# Evaluating Two Window Manipulation Techniques on a Large Screen Display

Russell Mackenzie, Kirstie Hawkey, Presley Perswain, Kellogg S. Booth
Dept. of Computer Science
The University of British Columbia
{rmacken1, khawkey, perswain, ksbooth}@cs.ubc.ca

**ABSTRACT**
Large screen displays are a common feature of modern meeting rooms, conference halls, and classrooms. The large size and often high resolution of these displays make them inherently suitable for collaborative work, but these attributes cause traditional windowing systems to become difficult to use because the interaction handles become smaller in visual space and in motor space. This may be exacerbated when a user faces the additional cognitive load of active, real-time collaboration. We describe a new window manipulation technique for such a collaborative meeting environment. Its design was inspired by recent collaborative systems in which a user must explicitly take control of a window in order to interact with its contents; actions are otherwise interpreted as navigational. Our Large Screen Optimized (LSO) window manipulation technique utilizes the entire window for manipulations instead of only the title-bar and borders. In addition, LSO includes 'snapping regions' that automatically move the cursor to the boundary of the window, allowing quick, accurate manipulations involving the edges and corners of the screen. We experimentally validated that our new technique allows users to move and resize windows more quickly than with a traditional window manipulation technique.

**Author Keywords**
Large screen display, window manipulation.

**INDEX TERMS:** H5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

## 1 INTRODUCTION

Display technology continues to make rapid advances, providing ever larger physical sizes, greater resolutions, and higher contrast at ever lower costs. It is now feasible to have multiple high-definition displays available in even small meeting rooms, and it is plausible that wall-size displays will soon become commonplace. Such displays can improve performance on navigation, search, and comparison tasks in information rich virtual environments [21], and are inherently well-suited for collaborative activities. However, it is often difficult to switch which user controls the display, and there is no broad support for multiple simultaneous users. While individually small, these technical hurdles disrupt collaborative activities and prevent effective use of available displays.

Research into collocated collaborative meeting systems for single display groupware (SDG) and multi-display environments is ongoing. Systems such as WinCuts [24], Integrated Tabletop [20], IMPROMPTU [3], WeSpace [25], and LACOME [17] allow multiple users to share content from their personal computers on public displays and to allow others to interact with that content. Window manipulation typically occurs only when a user drags the margin of the windows (title bar, border) as in traditional window management systems. One exception is LACOME [17], which has multiple control layers to better support differentiated control modes, so that manipulating windows (e.g., resizing, moving, iconifying) is distinct from interacting with content within published windows. Because a mode switch is required in order to interact with the content within a window, such a system is able to support alternative window manipulation techniques.

Traditional windowing interfaces such as those of Microsoft (MS) Windows or Mac OS X may not be as usable on a large display as they are on personal displays. Swaminathan and Sato [23] found that when a display exceeds a certain size, it becomes "qualitatively different." Larger displays enable users to create and manage many more windows [10, 2], as well as to engage in more complex multitasking behaviors [10]. A recent week-long study [2] comparing usage of a large display to single and dual monitor configurations for daily work suggests the need for different window management techniques for a large wall-size display. Users of the wall-size display performed more moving and resizing operations than in their normal environments. They also had difficulty accurately selecting the corner of an application window for resizing, particularly when the window was located in the peripheral region of the screen. Collocated meetings may provide additional challenges for large screen display use; users may face multiple cognitive demands simultaneously and are rarely in the ideal position to view a shared display. Moreover, when multiple users with multiple cursors interact on shared displays, larger target sizes may ease selection [18].

We are interested in exploring alternative window manipulation techniques, specifically for use on large displays during collocated collaboration when users' desktops are published to a very large display. We exploit the fact that, in such situations, window manipulations such as moving and resizing are more common than direct interaction with a window's content. We assume that the collaborative system either explicitly provides multiple control layers (as in [17]) or that mode switches could be accomplished by a hot key or some similar approach. In this paper we present a novel window manipulation technique, LSO, which is Large Screen Optimized. LSO provides manipulation handles that overlap content, providing a larger target area for moving and resizing windows. We experimentally validate our technique in a within-subjects laboratory study in which participants used both LSO and a traditionally based technique to move and resize windows on a wall-sized display. Our results show that participants were faster when using LSO and that they found it easier to use. While our technique was designed specifically for collaborative use of large displays, it may also be suitable for general use on personal computers. In particular, those users with larger monitors or secondary displays may more frequently arrange multiple visible windows on the screen [14].

## 2 BACKGROUND AND RELATED WORK

We briefly present related work on collaborative meeting software and background on standard window manipulation techniques in modern operating systems before describing research investigating users' windows management practices. We then summarize research investigating new window manipulation techniques.

### 2.1 Collaborative Meeting software

There have been many research projects developing systems to support collocated meetings and the sharing of multiple private desktops and windows on shared public displays. We highlight some of the more recent ones.

WinCuts [24] is an extension to the VNC protocol enabling a VNC server to replicate an arbitrary region of a desktop or window instead of an entire desktop. WinCuts has the potential to optimize use of limited screen space. IMPROMPTU [3] enables users to share windows both on a public group display and on their own personal displays. IMPROMPTU is tied to MS Windows. It encompasses a number of user interface elements that enable easy sharing of application windows, and visibility of who is sharing what via the system. Similarly WeSpace [25] enables the sharing of application windows on both a large shared wall display and a multi-touch table display, using VNC as the basis of its sharing mechanism. WeSpace implements a number of new features, such as a custom API for developing applications to extend the collaborative ability of the system. LACOME [17] also uses VNC and supports cross-platform publishing of content and manipulation of and interaction with that content. LACOME is unique in that it supports multiple platforms, does not require the use of any specific VNC client or server, and provides multiple access control modes such as navigation, control, and annotation.

With the exception of LACOME [17], all of these systems use traditional window manipulation techniques as supported by standard operating systems. For example, WinCuts [24] exposes only the immediately relevant sections of a window rather than the whole window, but does not address moving and resizing windows once they have been created or 'cut.'

### 2.2 Window Manipulation in Operating Systems

MS Windows and Apple's Mac OS X both build window manipulation functions directly into the operating system. On Unix and Linux, window manipulation is handled by a 'window manager,' a special X11 program that varies depending on the distribution. We will focus on Metacity and Compiz, the default window managers for the most common Linux distributions (i.e., Ubuntu, Debian, Fedora).

Standard approaches for window manipulation in commercial operating systems quickly emerged (see [5] for a description of the history of window management techniques). Dragging the titlebar to move a window is standard in all three systems [1, 7, 27, 28]. MacOS X also allows moving by clicking on optional toolbars and bottom bars that together with the titlebar are collectively referred to as the 'window frame' [27]. Linux has the unique capacity to move a window by holding down the 'ALT' key on the keyboard and dragging from anywhere in the window [7, 28].

MS Windows and Linux both enable resizing a window by dragging the small (~5 pixel) border around the window [1, 28, 8]; however, Windows 7 has increased the border to 8 pixels. MacOS X uses a different method for resizing that involves dragging the 'resize control' located in the lower right hand corner of the window. MacOS X windows cannot be resized anywhere else [27]. MS Windows and Linux both offer similar resize controls in the lower right hand corner; however, those controls are considered obsolete in the most recent version of MS Windows [26] and are only sporadically implemented in Linux applications [9]. Additionally, Compiz on Linux offers the ability to resize a window by holding down the 'ALT' key and drag (with the middle mouse button) anywhere in a window [8].

### 2.3 Window Management in Practice

Researchers have investigated how users manipulate their windows in practice ever since windowed systems became popular. In 1986, Bury and Darnell [4] reported that people performed more accurately in windowed systems than non-windowed systems, but they also performed more slowly. The reason for the slower performance is that people spent more of their time engaged in screen-management activity.

More recently, Hutchings and Stasko [14] examined how users managed their windows on regular single and dual display desktop computers. They classified users into 3 categories: *maximizers*, *near maximizers*, and *careful coordinators*. Users of larger screens preferred careful coordination; they simultaneously arrange multiple visible windows on the screen, which in turn requires more fine-grained window manipulation.

Bi and Balakrishnan [2] conducted a week-long user study to compare usage of a large display to single and dual monitor configurations for daily work. Users of the wall-size display performed higher percentages of moving and resizing operations than in their normal environment. Furthermore, users had difficulty accurately selecting the corner of an application window for resizing, particularly when the window was located in the peripheral region of the screen. Bi and Balakrishnan also found that many people had difficulty resizing windows due to small resize targets and long resize distances. They recommended that new methods of window manipulation be developed for large screen displays, focusing on improving the frequent (on a large display) interactions of resizing and moving windows, and deemphasizing less frequently used operations such as minimizing and maximizing.

Moraveji, et. al. [18] found that for large groups (>16), group performance of multiple cursors on a single display degraded with smaller target sizes. Because most windowing systems use relatively small interaction points for window manipulations such as move and resize, this could become a relevant point as the size of groups interacting on shared displays grows.

### 2.4 Windows Management Research

There have been various research projects addressing the problems of window management. Most of these systems are designed specifically for single user systems and may not be as appropriate for a multiuser shared display. Elastic Windows [16] reduces the number of move and resize operations that need to be carried out manually by grouping windows together and moving and resizing them as a group. However in multi-user systems resizing a second window due to manipulations of the first could disrupt another individual's work. QuickSpace [13] involves moving and resizing other windows in response to operations conducted on the active window. Unlike Elastic Windows [16], QuickSpace does not require window groupings, making it more general. However, similar to Elastic Windows, QuickSpace's automatic move and resize operations could disrupt others in a multi-user environment. E-conic [19] uses knowledge of the user's perspective to draw a consistent representation of objects in multi-display environments. This can reduce the problem of distant objects being hard to see and manipulate on large displays. However, because the system is based on a single user's perspective, objects appear skewed to all other participants, making the system impractical in co-located group interactions on shared displays. Metisse [5] is a window system that allows researchers to easily design, implement, and evaluate new window management techniques, such as ZoomOutAndMaximizeHeight and position-dependent window transformations that are indirectly applied when a user moves a window (e.g., shrinking the window as it is moved to the periphery).

Some systems require semantic information about window content. It is an open question as to whether such automated systems will work well with multiple users. For example, AdWiL [11] utilizes a genetic algorithm to place windows for people. Shrinking Window Operations [15] reduces the size of windows based on their relevant semantic content, similar to WinCuts [24].

However this requires access to the semantic content of the window and the ability to determine what is relevant.

Other approaches do not address the problem of moving and resizing windows. For example, Chapuis and Roussel [6] propose a system to automatically expose windows during copy and paste. In order to minimize the number of moves and resizes required to finish a task. Hoffmann et. al. [12] propose a series of methods for visually cueing window notifications and switching in large screen environments, an approach which may add confusion when multiple users are working in different regions of the screen. Sugawara and Maruta propose to replace many window management activities with a push-pin metaphor [22]. However, their system is designed for use on small screen devices and may not be suitable for shared large displays. Also, the total replacement of the window metaphor may confuse people more accustomed to traditional windowing systems.

## 3 LARGE SCREEN OPTIMIZED (LSO) TECHNIQUE

Our technique takes advantage of the fact that to interact with the content of a window in a system such as LACOME [17], the user must explicitly take control of that window. This allows the entire window to be used for manipulations instead of only the title-bar and borders. The LSO technique introduces several novel features. It allows window manipulations to take place anywhere in the window, including in the content pane. It includes 'snapping regions,' that automatically move the cursor to the boundary of the window. Finally, it supports a 'zooming' resize method.

A video figure illustrating the LSO technique is available at: http://www.cs.ubc.ca/labs/imager/video/2010/LargeScreenDisplayWindowManipulations/LSOTechnique.avi, and is recommended for readers interested in the details of the technique.

In the LSO technique, windows are divided into 9 regions: 4 edge regions, 4 corner regions, and the remaining centre region. For our setup, the edge regions were 50 pixels wide and the corners were 50 pixels square; 50 pixels corresponds to approximately 10cm of physical length.

To move a window, a user can click anywhere in the window, not merely the title-bar. If we consider initiating a move operation as a Fitt's Law task, the ability to click anywhere in the window greatly increases the target width compared to the Traditional technique. While the left mouse button is depressed, the cursor is replaced by an arrow-cross. A click in the central region, which is in neither a corner-snapping nor an edge-snapping region, causes the point under the cursor to remain under the cursor as the cursor moves around. If the click is in the snap region near a corner, the cursor is automatically moved to the exact corner. Similarly, if the click is in the snap-region of an edge, the cursor is automatically moved to the nearest point on that edge. Because the cursor can never leave the screen, this gives users a fast and precise method to position a window against an edge or in the corner of the screen

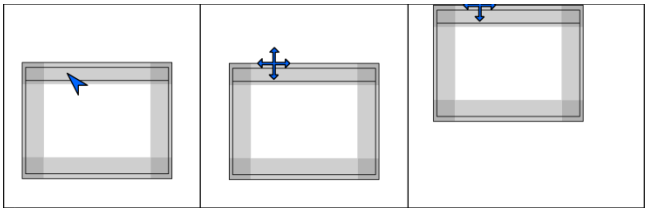

Figure 1: LSO move technique. Left: A user positions the cursor over an edge-snapping region. The snapping regions are temporarily highlighted when rolled-over; for clarity, all snapping regions are drawn in this figure. Center: The left mouse button is pressed; the cursor changes to an arrow-cross and is moved to the exact edge. Right: Moving the mouse positions the window against the top of the screen.

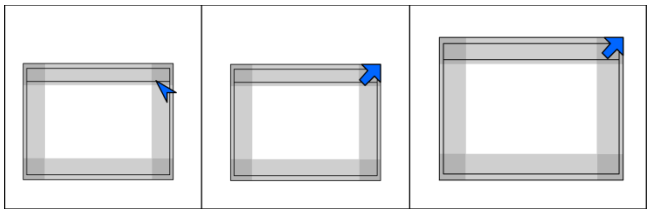

Figure 2: LSO resize technique. Left: A user positions the mouse over the corner snapping region. Centre: When the right mouse button is clicked, the cursor snaps to the exact corner. Right: As the user drags the top-right corner, the bottom-left remains fixed.

(Fig. 1). In Fitt's Law terminology, the edge or corner of the screen is a target of infinite width; the user need merely move the cursor as rapidly as possible towards the edge and it will stop when it reaches the edge. Since snapping positioned the cursor at the exact edge or corner of the window at the beginning of the move action, it will be properly positioned against the edge or corner of the screen after moving. The snap regions are indicated to the user by a semi-transparent white overlay that appears when the cursor rolls over the snap region, and linearly fades after 1 second to full transparency. It should be noted that the snap regions do not change size; they are simply highlighted to indicate to the user that the cursor is within that snap region.

To resize a window in the LSO technique the user clicks with the right-mouse button, anywhere in the window. While the right mouse button is depressed, the cursor changes to an arrow pointing in the direction it will move when dragged. If she clicks near a corner (Fig. 2, left), the cursor is moved to the exact corner (Fig. 2, centre). As in the traditional technique, the opposite corner remains fixed and the selected corner remains under the cursor as it moves (Fig. 2, right). If the user clicked near an edge, the cursor is moved to the nearest point on that edge and that point remains under the cursor as it moves.

If the user right-clicks in the central portion of the window (Fig. 3, left), the cursor remains stationary and the point under it remains fixed. The rest of the window scales about that point as the mouse is moved up and down (Fig. 3, right), analogous to 'zooming' in and out of that point. While the right mouse button is depressed, the cursor becomes an arrow pointing to the top-right.

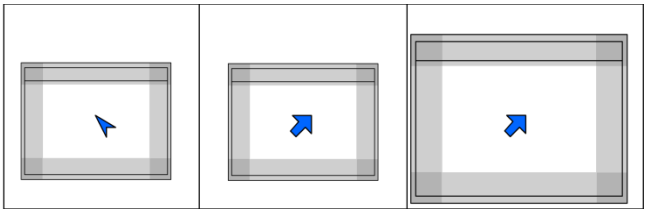

Figure 3: LSO zooming resize technique. Left: The mouse is not placed on a snapping region. Centre: The right mouse button is pressed, and the resize operation begins. Right: The window is resized about the cursor, which remains stationary.

## 4 EXPERIMENT

We conducted an experiment to examine whether the LSO technique was more effective for moving and resizing windows on a large screen display than a traditional manipulation approach. While one of the motivating scenarios for our technique is collaboration, we limited our evaluation to single users to allow us to focus on the effectiveness of the technique for window manipulations typical during large screen display use.

### 4.1 Experimental Design

A within-participant, crossed factorial design was used. As summarized in Fig. 4, we varied the window manipulation technique ('Traditional', LSO), start button position (25% from bottom, 25% from top; always centered horizontally), initial window size (355x236, 1033x614), target size (355x236, 1033x614), and target locations (12: each screen corner, center of each screen edge, 30% towards center from each corner). Only a single start position (center of screen) for the window was used. Between subjects, we counterbalance the ordering of window manipulation technique ('Traditional-then-LSO', LSO-then-Traditional), and content pane image (green or red-tinted pebbles).

### 4.2 Window Manipulation Techniques

The two window manipulation techniques are the LSO technique (described in Section 3) and a 'Traditional' technique that was designed to be as familiar as possible, with many similarities to the windowing mechanism of MS Windows. To move a window in the traditional technique, the user left-clicks the title-bar and drags the mouse. To resize a window the user left-clicks the window border, either on a corner or on an edge, and drags the mouse. If a corner was chosen, the position of the opposite corner remains fixed and the selected corner stays under the cursor as the cursor moves. If an edge was chosen, the position of the point directly across from the selection point remains fixed, and again the selected point stays under the cursor. When the cursor is placed over the title-bar it changes to an arrow-cross ("cross barby") to indicate that clicking will initiate a move; when over the window border, the cursor changes to an arrow pointing in the direction to which the cursor will be constrained while resizing.

For the purposes of the study, both techniques share the property that they preserve the aspect ratio of the content pane. This is motivated by the fact that the hypothetical use case of these windows is publishing computer desktops on a shared display; reshaping a window would thus entail either distorting or cropping the content of the window. While the window is being resized, the cursor is constrained to move only along a path that will preserve the aspect ratio. That is, the cursor will only move diagonally if a corner was selected, horizontally if the left or right edge was selected, and vertically if the top or bottom edge was selected. In the experiment an aspect ratio of 2.375:1 was used for the content pane (see Fig. 5).

### 4.3 Task

Participants were required to perform window manipulations involving moving and/or resizing. On each trial a 'Start' button is centered horizontally, and positioned either 25% from the bottom or 25% from the top of the screen. Two positions were utilized in order to investigate the effect of distance to the target. For the Traditional technique, the top position is closer than the bottom position, while for the LSO technique the distance is the same. The start button is 135x50 pixels in size. When clicked, the start button disappears and the trial starts. Immediately, a window centered on the screen appears. Each window, as illustrated in Fig. 4, shows an image of pebbles in its content pane for the experiment. Above the content pane is a solid-color title-bar, with a height of 32px (Fig. 5). The title-bar has no buttons or other decorations on it. Surrounding both the content pane and the title bar is an 8px solid-color border.

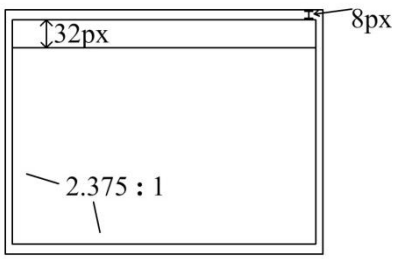


Figure 5. Design of windows used in the experiment.

The task is to fit the window over the target area, which appears in one of 12 locations when the start button is clicked, at the same time as the window. For some trials the window is initially the same size as the target, so it is only necessary to move the window. On other trials, the window also needs to be resized (as shown in Fig. 4). It is not required that the match be pixel-perfect; we consider a trial completed when all 4 corners of the window are within 10px of the corners of the target. Upon achieving a successful fit both the window and the target area disappear, and the Start button reappears for the next trial.

To allow the target behind the window to be visible at all times, the window is made semi-transparent (Fig. 4). While being manipulated it has an alpha value of 0.5 (equal blending); otherwise it has an alpha value of 0.8 (mostly opaque).

### 4.4 Study Protocol

Each participant was guided through three initial trials to teach them the basic mechanics of the task. They were given 30 training trials with continued verbal instruction about the technique, read from a script. Questions were encouraged during these training trials. Participants next completed a block of 48 trials, after which they were asked to move to a different table and complete a maze, which was intended to reduce fatigue or boredom effects from performing a large number of repetitive tasks. This was followed by a second block of 48 trials and a short questionnaire regarding the technique they had just used. This process (training, block 1, maze, block 2, questionnaire) was repeated for the second window manipulation technique. Finally, participants completed a questionnaire comparing the two conditions.

Each half of the experiment consisted of 96 unique trials, split into two blocks. The blocks were balanced, fully crossed on initial window size, target size, and start button position. Six of the target locations were used in each block (2 corners, 2 edges, 2 floating). See Fig. 4 for their placement. The 24 participants completed 96 blocks in total. The 48 trials within each block were intended to be randomly ordered. Unfortunately, due a programming error only four random orderings were actually used. Each of the first 12 participants saw the same ordering for Block 1 of each technique, and a second ordering for Block 2 of each technique. Similarly, each of the second 12 participants saw a third ordering for the Block 1, and a fourth ordering for Block 2.

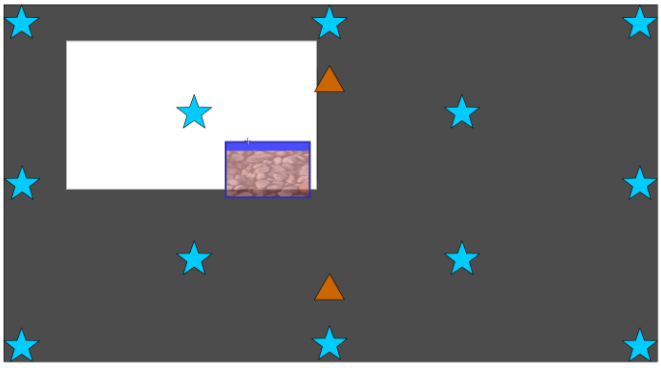

Figure 4: Screen shot of a trial in progress, with overlays of the 12 possible target window positions (blue stars) and 2 Start Button positions (red triangles). The white target window shows the large size, and the pebbled window shows the small size.

### 4.5 Apparatus

We used a large screen wall display measuring 5.31m in width and 2.97m in height (17.4'x9.8'). The display is comprised of an array of 12 projectors in a 4x3 tiling shining onto a single frosted-glass vertical surface. The bottom of the screen is at floor level, and it is centered in an approximately 10m x 6m room. A dual-link DVI output feeds into a small network of XPO3 video processors from Cyviz LLC, which 'chop' the output into 12 portions and blend the edges together. The projectors each have a native resolution of 800x600px, and were given a 160px (~30 cm) blended overlap with their neighbours in both the horizontal and vertical directions, for a total resolution of 2720x1480px. The final blending functions were not implemented at the time of the experiment so some variation in brightness and color was still present. Additionally, there were a small number of projector alignment errors that led to blurriness or tearing of the composite image. These alignment errors are most visible where there are high-contrast edges and/or fine details, such as black text on a white background. To minimize the effect of these alignment errors and blending variations we used solid colors of medium lightness for the screen background, window title-bars, and window borders, and an image of mottled pebbles for the window content.

Participants were seated at a table that is 1.5m long, 0.75m wide, and 0.7m high. The edge nearest the participant was centered and positioned 2.7m from the screen. This distance was chosen for a number of reasons. Earlier work by Bi and Balakrishnan [2] suggests that a distance of 2.0-2.5 meters is preferred for a screen of this width. However, at 6' tall their screen was shorter than ours, and pilot participants noted that a greater distance was required to allow natural viewing of the corners of the screen. Additionally, our screen is seated at ground level so shorter pilot participants could not see the bottom of the screen because of occlusion by the far edge of the table. This also required us to use a table of somewhat narrow width.

Participants used a "MS Wireless Optical Mouse 2000" to move the on-screen cursor. As the resolution was quite high, pilot participants found it difficult to move the cursor to the corners in one motion and still have the fine control necessary to complete the task, unless cursor acceleration was left on. We enabled cursor acceleration and set the cursor speed to 6 out of 11. The experimental software ran full-screen so users were unaware of any details of the underlying computing environment.

### 4.6 Participants

We recruited 24 participants (15 male, 9 female) from the university community. Participants were required to have normal or corrected to normal vision and a lack of colour blindness. All but one participant was right-handed. Only 10 participants indicated that they regularly use (more than once per week) a single type of input device; almost all (23/24) were regular users of mice, but touch pads (15), touch screens (3), track point (1) and stylus (1) were also in use. Roughly half (13/24) regularly use multiple operating systems: MS Windows (21), Linux (12), and Mac (5). No participants indicated that they regularly use a large screen display, but 11 indicated that they had performed window manipulations on wall/table top displays (5) and projected wall displays (6) in the past.

## 5 RESULTS

We first present manipulation time results and then subjective results from the questionnaires. We qualitatively support our findings with participants' comments about the ease of use and relative advantages of the two techniques.

### 5.1 Performance Data

Preliminary analysis revealed that, as expected, content pane image (green-tinted pebbles, red-tinted pebbles) had no impact on the findings, so we dropped that variable from our analysis. The combination of initial window size and target size results in two types of Action: Move (small to small, large to large) and Move+Resize (small to large, large to small). The dependent variable was manipulation time, which began when the start button was clicked and ended when the window was correctly positioned over the target.

Table 1. Main effects and significant interactions of independent variables on the mean manipulation time.

| †Huynh-Feldt correction applied | | Manipulation Time | | Test Statistics | | |
|---|---|---|---|---|---|---|
| *p<.05 | IV | μ | SD. | F | p | η2 |
| **Main Effects (Between Subjects)** | | | | | | |
| Tech. Order | LSO-1 | 5.41 | 3.48 | 0.508 | .484 | |
| | LSO-2 | 5.04 | 3.03 | | | |
| **Main Effects (Within Subjects)** | | | | | | |
| Tech. | Trad. | 5.57 | 3.25 | 21.660 | .000* | .496 |
| | LSO | 4.89 | 3.36 | | | |
| Action | Move | 3.09 | 1.46 | 223.506 | .000* | .910 |
| | Move+ Resize | 7.36 | 3.29 | | | |
| Final pos. type | Corner | 5.38 | 3.64 | 11.046 | .000* | .334 |
| | Edge | 5.29 | 3.33 | | | |
| | Float | 5.01 | 2.95 | | | |
| Block | First | 5.37 | 3.59 | 7.652 | .011* | .258 |
| | Second | 5.09 | 3.02 | | | |
| Start button pos. | Bottom | 5.30 | 3.39 | 12.602 | .002* | .364 |
| | Top | 5.15 | 3.25 | | | |
| **Significant Interactions** | | | | | | |
| Technique x Technique Order | | | | 11.136 | .003* | .336 |
| Start Button x Technique Order | | | | 5.224 | .032* | .192 |
| Technique x Action | | | | 8.411 | .008* | .277 |
| Action x Final Position Type | | | | 3.961 | .032† | .153 |
| Action x Block | | | | 4.635 | .043* | .174 |
| Final Position Type x Block | | | | 10.101 | .000* | .315 |

Changes in scores on the dependent variable were analyzed in a 2 (technique order: LSO-1st, LSO-2nd) x 2 (technique: Traditional, LSO) x 2 (Action: Move, Move+Resize) x 3 (Final Position Type: corner, edge, floating) x 2 (Block: first, second) x 2 (Start Button Position: top, bottom) mixed analysis of variance (ANOVA) with a between subjects measure on the first variable. Adjustments were made to the ANOVA results, using the Huynh-Feldt correction for the Action x Final Position Type interaction to account for the violation of the sphericity assumption revealed by Mauchly's Test of Sphericity. Table 1 provides a summary of the main effects and shows the six interactions that reached significance. Significant main effects were found for all the within-subjects variables, but not for the between-subjects variable. We will examine the interesting findings in turn.

#### 5.1.1 Main Effects

There was a statistically significant main effect of Technique, validating the usefulness of LSO technique for window manipulation on a large screen display. The mean performance advantage is 0.68 seconds per manipulation. As discussed below, there were interactions of Technique with both Technique Order (section 5.1.2) and Action (section 5.1.3).

As expected, there was a statistically significant main effect of Starting Button Position on manipulation time. Because a window in the Traditional condition must be dragged by the title-bar, participants must move the cursor further to begin movement when the start button is in the lower position. We had expected to see an interaction between Start Button Position and Technique,

because clicking anywhere in the window begins a move action in the LSO technique so there should be little no difference between the two start positions. While the mean difference in times for the LSO technique (Δ=0.074 s) was smaller than that of the Traditional technique (Δ=0.216 s), the difference was not statistically significant (p = 0.291).

There were also significant main effects of Action and Final Position Type. This is surely to be expected, as the Move+Resize condition requires strictly more interaction than the Move condition, and the corners, edges, and floating targets are all at different distances from the window's starting position.

There was also a significant main effect of Block, with manipulation time decreasing in the second block for each condition. If we plot manipulation time over all four blocks performed by each participant, there is a continued downward trend; we suspect that performance had not yet plateaued at the end of the experimental trials.

#### 5.1.2 Technique Order Interactions

There was no main effect of technique order; however, there was an interaction between both Technique Order and Technique and between Technique Order and Start Button Position. In the interaction of Technique Order and Start Button Position, those participants who began with the LSO technique were slower when starting from the lower position. Presumably, this is because they were unused to the precision required for this movement when they began to use the Traditional technique in the second half; see section 6.1 for a more complete discussion. We argue similarly to our explanation for the Technique Order by Technique interaction: participants who began with LSO were unused to the precision required in the Traditional technique and their performance suffered.

#### 5.1.3 Technique x Action

There were main effects for both technique and for action. There was also an interaction between the two. While LSO was somewhat faster for moving windows (Δ=−0.31s), it was much faster for resizing (Δ=−1.06s). This is well supported by the qualitative feedback from our questionnaires; users reported much difficulty resizing windows using the Traditional technique.

#### 5.1.4 Action x Final Position

There was a statistically significant interaction effect between Action and FinalPosition, which was not hypothesized in advance. The Move+Resize action is slower in corners (μ=7.59s) and at edges (μ=7.47s) than for floating targets (μ=7.02s) because participants would overshoot the target, and the window's resize handles would become inaccessible. When this happened, a recovery action was needed, such as moving the window so it was completely visible once again. This did not occur when only moving, because the resize handles were not needed, nor did it occur when the target was at a floating final position, as no portion of the window was likely to go off-screen. As all higher-order interactions were non-significant, the Action x Final Position Type interaction generalizes across all other variables.

The effect of this phenomenon was reflected in the questionnaire data; several participants explicitly mentioned the snapping regions when commenting on their preference for the LSO technique when moving and resizing window in the corners and edges of the screen (section 5.2.2). We examined the use of these snapping regions in the LSO technique and the impact that the use had on manipulation time. For those targets located on the edge of the screen and in the corners, the snapping technique would be most useful. We classified a trial as having an optimal snapping action (for a corner target, in the matching corner region; for an edge target, in the matching edge region or the adjacent corner regions) or not (snapping to another region, or not snapping at all). As seen in Table 2, participants more frequently took appropriate snapping actions for corner targets (48% of corner trials using LSO technique) than for side targets (27%). This is likely due to the fact that for edge targets there was still a need to position the target along the edge, while for corner targets the window gets effectively locked in place by the corner. Snapping appropriately resulted in improved performance times for Move+Resize (Δ=−0.17s) actions to an edge target and both Move (Δ=−0.21s) and Move+Resize (Δ=−1.11s) in corner targets; however, there was no performance gain for a Move to the edge.

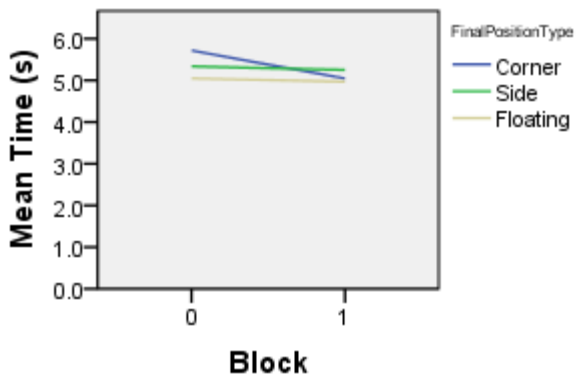


Figure 6: Graph of Final Position Type x Block interaction. There is a strong improvement for corner trials. (Mean time in seconds.)

Table 2. Comparison of mean manipulation times in the LSO technique when snapping was used optimally or not.

| Target Position | Action | Optimal? | Freq. | μ (s) | S.D. (s) |
|---|---|---|---|---|---|
| Move | Edge | Yes | 108 | 2.97 | 0.91 |
| | | No | 276 | 2.96 | 1.56 |
| | Corner | Yes | 169 | 2.86 | 2.37 |
| | | No | 215 | 3.07 | 1.72 |
| Move+ Resize | Edge | Yes | 98 | 6.91 | 2.94 |
| | | No | 286 | 7.08 | 3.85 |
| | Corner | Yes | 199 | 6.52 | 3.33 |
| | | No | 185 | 7.63 | 4.78 |

#### *5.1.5 Final Position Type x Block, Action x Block*

The Final Position Type x Block interaction, graphed in Fig. 6, shows a distinct improvement in fitting windows to the corners, far greater than the improvement attributable solely to block. In the Action x Block interaction, participants showed a larger improvement for Move+Resize trials (Δ=−0.47s) than for Move trials (Δ=−0.08s).

### 5.2 Questionnaire Data

Participants filled out questionnaires with ratings and rankings.

#### 5.2.1 Post-technique Results

After each technique, participants rated the technique according to its ease of use overall and for each Action/Final Position Type combination (Table 3. LSO was rated easier to use than the Traditional technique for both Move and Move+Resize). Non-parametric Wilcoxon matched-pairs signed-ranks tests (Z score reported), with a Bonferroni adjustment of the significance level to .007 to compensate for multiple comparisons, revealed this difference was significant except for free-floating target locations (marginally significant for Move+Resize to floating targets).

Table 3. Participant ratings of the window manipulation techniques according to their ease of use. (5 point scale: 1=hard, 5=easy)

| p< .007* | | Traditional | | LSO | | Test stats | |
|---|---|---|---|---|---|---|---|
| **Action** | **Target** | **μ** | **S.D.** | **μ** | **S.D.** | **Z** | **p** |
| **Move** | Float | 4.33 | .963 | 4.67 | .637 | 1.604 | .109 |
| | Edge | 3.46 | 1.062 | 4.54 | .509 | 3.696 | **.000*** |
| | Corner | 3.42 | 1.176 | 4.46 | .833 | 3.452 | **.001*** |
| **Move+ Resize** | Float | 3.42 | .830 | 4.17 | 1.007 | 2.452 | .014 |
| | Edge | 3.17 | 1.049 | 3.92 | .929 | 3.366 | **.001*** |
| | Corner | 3.13 | 1.191 | 4.00 | 1.103 | 3.827 | **.000*** |
| **Overall** | | 3.21 | .884 | 4.29 | .690 | 4.245 | **.000*** |

#### 5.2.2 Post-session Preferences

After the participants had completed both techniques, they completed a questionnaire that asked them to rank the techniques according to their overall preference, as well as to indicate their preferred technique for the various moving/resizing target locations (floating, edge, corner). The vast majority (23/24) of participants preferred the LSO technique over the Traditional window management technique; and in each specific condition at least 19/24 (79.2%) participants preferred the LSO technique (see Fig. 7). Chi-square analyses, with a Bonferroni adjustment of the significance level to .007 to compensate for multiple comparisons, revealed that this preference was significant in all cases ($p<.002$).

An examination of individual rankings for each combination of action and target location revealed that 15 participants consistently indicated a preference for LSO and 1 consistently indicated a preference for the traditional technique. Interestingly, the single participant who preferred the traditional technique was the only participant who was not a regular mouse user. This participant was also an outlier in terms of mean window manipulation time.

We examined participant comments for the reasons behind their preferences. For the Move and Move+Resize to the floating and corner target positions, about one third of participants explicitly mentioned the snap-to feature of the LSO technique. The ability to click anywhere in the LSO technique was mentioned for all targets (i.e., 16/19 for Move-floating, 7/22 for Move-corner), with some noting that the technique was "less rigid" than the traditional technique and that LSO "allows traditional manipulation if wanted." While most participants did not explicitly comment on the use of two buttons in LSO, the few that did were mixed in their opinions. For example, one said (for Move+Resize-floating) "using right click is much more convenient"; however, another consistently made the comment "using both R&L buttons is confusing." It was interesting that three of the participants specifically noted that the overloaded left mouse button in the traditional techniques was problematic, as one noted "there were also several occasions when I'd want to move the window via the top bar and would accidentally resize it instead (or vice versa)."

One aspect of the LSO technique that appears to have been problematic was the zooming-resize technique (Fig. 3). After using LSO, three participants made negative comments about this technique, with one commenting that it should resize more slowly. Two others commented that it was more difficult to shrink a window than to expand it.

#### 5.2.3 Tradeoff: 2 buttons, larger selection space

We asked participants to choose between using (a) both the left and right mouse buttons with large selection regions (i.e., handles), or (b) only one button with smaller selection regions; 22/24 participants indicated a preference for two buttons and a larger selection space. Having larger handles was the primary concern for 6 participants. While one commented, "I don't like using two buttons, but I like a larger target more," 5 others felt that using two buttons was not much more complex, although some training might be needed. A few participants considered the ramifications of using the right mouse button for resizing during normal use, with one commenting that their choice depended upon the importance of window manipulation to the task and another saying, "I think we can right click for resizing and design another key to replace right click's tasks [context menu]."

## 6 DISCUSSION

The LSO technique outperformed the Traditional window management technique for both moving and resizing, regardless of the start button position or the final target location. The ability to click anywhere in the window when moving reduced the distance participants had to travel and increased the target size. The performance advantage was greater for resizing than for moving, in part due to the snapping regions in LSO that constrain users from moving a window past the screen edge when appropriately snapped. Furthermore, the use of the right mouse button helped prevent participants from inadvertently selecting the wrong handle and moving rather than resizing or vice versa. Participants subjectively rated LSO as being easier to use than the Traditional technique and they preferred it for use on a large screen display.

### 6.1 LSO Beyond Large Collaborative Displays

While our LSO technique was designed specifically for collaborative use of large displays, it may also be suitable for general use on personal computers, particularly for those users with larger monitors or secondary displays who may more frequently arrange multiple visible windows on the screen [14]

We developed the LSO technique to improve window manipulation on large displays. We have since learned that some of the windows managers in distributions of Linux provide a similar technique of clicking anywhere within a window for moving (ALT+drag with left mouse button) [7, 28] and resizing (ALT+ drag with middle mouse button) windows [8]. We could find no evaluation of this technique in the literature, however the success of our similar technique during our study provides validation for this approach.

In order to utilize the LSO technique in other systems, we would have to provide some mechanism for a mode switch; the successful integration of modifier keys for this purpose in the Linux window managers shows that this approach to mode switching works. As our questionnaire data revealed, our participants were quite willing to complicate the manipulation process by using two different mouse buttons to initiate the correct action in return for the larger selection areas. Of course, they were

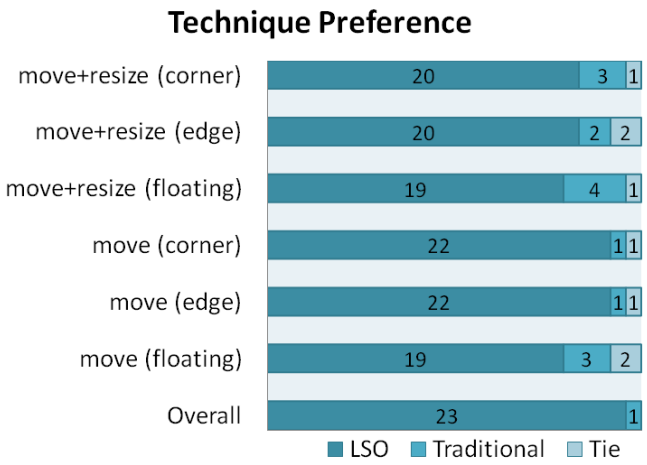


Figure 7. Preferences for the window manipulation techniques.

not required to perform mode switches during the experiment, so further study is required to determine if this switch would prove to be a hindrance in a real-world system.

### 6.2 Learning and Transfer Effects

We found a significant effect of Block, and significant interactions involving Technique Order, which indicate learning and transfer effects. Despite the apparent simplicity of the task, it appears that participants continued to improve overall across all four blocks; as a methodological issue, one should expect a longer acclimation period when users must adapt to using a large screen, even when using a fairly familiar technique such as our Traditional technique.

In this study, we used technique orderings of A-B and B-A (A=Trad., B=LSO). To establish a better baseline of learning curves, we could also use A-A and B-B orderings; that is, have participants use one technique for four consecutive blocks. This would help us separate transfer effects from learning effects.

As shown in Fig. 8, participants who began with the Traditional technique did extremely well when they switched to the LSO technique; those who started with LSO showed only minor improvements in the Traditional technique compared to those who started with Traditional. Because the questionnaire data showed that participants found the Traditional technique harder, we believe that those who started with Traditional were practicing much harder. After using the narrow edges and minute corners of the Traditional technique, these participants found the large regions of the LSO technique very easy to use. Those who began with the easier LSO technique did not receive this practice benefit, and so showed only minor improvement.

This unexpected asymmetric transfer effect calls into question the validity of using a within-subjects experiment design. However, examining only the first two blocks in Figure 8, before any transfer effects can happen, participants using LSO already see statistically significant gains. Additionally, using a within-subjects allowed us to collect data such as preference information, which would have been impossible in a between-subjects experiment.

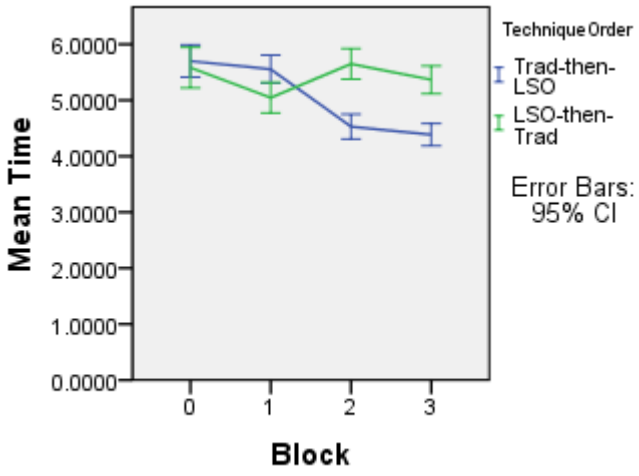


Figure 8: Each line has two Traditional blocks and two LSO blocks; the four higher points are Traditional and the four lower are LSO. Learning effects are apparent in each pair of blocks. Those who start with Traditional do much better in LSO, as shown by the blue line from top-left to bottom-right. (Mean time in seconds.)

### 6.3 Limitations

Our study was an initial look at evaluating the LSO window management technique. It was intended to show that practical and significant gains could be made over a traditional windowing system. However, windowing systems are used in many ways and in many contexts, and no one study can demonstrate the efficacy of the LSO technique across all of them. We next discuss a number of limitations to our evaluation that must be resolved through further study.

Users also found that they were unable to achieve the accuracy required using the 'zooming' resize method, and the feature was consequently little used. We believe that this technique remains promising for real-world situations and should be validated with different tasks, or different methodologies.

Our study was conducted with single users who were squarely facing the display. While we believe that our results should hold for multiple users and alternate seating positions, further validation is required for collaborative meeting environments.

The level of accuracy required in this experiment may have been higher than users typically require in practice. While users strongly preferred the LSO technique for the task given, it remains to be seen if this result holds during long term real-world usage when the moving and resizing actions are interspersed with other tasks.

All windows in this study were non-semantic; the improvements offered by the LSO technique should translate directly to systems where windows are primarily viewed, moved, and resized. If the LSO technique is to be used in more general systems where users frequently interact with the semantic contents of a window, an effective mode toggle must be found and the performance cost of performing this mode switch must be measured. Our future work will include a comparative study that will better evaluate the impact of mode switching.

## 7 CONCLUSIONS

LSO is a novel window manipulation technique, optimized for use on large-screen displays. Because LSO uses the full window area for manipulations, rather than only the borders and title-bar, it provides larger manipulation handles which are easier and faster to use than those on traditional windowing systems. By using LSO's "snapping regions," which move the cursor to the precise edge or corner of the window when clicked, users are able to place windows against the edge or corner of the screen more easily. These advantages may also be transferable to other form factors such as desktop monitors or mobile devices. By providing a mode switch, such as with a modifier key, a user could switch between interacting with the contents of a window and manipulating it.

We conducted a laboratory study with 24 users to investigate whether the LSO technique would enhance performance in window manipulation tasks. We found that users performed 10% better on Move tasks and 13% on Move+Resize tasks with the LSO technique. Users also strongly preferred the LSO technique; 96% of users preferred it overall and 62% preferred it for each and every combination of action and final position type.

## REFERENCES


1. About Windows. *Windows User Experience Interaction Guidelines*. http://msdn.microsoft.com/en-us/library/ms632597(VS.85).aspx
2. Bi, X. and Balakrishnan, R. Comparing usage of a large high-resolution display to single or dual desktop displays for daily work. In *Proc. CHI '09*, ACM Press (2009), 1005-1014.
3. Biehl, J., Baker, W., Bailey, B.P., Tan, D.S., Inkpen, K., Czerwinski, M. Impromptu: A New Interaction Framework for Supporting Collaboration in Multiple Display Environments and it's Field Evaluation for Co-located Software Development. In *Proc. CHI '08*, ACM Press (2008), 939-948.
4. Bury, K. and Darnell, M. Window Management in Interactive Computer Systems. *SIGCHI Bul*. 18:2 (1986), 65-66.

5. Chapuis, O. and Roussel, N. Metisse is not a 3D desktop! In *Proc. UIST '05*, ACM Press (2005), 13-22.
6. Chapuis, O. and Roussel, N. Copy-and-paste between overlapping windows. In *Proc. CHI '07* (2007), 201-210.
7. Compiz. http://wiki.compiz.org/Plugins/Move
8. Compiz. http://wiki.compiz.org/Plugins/Resize
9. Controls and Status Bars. *Gnome User Guide*. http://library.gnome.org/devel/hig-book/stable/controls-status-bars.html.en
10. Czerwinski, M., Robertson, G., Meyers, B., Smith, G., Robbins, D., Tan, D.S.. Large Display Research Overview. In *Ext. Abstracts of CHI '06*, ACM Press (2006), 69-74.
11. Haraty, M., Nobarany, S., DiPaola, S., and Fisher, B. AdWiL: adaptive windows layout manager. In Ext. Abstracts CHI '09, ACM Press (2009), 4177-4182.
12. Hoffmann, R., Baudisch, P., and Weld, D. S. Evaluating visual cues for window switching on large screens. In Proc. CHI '08, ACM Press (2008), 929-938.
13. Hutchings, D.R. and Stasko, J. QuickSpace: New Operations for the Desktop Metaphor. In Ext. Abstracts CHI '02, ACM Press (2002), 802-803/
14. Hutchings, D. R. and Stasko, J. Revisiting display space management: understanding current practice to inform next-generation design. In *Proc. of GI* (2004), 127-134.
15. Hutchings, D. R. and Stasko, J. Shrinking window operations for expanding display space. In *Proc. AVI '04*. ACM Press (2004), 350-353.
16. Kandogan, E. and Shneiderman, B. Elastic windows: improved spatial layout and rapid multiple window operations. In *Proc. AVI '96*, ACM Press (1996), 29-38.
17. Liu, Z. Lacome: A Cross-platform Multi-user Collaboration System for a Shared Large Display. Master's Thesis. Dept. of Computer Science, University of British Columbia, 2007.
18. Moraveji, N., Inkpen, K., Cutrell, E., and Balakrishnan, R.. A mischief of mice: examining children's performance in single display groupware systems with 1 to 32 mice. In *Proc. CHI'09,* ACM Press (2009), 2157-2166.
19. Nacenta, M., Sakurai, S., Yamaguchi, T., Miki, Y., Itoh, Y., Kitamura, Y., Subramanian, S., and Gutwin, C. E-conic: a perspective-aware interface for multi-display environments. In *Proc. UIST '07*, ACM Press (2007), 279-288.
20. Nakashima, K., Machida, T., Kiyokawa, K., Takemura, H. A 2D-3D Integrated Environment for Cooperative Work. In *Proc. of VRST '05,* ACM Press (2005), 16-22.
21. Ni, T., Bowman, D.A., and Chen, J.. Increased Display Size and Resolution Improve Task Performance in Information-rich Virtual Environments. In *Proc. of GI* (2006), 139-146.
22. Sugawara, K. and Maruta, R. A novel intuitive GUI method for user-friendly operation, *Knowledge-Based Systems*, Vol. 22 (2009), 235-246.
23. Swaminathan, K. and Sato, S. Interaction Design for Large Displays. *Interactions*, 4(1) (1997).
24. Tan, D. S., Meyers, B., and Czerwinski, M. WinCuts: Manipulating Arbitrary Window Regions for more Effective Use of Screen Space. In *Ext. Abstracts of CHI '04*, ACM Press (2004), 1525-1528.
25. Wigdor, D., Jiang, H., Forlines, C., Borkin, M., and Shen, C. WeSpace: The Design Development and Deployment of a Walk-up and Share Multi-surface Visual Collaboration System, in Proc. CHI '09, ACM Press (2009), 1237-1246.
26. Window Management. *Windows User Experience Interaction Guidelines*. msdn.microsoft.com/en-us/library/aa511262.aspx.
27. Windows. *Apple Human Interface Guidelines* http://developer.Apple.com/mac/library/documentation/UserExperience/Conceptual/AppleHIGuidelines/XHIGWindows/XHIGWindows.htm
28. Windows Manipulating. *Gnome User Guide* http://library.gnome.org/users/user-guide/stable/windows-manipulating.html.en