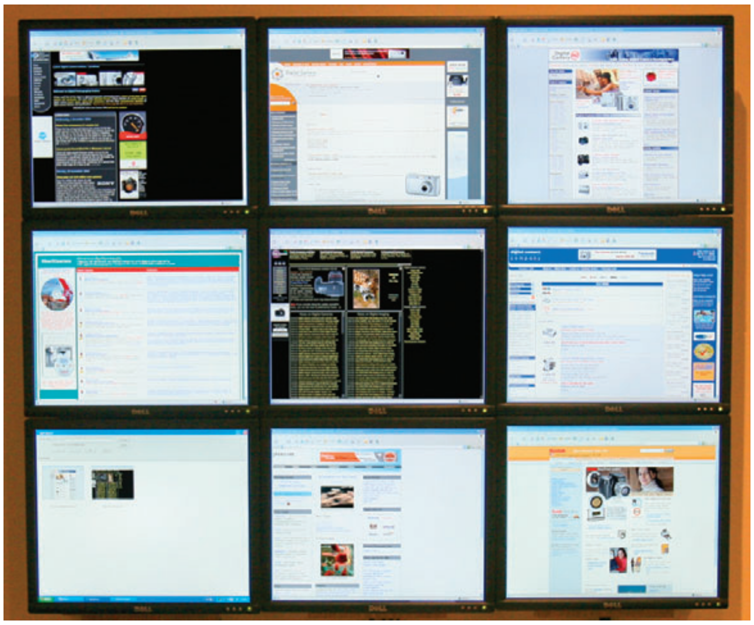
We designed GroupBar to let users group and regroup windows easily and quickly, and then operate on task or window groups as if they were a single unit. Our goal was to build on the existing task bar metaphor to improve the window management experience. To do this, we offer users a wider array of spatial arrangement preferences and a higher-level organizational structure (the group); we also extend existing window manipulation functions to the group level.

Scalable Fabric

Scalable Fabric[9] manages multiwindow tasks on the Windows desktop using a focus-plus-context display to allocate screen real estate in accordance with users' attention. Scalable Fabric lets users leave windows and window clusters open and visible at all times using a process that scales down windows and moves them



8 This Scalable Fabric example shows three task clusters in the periphery. By using the periphery, Scalable Fabric leverages the user's spatial memory and visual-recognition memory to aid task recognition.



9 SimulBrowser. The technology shows how large displays enhance everyday activities, such as Web searching. Here, SimulBrowser shows search results in eight monitors, and selected pages in the ninth (lower left).

to the periphery. Scalable Fabric is a focus-plus-context display because it gives users who are focused on a primary task the context of other work (that is, competing or potentially related tasks). It places this other work in the periphery, which leverages both the user's spatial memory and his or her visual-recognition memory for images to facilitate task recognition and location.

To use Scalable Fabric, the user defines a central focus area on the display surface by moving periphery boundary markers to desired locations (see Figure 8). The user's choice of focus-area location and size is influenced by the physical displays' configuration and capabilities. On a triplemon display, for example, users might define the central monitor as the focus area and use only the side monitors as the peripheral regions, with no upper or lower peripheral regions.

The focus-area windows behave like normal desktop windows. The periphery windows or tasks can be activated at any time. These windows are smaller, which lets users hold many tasks to the side while they are working on other things. Using this metaphor, users should rarely need to close or minimize windows and can take advantage of the extra screen real estate to keep peripheral windows visible.

When a user moves a window into the periphery, it shrinks monotonically with the distance from the focus-periphery boundary, getting smaller as it nears the screen's edge. When the user clicks on a peripheral window, it returns to its last focus position; this restore behavior appears as an animation of the window moving from one location to the other. When the user minimizes a focus-area window, it returns to its last peripheral position. These behaviors are similar to the management of ZoomScapes sheets,[10] but have been generalized to deal with windows and task management.

Designing applications for large displays

Some existing applications—such as power plant control, weather monitoring, financial systems, and software development environments—use large displays effectively. Each involves complex information presentations that are difficult to manage on small displays. The question we pose is whether everyday-use applications could effectively use and benefit from large displays.

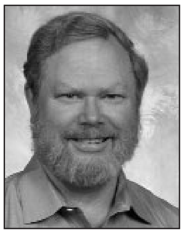
One example is SimulBrowser, an extension to Microsoft Internet Explorer that we designed specifically for large displays. SimulBrowser can be configured for any large or multimon display; users simply indicate how the space should be divided. Figure 9 shows a SimulBrowser running on a nine-monitor display, with Web search results for the query "digital cameras." The top eight results appear in eight of the monitors; the ninth holds a snapshots of page the user is interested in. That is, when the user marks a currently displayed page as interesting, a snapshot of it moves to the ninth monitor. The user can then advance to the next eight results, and so on. When the user wants to compare the selected pages, another operation replaces the search-result displays with the selected pages. SimulBrowser offers an example of how large displays might benefit users in everyday activities, but much work remains to explore the design space of such large-display-aware applications.

Conclusions

There is both a clear trend toward larger displays and mounting evidence that they increase user productivity and aid user recognition memory. However, as our user studies show, numerous usability problems inhibit the potential for even greater user productivity. Our research prototype techniques solve these usability problems, but we have not yet integrated them all into a single system. Nonetheless, solving these individual problems goes a long way toward improving the user experience on large displays. ■

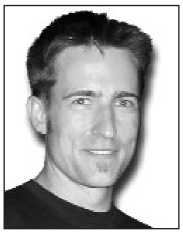
References

1. M. Czerwinski, "Toward Characterizing the Productivity Benefits of Very Large Displays," *Proc. of Interact 2003*, IOS Press, 2003, pp. 9-16.
2. D. Hutchings et al., "Display Space Usage and Window Management Operation Comparisons between Single Monitor and Multiple Monitor Users," *Proc. Working Conf. Advanced Visual Interfaces*, ACM Press, 2004, pp. 32-39.
3. P. Baudisch, E. Cutrell, and G. Robertson, "High-Density Cursor: A Visualization Technique that Helps Users Keep Track of Fast-Moving Mouse Cursors," *Proc. Interact 2003*, IOS Press, 2003, pp. 236-243.
4. P. Baudisch et al., "A Drag-and-Pop and Drag-and-Pick: Techniques for Accessing Remote Screen Content on Touch- and Pen-Operated Systems," *Proc. Interact 2003*, IOS Press, 2003, pp.57-64.
5. J. Grudin, "Partitioning Digital Worlds: Focal and Peripheral Awareness in Multiple Monitor Use," *Proc. Computer–Human Interaction*, ACM Press, 2002, pp. 458-465.
6. D.S. Tan, B. Meyers, and M. Czerwinski, "WinCuts: Manipulating Arbitrary Window Regions for More Effective Use of Screen Space," *Proc. Computer–Human Interaction*, ACM Press, 2004, pp. 1525-1528.
7. T. Richardson et al., "Virtual Network Computing," *IEEE Internet Computing*, vol. 2, no. 1, 1998, pp. 33-38.
8. G. Smith et al., "GroupBar: The TaskBar Evolved," *Proc. Australian Conf. Human–Computer Interaction,* University of Queensland, *2003*, pp. 34-43.
9. G. Robertson et al., "Scalable Fabric: Flexible Task Management," *Proc. Working Conf. Advanced Visual Interfaces*, ACM Press, 2004, pp. 85-89.
10. F. Guimbretiere, M. Stone, and T. Winograd, "Fluid Interaction with High-Resolution Wall-Size Displays," *Proc. Symp. User Interface and Software Tech.,* ACM Press, 2001, pp. 21-30.



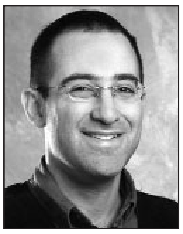
George Robertson *is a senior researcher in the Visualization and Interaction Research group at Microsoft Research. His research interests include information visualization, 3D user interfaces, and interaction techniques. Robertson has an MS in computer science from Carnegie Mellon University. He is a fellow of the ACM. Contact him at ggr@microsoft.com.*

Mary Czerwinski *is a senior researcher in the Visualization and Interaction Research group at Microsoft Research. Her research interests include attention and task switching, information visualization, and adaptive user interface design. Czerwinski has a PhD in cognitive psychology from Indiana University in Bloomington. Contact her at marycz@microsoft.com.*

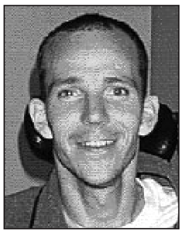


Patrick Baudisch *is a researcher in the Visualization and Interaction Research group at Microsoft Research. His research interests include interaction techniques for very large displays and visualization techniques for large documents on small screen devices. Baudisch has a PhD in computer science from Darmstadt University of Technology, Germany. Contact him at baudisch@microsoft.com.*

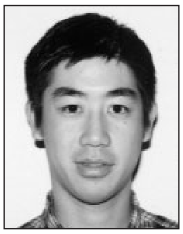
Brian Meyers *is a software design engineer in the Visualization and Interaction Research group at Microsoft Research. His research interests include interactions for ubiquitous computing. Meyers has a BS in computer science from the University of Puget Sound. Contact him at brianme@microsoft.com.*



Daniel Robbins *is a 3D user interface designer in the Visualization and Interaction Research group at Microsoft Research. His research interests include visual presentation of large information spaces and scalable interfaces. Robbins has a BA in visual arts from Brown University. Contact him at dcr@microsoft.com.*



Greg Smith *is a software design engineer in the Visualization and Interaction Research group at Microsoft Research. His research interests include human–computer interaction and data-driven visualization. Smith received a MS in computer science from Stanford University. Contact him at gregsmi@microsoft.com.*



Desney Tan *is a researcher in the Visualization and Interaction group at Microsoft Research. His research interests include building interfaces for large displays and computing environments with multiple devices. Tan has a PhD in computer science from Carnegie Mellon University. Contact him at desney@microsoft.com.*