Drag-and-Pop and Drag-and-Pick: techniques for accessing remote screen content on touch- and pen-operated systems

Patrick Baudisch[1], Edward Cutrell[1], Dan Robbins[1], Mary Czerwinski[1], Peter Tandler[2], Benjamin Bederson[3], and Alex Zierlinger[4]

[1]Microsoft Research, Redmond, WA; [2]Fraunhofer IPSI, Darmstadt, Germany; [3]HCIL, University of Maryland, MD; [4]Maila Push, Darmstadt, Germany

{baudisch, cutrell,czerwinski, dcr}@microsoft.com; tandler@ipsi.fhg.de; bederson@cs.umd.edu; alex@zierlinger.de

Abstract: Drag-and-pop and drag-and-pick are interaction techniques designed for users of pen- and touch-operated display systems. They provide users with access to screen content that would otherwise be impossible or hard to reach, e.g., because it is located behind a bezel or far away from the user. *Drag-and-pop* is an extension of traditional drag-and-drop. As the user starts dragging an icon towards some target icon, drag-and-pop responds by temporarily moving potential target icons towards the user's current cursor location, thereby allowing the user to interact with these icons using comparably small hand movements. *Drag-and-Pick* extends the drag-and-pop interaction style such that it allows activating icons, e.g., to open folders or launch applications. In this paper, we report the results of a user study comparing drag-and-pop with traditional drag-and-drop on a 15' (4.50m) wide interactive display wall. Participants where able to file icons up to 3.7 times faster when using the drag-and-pop interface.

Keywords: Drag-and-drop, drag-and-pick, interaction technique, pen input, touchscreen, heterogeneous display.

1 Introduction

With the emergence of pen- and touch-operated personal digital assistants (PDAs), tablet computers, and wall-size displays (e.g., Liveboard, Elrod et al., 1992; Smartboard, http://www.smarttech.com), touch and pen input have gained popularity. Over the past years, more complex display systems have been created by combining multiple such display units. Wall-size touch displays have been combined into display walls, such as the DynaWall (Streitz 2001), or the iRoom Smartboard wall (Johanson, 2002b). Recent PDAs and tablet computers allow connecting additional displays, such as another tablet or a monitor in order to extend the device's internal display space.

Touch/pen-operated screens that consist of multiple display units bring up a new class of input challenges that cannot always be solved with existing techniques, because many of the existing techniques were designed for indirect input devices, such as mice, track pads, or joysticks. Indirect input devices can be used on arbitrary display configurations, because they can simply be mapped to the respective topology (e.g., PointRight, Johanson 2002a). Touch/pen input, however, is based on the immediate correspondence between input space and display space and thus requires users to adapt their input behavior to the physicality of the display system. Here are three examples where this can become problematic.

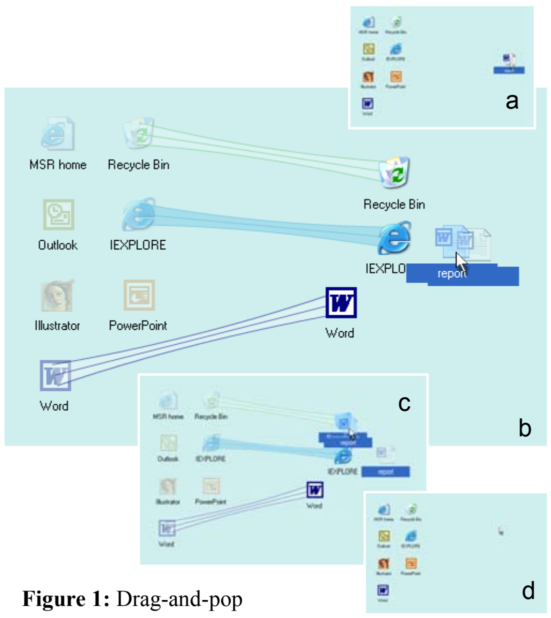

Figure 1: Drag-and-pop

Scenario 1: External monitors. One or more display units within a display system may not be equipped with a touch or pen sensor. Connecting an external monitor to a tablet computer or PDA, for example, allows users to see more material, but requires them to use an indirect input device, such as a mouse, when interacting with content on the external monitor. Since some tablet-specific tasks, such as scribbling, are hard to accomplish with a mouse, users find themselves continuously switching between pen and mouse.

Scenario 2: Interactions across display units. Some interaction techniques, such as drag-and-drop, require users to interact with two or more icons in a single pen-down interaction. If these icons are distributed across physically separate pen/touch input display units, users first have to bring all involved icons to the same display unit, a potentially time-consuming activity (Figure 2a-c).

Scenario 3: Bridging long distances. Accessing icons located far away from the user, e.g., on the opposite side of a 15' DynaWall, requires users to physically walk over, the time for which may in some circumstances increase linearly with distance (Guiard et at, 2001). In addition, *drag* interactions get more error-prone with distance, because users drop objects accidentally when failing to continuously keep the pen tip in contact with the display surface (Rekimoto 1997).

2 Drag-and-pop & drag-and-pick

Drag-and-pop and *drag-and-pick* are interaction techniques that address these issues. We will begin by giving an overview; more detailed descriptions of both techniques can be found in Section 4.

Drag-and-pop extends traditional drag-and-drop as illustrated by Figure 1. (a) The user intends to delete a Word memo by dragging it into the recycle bin. (b) As the user starts dragging the memo's icon towards the recycle bin, icons that are of compatible type and located in the direction of the user's drag motion "pop up". This means that for each of these icons a link icon is created (*tip icon)* that appears in front of the user's cursor. Tip icons are connected to the original icon (*base icon*) using a rubber band. (c) The user drags the memo over the recycle bin and releases the mouse button. The recycle bin accepts the memo. Alternatively, the user could have dropped the memo over the word processor or the web browser icon, which would have launched the respective application with the memo. (d) When the user drops the icon, all tip icons disappear instantly.

Figure 2d shows how drag-and-pop simplifies dropping icons onto targets located at the other side of a bezel that separates display units (scenario 2). Figure 9 shows a user performing a drag-and-pop interaction to drop an icon on a distant target.

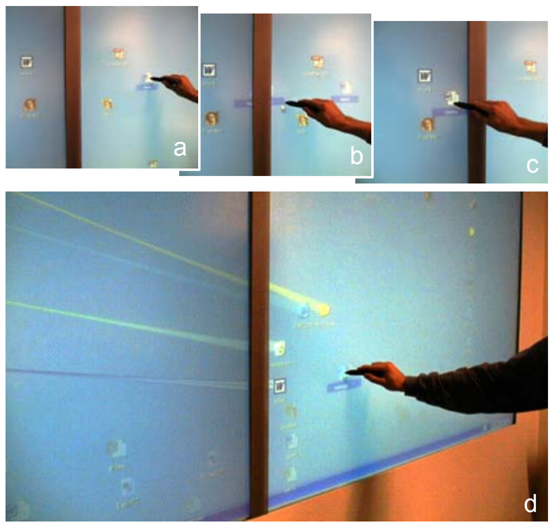

Figure 2: (a-c) Traditional drag-and-drop: Dragging an icon across the bezel requires the user to drop the icon half way across the bezel and pick it up at the other side (d) Drag-and-pop temporarily brings matching target icons to the current pen location, allowing the user to file icons without having to cross the bezel.

Drag-and-pick modifies the drag-and-pop interaction concept such that it allows activating icons, e.g., to open a folder or to launch a program. While drag-and-pop is initiated by the user dragging an icon, drag-and-pick starts with the user performing a drag interaction on empty screen space. The system's response to this drag interaction is similar to drag-and-pop, but with two differences. First, *all* icons located in the direction of the drag motion will pop up, not only those of compatible type (Figure 3). Second, as the user drags the mouse cursor over one of the targets and releases the mouse button, the folder, file, or application associated with the icon is activated as if it had been double clicked.

Figure 4 shows how this allows users to use the pen for launching an application, the icon of which is located on a monitor not supporting pen input.

In principle, drag-and-pick can be applied to any type of widget, e.g., any buttons and menus located on a non-pen accessible monitor. In this paper, however, we will focus on the manipulation of icons.

3 Related work

Drag-and-drop is a well-know interaction technique for transferring or copying information using a pointing device, while avoiding the use of a hidden clipboard (Wagner, 1995; Beaudouin-Lafon, 2000). Hyperdragging (Rekimoto, 1999), allows extending drag-and-drop across physically separate displays (Scenario 2), but requires an indirect input device, such as a mouse. Most techniques compatible with pen usage are based on point-and-click, e.g., pick-and-drop (Rekimoto, 1997) and take-and-put (Streitz et al., 2001). These techniques, however, cannot be used to access material on a display unit not providing pen support (Scenario 1).

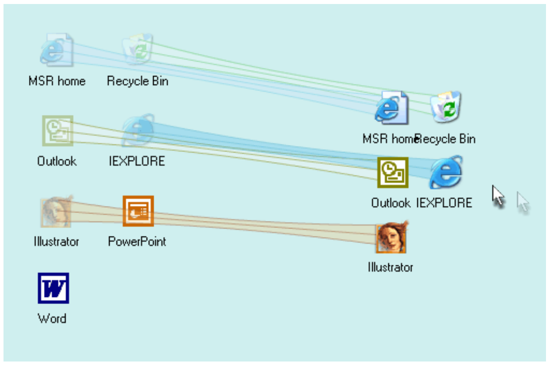

Figure 3: Drag-and-pick makes *all* icons in the direction of the mouse motion come to the cursor.

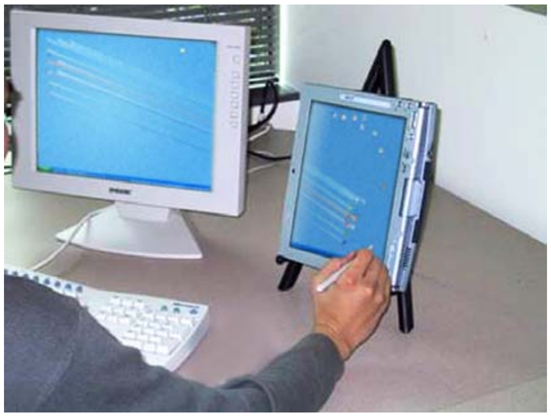

Figure 4: Drag-and-pick allows users to temporarily move icons from an external monitor to the tablet where the user can interact with them using the pen.

A different set of interaction techniques have been proposed to help users overcome large distances (Scenario 3). Manual And Gaze Input Cascaded (MAGIC) pointing (Zhai et al., 1999) uses eye tracking to move the cursor to the target area, from where the user guides the cursor manually (which requires an indirect input device). Gesture input techniques allow selecting a target and a command in a single interaction and are generally compatible with pen input (Rubine, 1991). 'Throwing' allows users to accelerate an object with a small gesture; the object then continues its trajectory based on its inertia (Geißler, 1998). The imprecision of human motor skills has prevented throwing from being used for reliable target acquisition. Myers et al. (2002) used laser pointers to acquire targets on a Smartboard, but found them to be slower than touch input.

A variety of mouse-based interaction techniques use destination prediction to simplify navigation (e.g., Jul, 2002). Dulberg et al. (1999) proposed a flying click or flick for snapping the mouse to target locations. Swaminathan and Sato (1997) proposed making relevant controls on the screen "sticky".

As an alternative way of launching applications, today's operating systems offer menus containing lists of available application or documents. A 'send to' option (Microsoft Windows) allows sending an icon to a target selected from a predefined list. Compared to 2D desktops, which typically use a larger amount of screen space than pull-down or pop-up menus, menus are limited to a smaller selection of choices unless they use a hierarchical menu organization, which makes their usage less transparent and often less efficient. Furthermore, invoking a content-menu may require hitting a qualifier key, which can be problematic on touch-based systems.

4 Design and algorithms

In this section, we will take a more detailed look at the design and algorithms behind drag-and-pop/pick.

4.1 Selecting candidates

In order to reduce clutter, drag-and-pop creates tip icons only for a subset of the icons on the screen. Drag-and-pop's candidate selection algorithm is initialized with the entire set of icons on the screen; it then successively eliminates candidates using the following four rules.

First, icons of incompatible type are eliminated. If the user drags a text file, the icon of a text processor can create a tip icon; the recycle bin icon can create a tip icon; the icon of another text file, however, cannot, because dragging two text files onto each other is usually not associated with any behavior. Drag-and-pick bypasses this selection step in order to allow users to activate any type of icon.

Second, icons located between the cursor and the location where the tip icons cluster will appear (see following section) are eliminated. This rule avoids creating tip icons that move away from the cursor.

Third, only icons that are located within a certain angle from the initial drag direction (the *target sector*) are considered. The initial drag direction is determined the moment the user drags an icon further than a given threshold (default 15 pixels). During preliminary testing on a Smartboard, we got good results with first-time users when using sector sizes of ±30 to ±45 degrees. The sector size could be reduced to sector sizes of ±20 degrees as users gained more experience.

Forth, if the number of qualifying icons is above some hard limit, drag-and-pop eliminates tip icon candidates until the hard limit is met. Icons are removed in an order starting at the outside of the target sector moving inwards. This rule assures the scalability of drag-and-pop to densely populated displays, but requires drag-and-pop users working with densely populated screens to aim more precisely. We typically use hard limits between 5 and 10.

4.2 Computing the tip icon layout

Once tip icon candidates have been selected, drag-and-pop determines where on the screen to place the tip icons. In order to avoid interference between tip icons, the location of all tip icons is computed in a centralized fashion.

Our drag-and-pop prototype uses the following algorithm that is illustrated by Figure 5: (1) Snap icons to a grid and store them in a two-dimensional array, with each array element representing one cell of the grid. If two or more icons fall into the same cell, refine the grid. (2) Shrink the icon layout by eliminating all array columns and rows that contain no icons. (3) Translate icon positions back to 2D space by mapping the array onto a regular grid. By default, the output grid is chosen to be slightly tighter than the input grid, which gives extra compression.

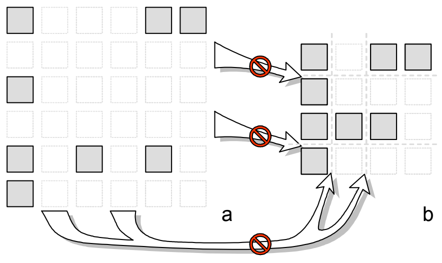

Figure 5: Drag-and-pop computes tip icon layouts (a) by snapping icons to a grid and then (b) removing empty rows and columns.

We chose this algorithm, because it preserves alignment, proximity, and spatial arrangement between icons, which allows users to use their spatial memory when identifying the desired target within the tip icon cluster. This is especially useful when tip icons look alike (e.g., a folder in a cluster of folders). In order to help users distinguish local icon clusters from surrounding icons more easily, the algorithm may be adjusted to *shrink* empty rows and columns during layout computation instead of removing them entirely.

After the tip icon layout has been computed, drag-and-pop positions it on the screen such that the center of the layout's bounding box is located at the direct extension of the user's current mouse motion. The distance of the tip icon cluster to the user's current cursor position is configurable. For inexperienced users, we got best results with distances of around 100 pixels; shorter distances made these users likely to overshoot the cluster. For more experienced users, we were able to reduce the distance to values around 30 pixels, which allowed these users to operate drag-and-pop with less effort, in a more "menu-like" fashion. In order to reduce visual interference between tip icons and icons on the desktop, drag-and-pop diminishes desktop icons while tip icons are visible.

4.3 The rubber band

When the tip icon cluster is displayed, users need to *re*-identify their targets within the tip icon cluster in order to be able to successfully acquire them.

Our first implementation of drag-and-pop created tip icons on top of their bases and used slow-in-slow-out animation (Shneiderman 1998) to move tip icons to their final location. While this approach allowed users to locate the final position of the desired tip icon by visually tracking it on its way from basis to final position, it also required users to either wait for the animation to complete or to acquire a moving target. We therefore chose to abandon the animation and immediately display tip icons at their final destinations.

In lieu of the animation, we provided tip icons with rubber bands. The design prototype of the rubber band is shown in Figure 6. For performance reasons, our prototype, which is shown in all other screenshots, uses rubber bands of a lower level of graphical detail, i.e., a tape and three lines in the color scheme of the corresponding icon.

The purpose of the rubber band is to offer the functionality of the animation, but without the problems alluded to above. The rubber band, decorated with the respective icon's texture, can be thought of as having been created by taking a photograph of the tip icon animation with a very long shutter speed (so-called *motion blur*, e.g., Dachille and Kaufman, 2000). Like the animation, the rubber band allows users to trace the path from base to tip icon. However, users can do this at their own pace and the customized texturing of the rubber band allows users to start tracing it anywhere, not only at the base.

The rubber band is provided with a narrow midriff section, suggesting that the rubber band is elastic. This design was chosen to help users understand that tip icons have retracted to their bases when they disappear at the end of the interaction. This feature may also help users find their way to the tip icon faster, because it provides users with a visual cue about how far away the tip icon is located. A thick rubber band section implies that the tip icon (or base) is close; a thin rubber band section indicates that the target is further away.

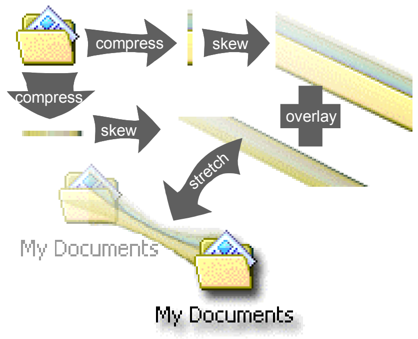

Figure 6: The motion blur textures on the rubber bands that connect tip icons with their bases are made by overlaying skewed copies of that icon.

4.4 Aborting drag-and-pop interactions

As soon as tip icons and rubber bands are shown on the screen, drag-and-pop waits for the user to acquire one of the tip icons to complete the ongoing drag-and-pop or drag-and-pick interaction. There are two cases, however, in which users will want to abort the interaction without acquiring a tip icon.

The first case is when the user dragged the mouse at a wrong angle so that the desired target icon did not pop up. In this case, the user may either drop the icon and try again or complete the interaction as a regular drag-and-drop interaction, i.e., by dropping the icon onto the target icon's base instead.

The other case occurs if the user is intending to perform a regular mouse drag operation, for example to rearrange icons on the desktop or to capture a set of icons using a lasso operation. For these cases, drag-and-pop allows users to terminate tip icons on-the-fly and to complete the interaction without drag-and-pop/pick. To abort, users have to move the mouse cursor away from the tip icon cluster while still keeping the mouse depressed. This can be done by overshooting the cluster or by changing mouse direction. In particular, this allows users to access the underlying drag-and-drop and lasso-select functionality by introducing a simple zigzag gesture into their cursor path. The zigzag contains at least one motion segment moving away from the tip icons, thus terminating tip icons as soon as they appear.

The algorithm: the tip icon cluster is kept alive as long as at least one of the following three rules is successful. The first rule checks whether the mouse cursor has moved closer to the center of at least one of the icons in the tip icon cluster. This rule makes sure that the cluster does not disappear while users approach their targets. The second rule checks if the cursor is in the direct vicinity of an icon. This rule provides tolerance against users overshooting a tip icon while acquiring it. The third and last rule keeps the cluster alive if the cursor is stationary or if it is moving backwards very slowly (up to 5 pxl/frame). This rule makes drag-and-pop insensitive to jitter. Figure 7 illustrates the resulting behavior.

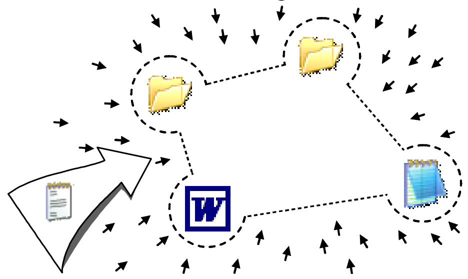

Figure 7: The tip icon cluster is kept alive as long as the user moves towards the cluster (arrows) or inside the convex hull surrounding the cluster (dashed).

5 User study

In this section, we report the results of a user study comparing drag-and-pop with the traditional drag-and-drop technique. To examine the effects of bezel-crossing as well as distance, as described in Scenarios 2 and 3, we chose to run the study on a tiled wall-size display. During the study, in which participants filed icons into folders or dragged them onto the icons of matching applications, we recorded the time and accuracy of these movements. Our main hypothesis was that participants would perform faster when using the drag-and-pop interface, primarily because it would avoid the need for crossing the bezels, but also because it would bridge the space to very distant icons more efficiently.

5.1 Desktop layout

To obtain a representative set of icon arrangements for the study, we gathered desktop screenshots from 25 coworkers who volunteered their participation (15 single, 6 dual, and 4 triple monitor users). Overall resolutions ranged from 800,000 pixels to 3,900,000 pixels (66% more than the display wall used in the experiment).

We clustered the obtained desktops by number of icons and arrangement pattern. Then we chose representatives from each of the three resulting main clusters for the study (Figure 8). The "sparse" desktop reflected the desktops of roughly two thirds of the participants. It contained only 11 icons, most of which were lined up in the top left corner of the screen. The "frame" desktop reflected the desktops of three of the participants. It contained 28 icons arranged around the top, left, and right edge of the screen. The "cluttered" desktop, finally, contained 35 icons that were spread primarily across the top and left half of the screen. Five participants had chosen this style of arranging their icons.

Icon layouts were stretched to fit the aspect ratio of the display wall used in the experiment. An area at the bottom right of the screen was reserved for the starting locations of the icons to be filed during the study (dashed shape in Figure 8).

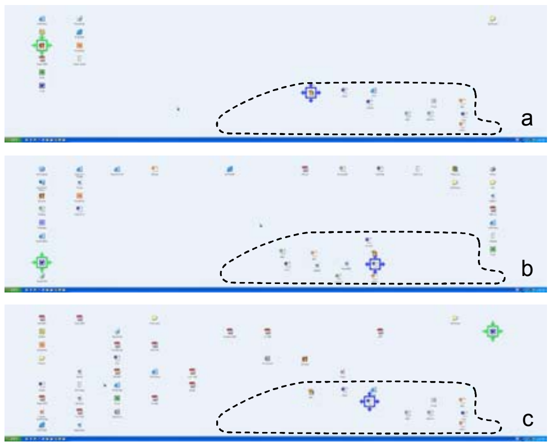

Figure 8: The (a) sparse, (b) frame, and (c) cluttered desktop layouts used in the study. The dashed line indicates the space reserved for the icons users had to file. Boxes around icons indicate icon to be filed and target.

5.2 Participants

Eight colleagues with no experience using drag-and-pop were recruited for this experiment. Due to technical problems, the data from one of these participants had to be dropped leaving us with 7. There were 2 female and 5 male participants ranging in age between 18 and 35. All were right handed with normal or corrected-to-normal vision.

5.3 Method

The test was run on the DynaWall (Streitz, 2001), a display wall consisting of three Smartboard units (Figure 9). Each Smartboard consisted of a back-projected 72"display with resistive touch input, so that the entire display was 15' (4.50m) long and 45" (1.12m) high. Display units could be operated by touching the display, but for easier handling participants were provided with color-free felt pens. Each of the three display units ran at a resolution of 1024x768 pixels, offering an overall resolution of 3072x768 pixels. The three display units were connected to a single PC equipped with two Matrox Millennium graphics cards and running WindowsXP. During the experiment, the DynaWall ran a simulated Windows desktop. We compared drag-and-pop to a control condition of drag-and-drop. Since our preliminary Windows-based version of drag-and-pop did not support the full functionality required for the study, we implemented a simulation using Macromedia Flash (www.macromedia.com). The drag-and-pop interface used in the experiment was configured to a ±30 degree target sector, 35 pixel target distance, and a maximum number of 5 tip icons.

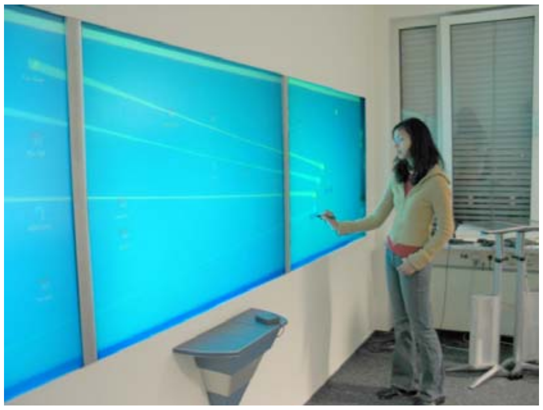

Figure 9: DynaWall setup used in user study

To each desktop layout we added 10 document icons in the lower right quadrant of the screen. These appeared in six different arrangements (Figure 8 shows 2 of them). The participants' task was to drag these icons into a given target folder or application. Icons of image files, for example, were to be filed in a folder labeled "My Pictures" and all Word documents should be dropped onto the Word application. To counterbalance for order effects, we required participants to file the documents in a randomized order. That is, for each movement, the item to be filed was highlighted along with the target icon. All other document icons were frozen, so that participants could only move the highlighted icon. As soon as participants began moving an item, all highlighting was removed, forcing participants to remember the destination item. We did this to assure that participants would have to re-identify tip icons when using the drag-and-pop interface, just as they would in a real-world task.

Participants were allowed several minutes to practice moving and filing icons in the prototype to get them accustomed to both the DynaWall display and the drag-and-pop interface. Once it was clear that users understood how to use the display and the interfaces, they were allowed to go on to the study. Participants filed 2 sets of icons for each interface (drag-and-pop and control), for each of the three desktops. Thus participants filed 2 x 10 icons x 2 interface x 3 desktops for a total of 120 movements.

To mitigate learning effects associated with new desktop arrangements or a new interface, we omitted the first 5 trials for any desktop-interface combination from our analyses, yielding ~15 correct trials per cell or 90 movements per participant.

5.4 Results

5.4.1 Task performance

Task performance was evaluated through speed and accuracy measurements. Error rates were considerably larger for drag-and-pop than for the control (6.7% vs. 1%). We observed two things that made this type of error more likely in the drag-and-pop condition. First, in the drag-and-pop condition candidate targets were brought closer together, making it easier to accidentally drop an item on the wrong target. Second, because drag-and-pop targets had been translated away from their "home" location, participants would sometimes forget which item was in fact the target, especially if visually similar icons (e.g., other folders) had created tip icons as well.

All data analyses for movement times were performed on the median movement times for each participant in each condition to normalize the typical skewing associated with response time data. Summary statistics report the means of these times.

Target icons could be located in the same display unit as the icon to be filed, in a neighbor display unit, or in the display unit at the other end of the display wall, requiring users to cross 0, 1, or 2 bezels in order to file the icon. To test the effect of bezel crossing on performance, we ran a 2 (Condition) x 3 (Bezels Crossed) within subjects ANOVA on the median movement data. This revealed a significant main effect for condition, $F(1,6) = 18.2$, $p<0.01$. Collapsed across all distances, drag-and-pop was significantly faster than the control. There was also a significant main effect of bezels crossed, $F(2,12) = 19.5$, $p<0.01$; movement time increased as the number of bezels participants had to cross to get to the target icon increased. As hypothesized, we also saw a significant interaction between condition and number of bezels crossed, $F(2,12) = 15.2$, $p<0.01$. As seen in Figure 10, an increase in the number of crossed bezels resulted in only a small increase in movement time for drag-and-pop, whereas it had a huge effect for the control interface.

When no bezels had to be crossed, drag-and-pop appeared to be slightly slower than control, although follow-up t-tests showed that this difference was not significant, $t(6)=1.73$, ns. When 1 or 2 bezels had to be crossed, drag-and-pop was significantly faster than drag-and-drop ($t(6)=4.02$, $p<0.01$ & $t(6)=4.12$, $p<0.01$, respectively). With 1 bezel crossed, drag-and-pop was twice as fast as the control and with 2 bezels it was 3.7 times as fast.

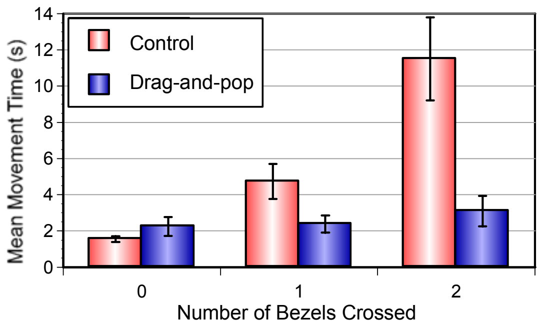

Figure 10: Mean movement time for control and drag-and-pop interfaces (± SEM).

Figure 11 shows a scatter plot of movement time versus target distance for both conditions. The best linear fit for drag-and-drop was $f(x)=0.007x-1.76$, $r^2=0.23$. The linear fit for drag-and-pop was $f(x)=4.19$, $r^2<0.0001$. This reinforces what can be seen in Figure 10—movement time increases with distance for the control interface, but stays relatively constant for the drag-and-pop interface.

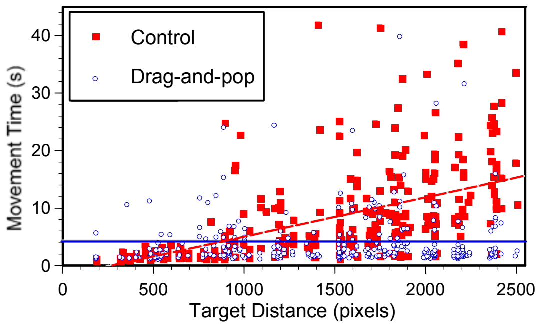

Figure 11: Movement time vs. target distance.

5.4.2 Questionnaire and subjective feedback

At the end of the study, participants answered a short questionnaire about their experience using the DynaWall and drag-and-pop. Participants were very enthusiastic about drag-and-pop. On a 7 point Likert scale (where 7=strongly agree and 1=strongly disagree), there was a mean > 6 for questions such as, "I liked using drag-and-pop", "I always understood what was happening when drag-and-pop was on," and "I would use drag-and-pop for large displays." There was a mean of less than 3 for "It took a long time to get used to drag-and-pop" and "It was hard to control what the targets did when drag-and-pop was on." Participants reported the drag-and-pop interface to cause less manual stress and fatigue than the control interface.

The most common problem with drag-and-pop was in getting the right group of targets to pop up, and several participants requested a wider angle for destination targets. This relates to an observation we made about how people interact with touch-sensitive wall-displays. On the wall display, participants had to employ their whole arm to make a movement, resulting in targeting motions in the shape of arcs. This means that the initial direction of the movement was *not* in the direction of the target. To accommodate such arcs in the future, we have adapted the target selection algorithm of drag-and-pop by giving the target sector extra tolerance for movements towards the top of the screen.

6 Conclusions and future work

The substantial time-savings found in the user study confirm our expectations. Although when used within a single screen unit drag-and-pop does not seem to by faster than traditional drag and drop (first pair of bars in Figure 10; drag-and-pop's capability of bridging distance to the target seems to be nullified by the need for re-orientation), its advantages on very large screens and its capability of bridging across display units are apparent. On the usability side, we were glad to see that participants had no trouble learning how to use the technique and that they described the technique as understandable and predictable. The single biggest shortcoming, the target selection, is the subjects of current work. In addition to the changes described above, we consider dropping the notion of a fixed target sector size and replace it with a mechanism that adjusts the sector size dynamically based on the number of matching targets.

Given the recent advent of commercially available tablet computers, our next step will be to explore how drag-and-pop and especially drag-and-pick can help tablet computer users work with external monitors. While this paper focused on icons, we plan to explore ways of operating menus, sliders, and entire applications using the techniques described in this article.

Acknowledgments

Thanks to Diane Kelly, Dieter Böcker, Lance Good, Amanda Williams, and LDUX. Work done at IPSI was supported by a grant from Microsoft Research.

References

Beaudouin-Lafon, M. (2000) Instrumental Interaction: An Interaction Model for Designing Post-WIMP Interfaces. In *Proc. CHI '00*. pp. 446–453.

Dachille, F. and Kaufman, A. (2000) High-Degree Temporal Antialiasing. In *Proc. Computer Animation* '00, pp. 49-54.

Dulberg, M.S., St. Amant, R., and Zettlemoyer, L.S. (1999), An Imprecise Mouse Gesture for the Fast Activation of Controls. In *Proc. Interact '99*, pp. 375–382.

Elrod, S., et. al. (1992) Liveboard: a large interactive display supporting group meetings, presentations, and remote collaboration. In *Proc. CHI'92*, pp. 599-607.

Geißler, J. (1998) Shuffle, Throw or Take It! Working Efficiently with an Interactive Wall. In *CHI '98 Late–Breaking Results*, pp. 265–266.

Guiard, Y., Bourgeois, F., Mottet, D. & Beaudouin-Lafon, M. (2001) Beyond the 10-Bit Barrier: Fitts' Law in Multiscale Electronic Worlds. In *Proc IHM-HCI 2001*, pp. 573-587.

Johanson B., Hutchins, G., Winograd, T., and Stone, M. (2002a) PointRight: Experience with Flexible Input Redirection in Interactive Workspaces. In *Proc. UIST '02*, pp. 227–234.

Johanson, B., Fox, A., and Winograd, T. (2002b) The Interactive Workspaces Project: Experiences with Ubiquitous Computing Rooms, *IEEE Pervasive Computing,* 1 (2), 67-74.

Jul, S. (2002) Predictive targeted movement in electronic spaces. In *CHI'02 Extended Abstracts,* pp.626-627.

Myers, B., Bhatnagar, R., Nichols, J., Peck, C., Kong, D., Miller, R., and Long, C. (2002) Interacting At a Distance: Measuring the Performance of Laser Pointers and Other Devices. In *Proc CHI'02*, pp 33-40.

Rekimoto, J. and Saitoh, M. (1999) Augmented Surfaces: A Spatially Continuous Work Space for Hybrid Computing Environments. In *CHI'99*, pp. 378-385.

Rekimoto, J. Pick–and–Drop: A Direct Manipulation Technique for Multiple Computer Environments. In *Proc. UIST'97*, pp. 31–39, 1997.

Rubine, D. (1991) Specifying Gestures by Example. In *Proc. Siggraph'91*, pp. 329-337.

Shneiderman, B. (1998) *Designing the User Interface: Strategies for Effective Human-Computer Interaction.* Third edition. Reading MA: Addison-Wesley.

Streitz, N.A., Tandler, P. Müller-Tomfelde, C., Konomi, S. (2001) Roomware. In: Carroll, J.A. (Ed.), *Human-Computer Interaction in the New Millennium,* Addison Wesley, pp. 553-578.

Swaminathan, K. and Sato, S. (1997) Interaction design for large displays. In *Interactions* 4(1):15 – 24.

Wagner, A., Curran, P., O'Brien, R. (1995) Drag me, drop me, treat me like an object. In *Proc CHI '95*. pp. 525–530.

Zhai, S., Morimoto, C. Ihde, S. (1999) Manual And Gaze Input Cascaded (MAGIC) Pointing. In *Proc. CHI '99*, pp. 246-253.