# Large Display User Experience

**George Robertson, Mary Czerwinski, Patrick Baudisch, Brian Meyers, Daniel Robbins, Greg Smith, and Desney Tan**
Microsoft Research
One Microsoft Way
Redmond, WA 98052, USA
1-425-703-1527
{ggr, marycz, baudisch, brianme, dcr, gregsmi, desney} @Microsoft.com

**ABSTRACT**
As large displays become more affordable, researchers are investigating the effects on productivity, and techniques for making the large display user experience more effective. Recent work has demonstrated significant productivity benefits, but has also identified numerous usability issues that inhibit productivity. Studies show that larger displays enable users to create and manage many more windows, as well as to engage in more complex multitasking behavior. In this paper, we describe various usability issues, including losing track of the cursor, accessing windows and icons at a distance, dealing with bezels in multimonitor displays, window management, and task management. We also present several novel techniques that address these issues and make users more productive on large display systems.

**Author Keywords**
Large Displays, User Experience, Window Management, Task Management

**INTRODUCTION**
The increasing graphical processing power of the PC has fueled a powerful demand for larger and more capable display devices. Despite the increasing affordability and availability of larger displays, most users' display space represents less than 10% of their physical workspace area. Similarly, many of our current interfaces are designed assuming a relatively small display providing access to a much larger virtual world. How might users cope with and benefit from display devices that provide 25% to 35% of their physical desk area or perhaps one day cover an entire office wall? This question has been an issue of interest for many researchers [4, 5, 6, 8, 9]. To examine this issue, we evaluated usability issues for large displays and developed a series of research prototypes that address various issues we discovered.

**Figure 1. "DSharp" - A Wide Screen Multi-Projector Display**

Because the ability to work with multiple displays has been supported for some time in several operating systems (OS) and due to the advancements of graphic cards over the past ten years or so, a growing number of computer users take advantage of multiple monitor (multimon) capabilities. Our own survey research indicates that as many as 20% of the Windows™ OS "information worker" users today run multiple monitors from one PC or laptop. Most users are becoming aware that running multimon is an option. The two top reasons participants in our survey cited for not running multiple monitors included not having enough physical desktop space and having price concerns. Display manufacturers are predicting trends for the price of liquid crystal displays (LCDs), which have smaller desktop footprints, to drop dramatically over the next

four years. This price drop has already begun and the average computer consumer can now readily get 25% more pixels by buying dual 17″ LCDs than by buying one 21″ LCD for approximately the same price. Since all laptop manufacturers also are selling their products with built-in support for multiple monitors, we may see a dramatic increase in the number of users who will be opting for more screen real estate (pixels) by running multimon configurations, and we need to develop user interfaces that take advantage of this possibility.

Grudin [5] documents the usage patterns of CAD/CAM programmers and designers running multiple monitors. Despite the limitations observed in current OS support, multimon users clearly love the extra screen real estate, and they adapt their windows and application layouts optimally for the number, size, orientation and resolution of their displays. Most current multimon users claim they would never go back to a single monitor.

Czerwinski et al. [4] document a series of user studies demonstrating productivity benefits from using multimon or large displays with an eye toward novel software applications that might better support the way information workers multitask between their projects and applications. These studies showed a 12% significant performance benefit as well as a satisfaction preference for large displays.

While these studies demonstrate the advantages of using large displays, we have found in our work that there are also serious usability issues with current software systems. In the remainder of this paper, we describe basic usability issues as well as proposed solutions for key problems, including keeping track of the cursor, accessing distal windows and icons, dealing with bezels in multimon configurations, and managing windows and tasks.

There are two basic approaches to providing a large display experience. Wall-sized displays are often seamless display surfaces created using multiple tiled projectors. These systems tend to use touch or pen input. Large desktop displays are often multiple monitor configurations with seams between the monitors caused by the monitor bezels. These systems tend to use keyboard and mouse for input. Most of the usability issues and solutions presented here are relevant for both approaches. The seams in multiple monitor configurations offer additional challenges and opportunities.

## COGNITIVE BENEFITS OF LARGE DISPLAYS

In addition to productivity benefits mentioned earlier, Czerwinski et al. [4] also document results showing that larger displays lead to improved recognition memory and peripheral awareness.

Tan et al. [13] report a series of studies demonstrating the advantages of large displays on 3D navigation in virtual worlds. They show that while large displays increase performance for all users on average, females improved so much that the normal advantage male users have over females in virtual 3D navigation disappears when using large displays. These studies reveal that the wider fields of view provided by large displays lead to increased ability to process optical flow cues during navigation, cues that females are more reliant upon than males. They ran their studies on DSharp, a seamless wide screen multi-projector display shown in Figure 1, and found that the optimal field of view for the tasks tested was about 100 degrees, the equivalent of a triple monitor display. Thus, large displays actually serve to eliminate a gender bias, at least for the set of tasks tested.

## LARGE DISPLAY USABILITY ISSUES

Several of the usability issues we have identified and describe below were observed while evaluating the productivity benefits of large displays using formal laboratory studies [4]. We also identified usability issues by developing and deploying windowing system logging tools to observe real Windows users in the field, and by analyzing product support calls.

### Gathering Data

It is difficult to adequately design for large displays and multiple monitor systems without understanding how multimon users differ from, or are similar to, single monitor users. Therefore, we deployed a tool, called *VibeLog* [7], to a group of single monitor and multimon users to log window management activity. We focused on window management activity in order to discover higher-level patterns of activity for different sized displays (e.g., number of opened windows, frequency of window activation, and frequency of window movement). Analysis of the data collected from this tool revealed that usage of interaction components may change with an increase in number of monitors and that *window visibility* can be a useful measure of user display space management activity, especially for multiple monitor users. The results from this analysis

begin to fill a gap in research about real-world window management practices, and have influenced the design choices for many of the prototypes discussed in this paper, as well as products in development.

### Basic Usability Issues

The usability issues we have observed fall into six broad categories:

1. Losing track of the cursor. As screen size increases, users change mouse acceleration to compensate and it becomes hard to keep track of where the cursor is.

2. Bezel problems with multimon displays. Bezels introduce visual distortion when windows cross them and interaction distortion as the cursor crosses them.

3. Distal access to information. As screen size increases, it becomes increasingly more difficult and time-consuming to access icons, windows, and the Start Menu across large distances.

4. Window management problems. Large displays lead to notification and window creation problems, as windows and dialog boxes pop up in unexpected places. Window management is made more complex on multimon displays because users wish to avoid having windows placed so that they cross bezels.

5. Task management problems. As screen size increases, the number of windows that are open increases and users engage in more complex multitasking behavior – better task management mechanisms become a necessity.

6. Configuration problems. The user interface for configuring multimon displays is overly complex and hard to use. When a monitor is removed from the display configuration, it is possible to lose windows off-screen. Also, different monitors may have different pixel densities and there is currently poor support for dealing properly with this sort of heterogeneity.

In the sections that follow, we will describe research prototypes that address the problems raised by the first five of these categories of usability issues.

## WHERE IS MY CURSOR?

As screen size increases, faster mouse movement and higher mouse acceleration settings are used in order to traverse the screen from side to side reasonably fast. 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.

### High Density Cursor

One key reason the cursor is lost during movement is that the 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. *High Density Cursor* [3] addresses this issue by using temporal supersampling. By filling the space between the current cursor position and the previous one with additional fill-in cursor images, the high density cursor bridges the gaps between cursor positions, resulting in an effect similar to increasing the display frame rate. Since all cursor images exist only for a single frame, the high density cursor does not introduce any lag, which makes it different from similar-looking techniques, such as the Microsoft Windows mouse trail, as shown in Figure 2.

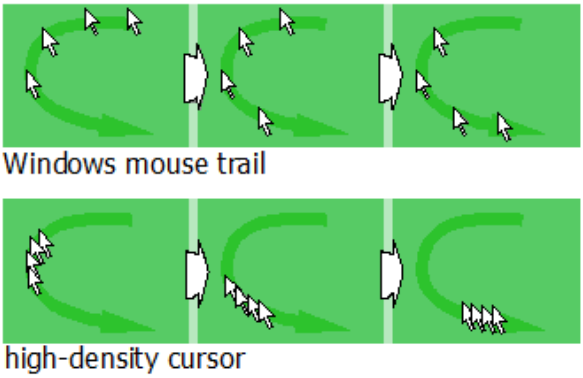


**Figure 2. High density cursor compared to Windows mouse trail.**

### Auto-Locator Cursor

To handle the problem of finding a stationary cursor, Windows provides a locator function as part of the standard Mouse Properties. If this function is enabled, it will show the cursor location whenever the control key is pressed. It shows the location using an animated circle centered on the cursor, starting large and shrinking over the course of a one second animation. While this works well for a stationary cursor, it works less well for a cursor that the user simply moved too fast and lost track of. In that case, the user's hand is on the mouse rather than the keyboard, so an extra action is required.

To solve this problem we introduce an *Auto-Locator Cursor* . Whenever the cursor moves a long distance at a rapid speed, it automatically invokes a similar animated location indicator. Since this happens automatically only for fast and long-distance moves, it only appears when it is needed and requires no additional action on the part of the user.

## DISTAL ACCESS

As display size increases, it becomes increasingly more difficult to access information at a distance. In the extreme case, consider a wall-sized display with the user standing near the lower right corner. To interact with a window or icon in the upper left corner, the user must physically move to the far side of the display and may not be physically able to reach the upper parts of the display. But, this problem exists even in the case of a more modest three monitor configuration (triplemon). Accessing an icon or window at a distance requires moving the cursor a long distance, which suffers from the problems of losing the cursor mentioned in the previous section and takes time. The problem can also be exacerbated by mixed monitor types. Consider, for example, a large touch sensitive display like the SmartBoard combined with a smaller display which has no touch input. It would be difficult, or impossible, to drag an object from the touch surface onto the non-touch surface.

We have developed four prototypes that solve these various problems. The *Missile Mouse* provides a way to move the cursor a long distance with only a small hand motion. The *Target Chooser* provides a way to select a window from a distance with very small hand movements. *Drag-and-pop* is a technique for doing drag and drop actions at a distance. And *Tablecloth* is a technique that allows the user to temporarily move portions of the desktop to the user for interaction, and then return them to their normal location.

### Missile Mouse

Moving the mouse a long distance on a large display takes time and much hand movement, often requiring mouse clutching. As mentioned above, compensatory higher mouse acceleration settings lead to the user losing track of the cursor. Missile Mouse is an alternative technique that allows a small mouse movement to "launch a missile" (the cursor) across the screen. The cursor continues moving in the direction and at the speed of the original motion until the user reacquires control of the cursor by moving the mouse again. The Missile Mouse behavior is enabled by holding down the shift key while initiating mouse movement. With a little practice, we believe it is possible to move to any location on the screen with minimal hand motion (although this has not been formally tested yet).

A variation of the Missile Mouse is a "wire-guided" missile mouse. This version is launched in the same way, but as the cursor is in flight the user can control its direction of movement with small motions of the mouse, analogous to the directional guiding done with a wire-guided missile.

### Target Chooser

We know that larger displays lead to more windows being open and un-minimized. This makes it more difficult to select a window for interaction. The desired window may be at a distance, requiring time to acquire, or may be obscured by other windows. The standard Windows Alt-Tab solution no longer works well because of the number of windows involved.

Target Chooser is a technique that addresses this problem. Like Missile Mouse, it is initiated by holding down a modifier key (like the shift key) while holding down the left mouse button and moving the mouse. In this case, the mouse movement casts a ray across the screen and tentatively selects the window whose center lies closest to the cast ray. That window is highlighted by drawing a few visual cues on top of the window. This highlighting is performed even if the window is obscured behind other windows, and includes showing the window title. The user can then use small mouse movements to alter the tentative selection. Moving the mouse up or down will select other windows above or below the initial selection. Moving the mouse right or left will select other windows to the right or left of the initial selection. Once the desired window is tentatively selected, the user releases the mouse button and the window is selected and the cursor is placed in that window.

### Drag-and-pop

Drag-and-pop [2] is a technique designed to accelerate drag-and-drop interactions on large screens. By animating potential targets and bringing them to the dragged object, drag-and-pop reduces the user effort required for dragging an object across the screen to a desired target. To preserve users' spatial memory, targets are not moved away from their original location, but are

instead stretched using a rubber band-like visualization, as shown in Figure 3.

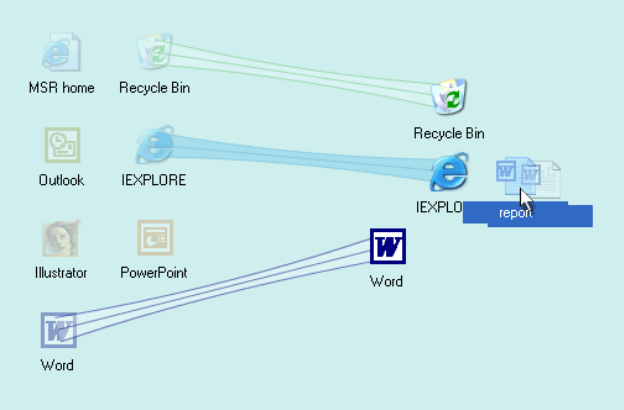


**Figure 3. Drag-and-pop showing relevant icons rubber-banded toward object being dragged.**

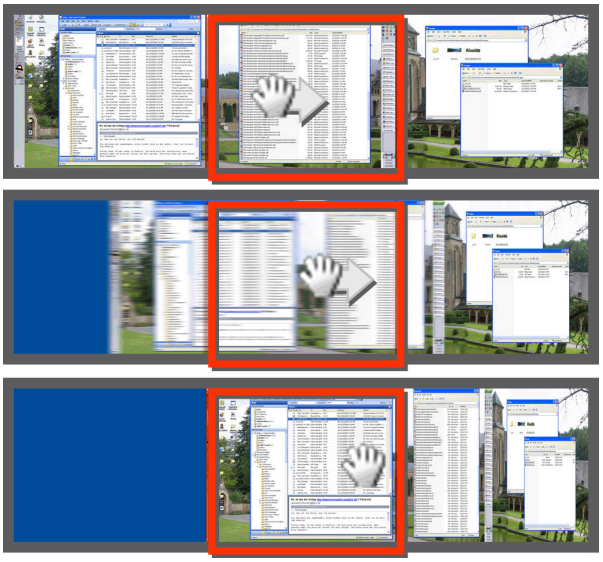

**Figure 4. Tablecloth on a wall-sized display -- user is standing in center and accessing left part of display.**

### Tablecloth

*Tablecloth* provides users with access to distant screen content by adapting the notion of window scrolling to the desktop as a whole. Users interact only with the area right in front of them, which is conveniently accessible. To interact with all other content, Tablecloth allows users to scroll the screen from within the focus area until that content appears in the focus area, similar to pulling a tablecloth in order to access a salt shaker at the other end of the table. Tablecloth offers a variety of incremental (dragging the screen, scrollbar, "hyper panning") and absolute (target buttons with corresponding shortcuts) methods for panning the screen. It also allows users to drag objects between screen areas by invoking panning methods during drag interactions.

## WHAT ABOUT THE BEZELS?

The bezels in a multimon display present both opportunities and problems. The primary opportunity is for users to organize their work into different activities that are partitioned onto the different monitors [5]. In this case, the bezels may actually help to differentiate between these different activities.

The problems occur when crossing bezels. If a window is too big to fit on a single monitor, or if a window is not carefully placed, a window may lie across one or more bezels. At the points where a window crosses a bezel, there is a visual discontinuity that makes it hard to read text or correctly perceive patterns in an image. In addition, when the cursor is moved across a bezel, it often appears to be deflected in its path, because there is no virtual space corresponding to the physical space occupied by the bezel.

We have developed several techniques to address these problems. *Snapping* and *Bumping* are used to keep windows from accidentally being placed across a bezel. *Mouse Ether* is a technique for compensating for the apparent warping of the cursor as it is moved across a bezel. And *OneSpace* is a technique that compensates for distortions that occur in images that cross bezels.

### Snapping and Bumping

Larger displays offer opportunities for more complex window layout arrangements. In the case of multimon displays, users try to avoid placing windows so that they cross bezels, but this is a time consuming and error prone activity. In addition, it is easy to waste screen real estate during window layout activity. We introduce two techniques for making these tasks easier.

*Snapping* is a technique that allows a user to easily snap a window to a bezel, another window, or any display edge. Snapping is enabled whenever a user is moving a window. As an edge of the window being dragged nears a bezel, window, or display edge, it will snap to that target. The user can drag beyond the target to unsnap it. This technique makes it much easier to avoid leaving a window across a bezel.

*Bumping* is a technique that allows a user to move a window to another display or to nearby empty space. The user indicates the direction and type of bump and the system finds the appropriate place to move the window. Since the user is not manually moving the window, leaving a window across a bezel is easily avoided.

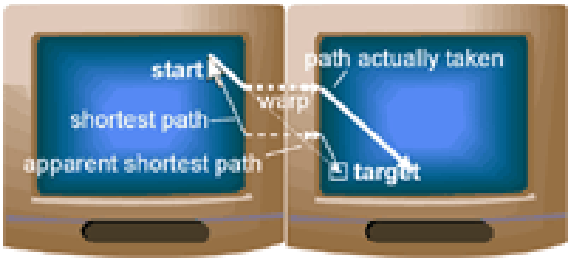


**Figure 5. Mouse Ether showing the desired apparent shortest path.**

### Mouse Ether

When acquiring a target located on a different screen, multimon users face a challenge: differences in resolution, vertical offsets, and horizontal offsets between screens cause the mouse pointer to get warped, making the attempt to acquire the target difficult. Mouse Ether eliminates warping effects by applying appropriate transformations to all mouse move events. 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, as shown in Figure 5. A user study demonstrated that mouse ether improved participants' performance on a target acquisition task across two screens running at different resolutions by up to 28%. Seven of the eight participants also strongly preferred using Mouse Ether.

### OneSpace

Since the computer's geometrical model of the monitor setup does not take the monitors' bezels into account, images displayed across multiple monitors look distorted. OneSpace addresses this issue by making the computer's geometric model reflect the actual physical distance between monitors on the user's desk. While this approach requires hiding image material located "behind" the bezels, it allows users to view image material distortion free, as shown in Figure 6. OneSpace is meant to be applied selectively to the content windows of image viewers, image processing tools, background images, CAD tools, map viewers, or 3D-games; control elements, as well as other applications running on the same desktop, would continue to run in the traditional clipping-free space.

Mouse Ether and OneSpace both require a geometric calibration step to inform the system of the exact placement of the bezels. As part of this configuration step, the red arrow in Figure 6 is adjusted so that it appears aligned across the bezels. This need be done only once for a given configuration of multiple monitors.


**Figure 6. OneSpace used to eliminate distortion in image across a bezel.**

## WINDOW MANAGEMENT

Large displays pose a variety of window management issues. When the user invokes an action that creates a dialog box or a new window, where does that window get placed? If the window or dialog box is placed in an unexpected location at a distance from the user's current focus of attention, the user may not even notice it and may assume that the system is broken in some way. On the other hand, if the window is placed directly in front of the current window or cursor position, it may obscure content that the user wished to keep visible as part of the current multi-window activity. While research continues on better context-awareness for initial window placement, tools such as Snapping and Bumping (described earlier) and GroupBar and Scalable Fabric (described below) can help to make overall window manipulation more efficient and make re-positioning incorrectly-placed windows easier.

Another problem posed by multimon displays is the ambiguous nature of the "maximize" window operation. Does it mean maximize to the full screen size or to the monitor the window is currently on? If the user chooses to maximize on the current monitor, then there is an additional problem if the user wants to move the maximized window to another display. Currently, the user must un-maximize the window, move it, then re-maximize it. This is another case where techniques like Bumping or GroupBar Maximize can simplify window management on large displays. We have also built an alternative to the maximize button that is tri-state; it first maximizes to the current monitor, then to the whole screen, and finally returns to the original size.

#### Start Anywhere

Another problem with window management on large displays, particularly multimon displays, is where the TaskBar and Start Menu appear. If there is a single TaskBar, then the Start Menu may be quite a distance away from the user, making it difficult to invoke a variety of actions available from the Start Menu.

We introduce an alternative technique, called *Start Anywhere*, which allows the Start Menu to be invoked wherever the cursor is currently located, using the Windows key or some designated mouse or keyboard key. Using this technique, no mouse movement is required to invoke the Start Menu.

#### WinCuts

Each window on a computer desktop provides a view into some information. Although users can currently manipulate multiple windows, we assert that being able to spatially arrange smaller regions of these windows could help users perform certain tasks more efficiently. This is particularly important on large displays where the windows in question might be quite far from each other. *WinCuts* [14] is a novel interaction technique that allows users to replicate arbitrary regions of existing windows into independent windows. Each WinCut is a live view of a region of the source window with which users can interact. Users can also share WinCuts across multiple devices, as shown in Figure 7.

To create a new WinCut, users hold down a keyboard modifier combination which brings up a semi-transparent tint over the entire desktop. They then click and drag the mouse over a region of a window to specify a rectangular region of interest (ROI). When they are satisfied with the ROI, they release the keyboard keys. The tint disappears and a new WinCut appears on top of the source window, slightly offset from the location of the ROI. The source window is unaffected. The WinCut is differentiated from regular windows by a green dotted line around the content region of the WinCut. Users may make as many WinCuts as they wish, either from a single source window, or from multiple windows.

**Figure 7. WinCuts used to send portions of two personal laptops to a shared large display.**

Each WinCut is a separate window and can be managed much like a regular window. Unlike other windows, however, maximizing or resizing a WinCut preserves the relevant information that is shown and instead rescales the content within the WinCut. This allows the user to make the information fill as little or as much space as they would like.

WinCuts contain live representations of the content that appears within the ROI on the source window. In other words, the user can not only view updating content from the source window through the WinCut, but can also directly interact with content in it, just as they would in the original window. Since this view is tethered to the source window and not a region of the screen, users can move and even hide the source window without affecting the WinCut. This flexibility makes it possible to arrange regions of interest near each other, in spite of the fact that the source windows may be quite far apart on a large display.

WinCuts bears some similarity to VNC [10]. VNC is used to share either whole windows or the entire desktop and is based on the contents of the frame buffer; hence sharing overlapped windows is problematic.

WinCuts, on the other hand, allows the sharing of regions of windows and uses off-screen rendering, hence will work correctly for overlapped windows.

## TASK MANAGEMENT

While working on their high-level tasks, users need easy access to particular sets of windows and applications that contain relevant information. Hence, we assert that an effective task management system should provide convenient mechanisms for users to group relevant sets of windows, to organize the groups and windows within the groups, to switch between groups, and to lay out the groups and windows on the screen. Effective multi-window task management is critical to a successful user experience on large displays.

Apple Exposé [1] is a window management system that provides some assistance for dealing with large numbers of opened windows. However, it is window and application based, rather than task-based. Hence, it does not address the deeper question of how a user can organize and manage tasks on large displays.

In this section, we present two systems that explore different facets of task management for large displays: *GroupBar* and *Scalable Fabric*. GroupBar adds new semantics to the existing Microsoft Windows Taskbar for organizing and managing tasks. 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 that they are simultaneously visible.

### GroupBar

We designed GroupBar [12] with the goal of providing task management features by extending the current Windows Taskbar metaphor. GroupBar preserves basic Taskbar 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 to minimize the window if it is already active. Going beyond current Taskbar functionality to offer task management support, GroupBar allows users to drag and drop tiles that represent open windows into high-level tasks called 'Groups', represented by the green group tab shown in Figure 8. This group control allows users to switch between tasks with a single mouse click and perform window operations (minimize, maximize, close) on the entire group at once.

With GroupBar we wanted to allow users to group and regroup windows easily and quickly, and then allow them to operate on groups of windows (or tasks) as though they were a single unit. We thought that by incorporating a wider array of spatial arrangement preferences, offering users a higher-level organizational structure (the group), and extending existing window manipulation functions to the group level, we could design an improved window management experience that built upon the existing Taskbar metaphor.

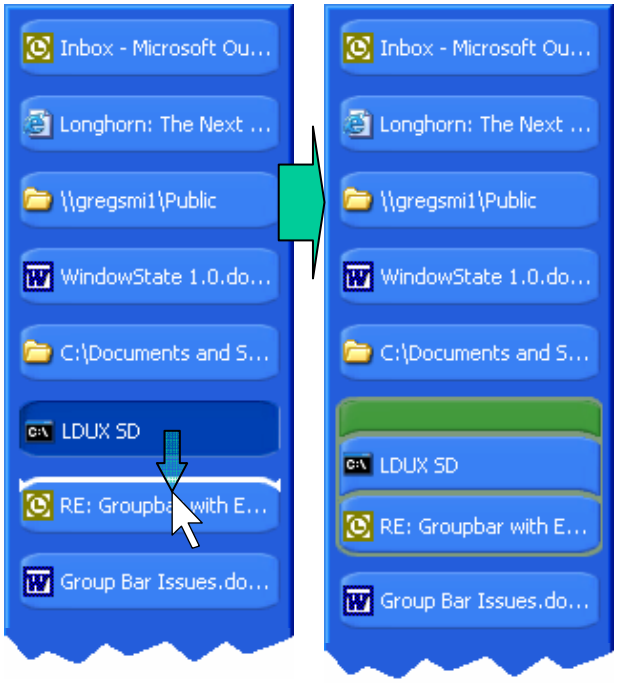


**Figure 8: Dragging a window tile onto another tile combines both into a "Group".**

### Scalable Fabric

Scalable Fabric [11] is a system based on managing multi-window "tasks" on the Windows desktop, this time using a focus-plus-context display to allocate screen real estate in accordance with users' attention. Scalable Fabric allows users to leave windows and clusters of windows open and visible at all times via a process of scaling down and moving the windows to the periphery. Scalable Fabric is a focus-plus-context display in the sense that users focusing their attention on a primary task are provided with the context of other work (i.e., competing or potentially related tasks)

displayed in their periphery. This use of the periphery leverages both the user's spatial memory and also their visual recognition memory for images in order to facilitate task recognition and location.


**Figure 9. Scalable Fabric showing three task clusters in the periphery.**

In Scalable Fabric, the user defines a central focus area on the display surface by moving periphery boundary markers to desired locations (see Figure 9). The user's choice of focus area location and size is influenced by the configuration and capabilities of the physical displays. For example, on a triplemon display, users may prefer to define the central monitor as the focus area, having no upper or lower peripheral regions and using the side monitors as the only peripheral regions.

Within the focus area, windows behave as they normally do on the Windows desktop. The periphery contains windows and clusters of windows, or tasks, which are not currently in use, but may be put to use at any time. Windows in the periphery are smaller so that more tasks can be held there when the user is focusing on something else. With this metaphor, we believe users will rarely need to close or minimize windows in the traditional sense. Users can take advantage of extra screen real estate to allow the peripheral windows to always be 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 edge of the screen. When the user clicks on a window in the periphery, it returns to its last focus position; this is the new 'restore' behavior, and is accomplished with an animation of the window moving from one location to the other. When the user 'minimizes' a window in the focus area by clicking the window's 'minimize' button, it returns to its last peripheral position. These behaviors are similar to the management of sheets in ZoomScapes [6], but have been generalized to deal with windows and task management.

## APPLICATION USE OF LARGE DISPLAYS

In the preceding sections, we described a series of large display usability issues and prototype solutions to each of these problems. What remains is to consider how applications might be designed to fully utilize large displays. There are existing applications that effectively use large displays, like power plant control, weather monitoring, financial systems, and software development environments. Each of these cases involves complex information presentations that are not easily done on small displays.

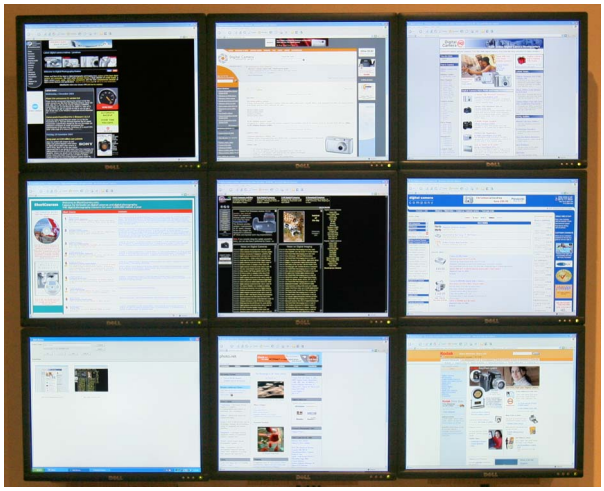

**Figure 10. SimulBrowser on a nine monitor display, with the selected pages on the lower left monitor.**

The question we pose is whether there are everyday use applications that could effectively use large displays. One example that we developed is *SimulBrowser*, an extension to Internet Explorer designed specifically for large displays. Figure 10 shows an example of SimulBrowser running on a nine monitor display showing the results of a search for "digital cameras". This can be configured for any multimon display or any large display (simply by indicating how the space should be divided). When the user does a web search, the top eight results are displayed in eight of the monitors. The ninth monitor is used to hold snapshots of selected pages of interest. That is, the user can mark any of the currently displayed pages as interesting and the snapshot(s) will be moved 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 will replace the search results displays with the selected pages. SimulBrowser is an example of an everyday use application that has been designed or tuned for large displays. Much work remains to be done to explore the design space of such large-display-aware applications.

## CONCLUSIONS

There is a clear trend in the industry toward larger displays, either as a single display surface or in multiple monitor configurations. There is also evidence that larger displays increase user productivity, aid user recognition memory, and in some cases can eliminate gender bias (e.g., in 3D virtual environment navigation). User studies have identified numerous usability problems that inhibit even better user productivity, including: keeping track of the cursor, distal access to windows and icons, dealing with bezels, window management, and task management. In this paper, we have presented a series of research prototypes that demonstrate techniques that solve each of these problems. The work of integrating these various prototype solutions into one system remains to be done. Correcting these problems goes a long way toward improving the user experience on large displays.

## REFERENCES


1. Apple Exposé, http://www.apple.com/macosx/features/expose/.
2. Baudisch, P., Cutrell, E., Robbins, D., Czerwinski, M., Tandler, P., Bederson, B., and Zierlinger, A. Drag-and-Pop and Drag-and-Pick: techniques for accessing remote screen content on touch- and pen-operated systems. In *Proceedings of Interact 2003*, pp.57-64.
3. Baudisch, P. Cutrell, E, and Robertson, G. High-density cursor: a visualization technique that helps users keep track of fast-moving mouse cursors. In *Proceedings of Interact 2003,* pp. 236-243.
4. Czerwinski, M., Smith, G., Regan, T., Meyers, B., Robertson, G. and Starkweather, G. (2003). Toward characterizing the productivity benefits of very large displays. In *Proceedings of Interact 2003*, pp. 9-16.
5. Grudin, J. (2002). Partitioning digital worlds: Focal and peripheral awareness in multiple monitor use. In *Proceedings of CHI 2002,* pp. 458-465.
6. Guimbretiere, F., Stone, M., and Winograd, T. (2001). Fluid interaction with high-resolution wall-size displays. In *Proceedings of UIST 2001,* pp. 21-30.
7. Hutchings, D., Czerwinski, M., Smith, G., Meyers, B., Robertson, G Display space usage and window management operation comparisons between single monitor and multiple monitor users. In *Proceedings of AVI 2004.*
8. MacIntyre, B., Mynatt, E., Voida, S., Hansen, K., Tullio, J., Corso, G. (2001). Support for multitasking and background awareness using interactive peripheral displays. In *Proceedings of UIST 2001,* pp. 41-50.
9. Mynatt, E., Igarashi, T., Edwards, W., and LaMarca, A. (1999). Flatland: new dimensions in office whiteboards. In *Proceedings of CHI 1999,* pp. 346-353.
10. Richardson, T., Stafford-Fraser, Q., Wood, K., and Hopper, A., Virtual Network Computing, in IEEE Internet Computing, vol. 2, no. 1, January, 1998.
11. Robertson, G., Horvitz, E., Czerwinski, M., Baudisch, P., Hutchings, D., Meyers, B., Robbins, D., and Smith, G. (2004), Scalable Fabric: Flexible Task Management. In *Proceedings of AVI 2004* pp. 85-89.
12. Smith, G., Baudisch, P., Robertson, G., Czerwinski, M., Meyers, B., Robbins, D., and Andrews, D. (2003). GroupBar: The TaskBar Evolved. In *Proceedings of OZCHI 2003*, pp. 34-43.
13. Tan, D., Czerwinski, M., and Robertson, G. (2003). Women Go With the (Optical) Flow. In Proceedings of CHI 2003, pp. 209-215.
14. Tan, D.S., Meyers, B. & Czerwinski, M. (2004). WinCuts: Manipulating arbitrary window regions for more effective use of screen space. In *Extended Abstracts of Proceedings of CHI 2004*, pp. 1525-1528.

***George Robertson*** *is an ACM Fellow and 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. He received an M.S. in Computer Science from Carnegie Mellon University. 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 span the areas of attention and task switching, information visualization and adaptive user interface design. Mary received her 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 focus is on interaction techniques for very large displays and visualization techniques for large documents on small screen devices. He holds 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 interests are interactions for ubiquitous computing. He has a B.S. 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 current projects include visual presentation of large information spaces and scalable interfaces. Dan has a visual-arts degree 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. Greg received his M.S. 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 center on building interfaces for large displays as well as for computing environments consisting of multiple devices. Desney has a Ph.D. in Computer Science from Carnegie Mellon University. Contact him at desney@microsoft.com.*