Drop-and-Drag: Easier Drag&Drop on Large Touchscreen Displays

Sebastian Doeweling
SAP Research Center Darmstadt
Bleichstr. 8, 64283 Darmstadt, Germany
sebastian.doeweling@sap.com

Urs Glaubitt
Michaelisstr. 16A, 64293 Darmstadt,
Germany
urs@glaubitt.com

ABSTRACT
Large displays have been found to offer a number of benefits over average-sized desktop displays: They increase productivity in office settings, improve performance on spatial tasks and offer increased user satisfaction in several contexts. However, their physical dimensions can complicate drag&drop interactions for users, especially when touch or pen input is used. Existing approaches (e.g. push-and-pop) have addressed this problem for simple drag&drop operations, but fall short when it comes to more complex ones (e.g. dropping a target onto a currently hidden node of a file tree or a specific location on a digital map).

To address this issue, we propose drop-and-drag, an interaction technique which introduces fully interactive proxy targets and which allows the interruption and resumption of drag&drop operations. The results of a controlled experiment show that drop-and-drag is significantly faster than traditional drag&drop for sufficiently distant targets. Additionally, the results report improved user satisfaction when drop-and-drag is used, especially for complex drop targets.

Author Keywords
drop-and-drag, wall-size display, drag&drop, touch screen, interaction technique

ACM Classification Keywords
H.5.2 Information Interfaces and Presentation: User Interfaces—*Graphical user interfaces*

INTRODUCTION
Over the last few years, large interactive display walls as well as larger tabletop systems have gained attention in both research communities and the general public (especially the presentation of Jeff Han [10] at the TED[1] conference in 2006 or commercially available systems like Microsoft Surface [15]). In office settings, larger displays were found to generally improve performance and were preferred to average-sized displays by the majority of users [8, 6]. Furthermore, tasks that require spatial orientation were found to benefit from the use of large displays (e.g. [22]). This effect was found to be stronger when resolution is also increased [1].

Additionally, Hawkey et al. [12] report that the interaction setting defined by large touch-operated displays, i.e. users near to each other as well as near to the display, is most effective for collaborative tasks. This claim is backed by a number of field reports, e.g. [20] (office and research settings) or [13] (digital design studios).

However, as reported by e.g. Robertson et al. [19], there are also challenges that accompany especially the use of large displays: Discontinuities in the displayed image caused by bezels for installations formed by multiple screens/projectors; configuration problems when screens are added to or removed from multiple monitor installations; cursor tracking problems; problems with window and task management; and problems related to accessing distal information. As display size increases, distal access gains particular importance: large touch- or pen-operated wall displays may well confront the user with the unattainable challenge of interacting with elements beyond his/her physical limits; in multi-user settings spatial interference, both physical[16] and virtual[23], may be an additional challenge.

Drag&drop is a very common interaction with desktop applications that suffers from these limitations. Especially for touch-operated devices, which require the user to maintain physical contact[2] during the whole operation, drag&drop becomes more challenging: if contact is lost, an accidental drop is performed, canceling the operation or even leading to unintended system actions such as moving files to an incorrect folder. Consequently, drag&drop was found to be more error-prone with increasing distance on touch-screen displays [18].

While state-of-the art approaches such as drag-and-pop [3] or push-and-pop [7] already offer support for drag&drop interactions with simple targets, they fall short when taking more complex ones into account, i.e. drop targets that may require state modifications prior to the actual drop operation:

[1]Technology, Entertainment, Design

[2]strictly speaking, for some camera-based systems very close proximity is sufficient

A common scenario involving complex drop targets is copying a file from one file tree to another using drag&drop. In order to complete the operation, one must frequently modify the state of the target file tree, i.e. collapse or expand branches, to access the desired drop location. This has to be done either prior to or during the actual drag&drop operation. While the former is time consuming for remote targets on large screens, the latter is often burdensome, requiring the user to hover above a tree node for a given time in order to expand it. Furthermore, there may even be components such as digital maps which do not offer the possibility to modify their state during a drag&drop operation, at all.

To address this problem, we designed drop-and-drag with three major goals in mind:

- Simplify drag&drop interactions with distant targets.
- Support drop targets that require complex interactions prior to the actual drop operation.
- Design for minimal interference in multi-user settings.

The remainder of this paper is structured as follows: First, we present related approaches and state-of-the-art techniques for drag&drop on large displays. Then, we elaborate on the design of our approach, following a discussion of the set-up and results of a controlled experiment, which we conducted to analyze the performance and user satisfaction of drop-and-drag. Finally, we argue about the shortcomings and limitations of our approach, present our conclusions and propose further enhancements to our interaction technique.

RELATED WORK

A number of techniques have been proposed to ease the problems, which accompany drag&drop on large displays. We separate these into techniques that are not geared towards drag&drop interactions specifically, but aim at allowing more efficient access to remote objects on large screens in general and those that focus on drag&drop. All techniques were reviewed according to the following criteria. Although some requirements may be more relevant to some scenarios than to others, an ideal interaction technique will meet all of them, allowing optimal flexibility:

- support for easier interaction with remote drop targets.
- support for complex drop targets.
- support for direct input techniques, i.e. pen- and touch-input.
- minimal requirements on available hardware (ideally the technique should work on single-touch or single-pen systems with pressure or proximity detection).
- minimal overhead to invoke the drag&drop operation.
- minimal interference in multi-user settings.

The results of the review can be found in Table 1.

	easier interaction	complex targets	direct input techniques	minimal hardware reqs.	minimal overhead	minimal interference
Missile Mouse	+	o	-	n/r	+	+
Tablecloth	-	+	o	+	-	-
ScaleView Portals	+	+	+	+	-	o
Push-and-throw	+	o	+	+	+	o
Drag-and-pop	+	-	+	+	o	o
Push-and-pop	+	-	+	+	+	o
The Vacuum	+	+	+	-	o	-

Table 1. Review of state of the art techniques for drag&drop on large displays, according to the six categories introduced at the beginning of the related work section (abbreviated for this table). A rating of '+' has been given where the technique fits the requirement very well; 'o' indicates an average fit, e.g. there is no active support, but also no obstacle to the requirement; '-' is used, when the requirement is not met. 'n/r' (not rated) has been used for Missile Mouse and its requirements on hardware because support for direct input is missing there.

Techniques facilitating general access to remote objects

Missile Mouse, devised by Robertson et al. [19], is a technique which is targeted at very large displays - with, however, mouse input. Because its primary focus is accelerating the cursor with relative input devices, it does not lend itself very well to touch-operated devices.

Tablecloth [19] is another approach by the same authors, which allows the user to scroll the entire screen content for access to distant areas. However, there are two major issues when this technique is used for drag&drop:

- First, it is geared towards single-user multi-monitor setups and facilitates access to remote areas by scrolling the relevant part of the desktop to the central screen - this may cause conflicts in multi-user settings where different users may want to access different content.
- Second, in its current form, the user may either drag the desktop to access remote content or execute a standard drag&drop operation, but not both a the same time. Thus the user is effectively required to use standard drag&drop.

ScaleView portals is an approach presented by Bezerianos [5]. These user-created portals represent alternative views of any area of the screen. Not only the visuals of the original area are recreated, but all actions performed within the portal are also delegated to the remote area and vice versa. Objects can be passed to the remote area by dragging them into the portals. Thus, the portals reduce the distance in visual and motor space alike. However, as ScaleView portals are geared towards more persistent interaction with remote locations, the creation of these portals imposes significant overhead for ephemeral interactions like drag&drop. Thus, drop-and-drag takes up the idea of fully interactive portals to remote locations, but it modifies this idea for the context of drag&drop interactions.

Techniques which specifically address the challenges of drag&drop interaction on large displays

Push-and-throw, an approach proposed by Hascoët and Collomb [11], enables users to throw the object instead of dragging it. To counter the imprecision of human motor skills, a technique called trajectory snapping is integrated into push-and-throw. Trajectory snapping allows for minor inaccuracies by snapping the throwing trajectory to the drop target, if the drop target is not located on the trajectory but rather in close proximity. This technique can be expected to work well for simple, free-standing drop targets. However, it is questionable whether it is still applicable when multiple drop targets are clustered in a small remote area or when preceding interactions like expanding branches within a file explorer are required.

Baudisch et al.[3] describe a drag&drop extension called drag-and-pop which takes the opposite approach to push-and-throw. Upon the invocation of a drag-and-drop operation, so-called tip icons, which represent the original drop target, are shown around the drag source. The connection to the original target is made via a virtual "rubber band". In order to avoid confusion caused by displaying too many tip icons, the set of targets is reduced to those which accept the data of the drag source and which are located in the direction of the initial drag movement. However, as the authors note, "the most common problem with drag-and-pop was getting the right group of targets to pop up". Furthermore, the iconified versions do not allow complex interactions or precise positioning on the respective drop target.

Based on push-and-throw and drag-and-pop, Collomb et al. propose a drag-and-drop technique push-and-pop [7]. When starting a drag motion, a take-off area is created, surrounding the current cursor position. This take-off area is still a miniature version of the screen as it is in push-and-throw. However, it contains tip-icons of all targets accepting the dragged object, thus eliminating the risk of invoking the wrong set of targets. Additionally, the tip-icons are positioned within the take-off area relative to their locations on the screen. As a result the layout of the take-off area is always the same for a given drag source, independent of its location or the direction of the initial drag motion. This enables the user to perform drag-and-drop actions based on muscle memory, i.e. the user memorizes the particular movement routine due to frequent repetition. Similar to push-and-throw and drag-and-pop, push-and-pop does not allow for complex interactions with the available drop targets. However, the idea of projecting the relevant drop targets into the vicinity of the drag source while preserving the proportions of the original screen layout, is taken up in our approach.

Another approach, based on drag-and-pop, is The Vacuum, developed by Bezerianos and Balakrishnan [4] and geared towards pen-input. This technique addresses the problem of getting the right group of targets to pop up by introducing a flexible arc which draws all the objects in the area it covers towards the origin of the arc, i.e. the current pen position - to fit into the influence area, targets are reduced in size accordingly. The Vacuum can be used for both selection and drag&drop, depending on whether the user invoked it on empty screen space or on a specific object. While it does allow for complex interactions with remote targets, drag&drop operations with such targets will require the user to trigger two interactions: The first on empty screen space to manipulate the target, the second to perform the actual drop operation - an overhead that may be especially challenging for users unfamiliar with pen-input systems. Furthermore, The Vacuum is designed to use proximity information, i.e. a hover state, which is not available for a larger number of multitouch systems (e.g. those based on frustrated total internal reflection). Also, its clearly highlighted arc of influence may cause interference in multi-user settings.

THE DESIGN OF DROP-AND-DRAG

Based on the evaluation criteria introduced in the related work section, we derived the following design decisions for drop-and-drag (respective evaluation criterion in brackets):

- Easier interaction with remote targets will be facilitated via proxy targets, i.e. visually and functionally identical copies, of all compatible drop targets projected into the area the user can immediately reach[3]. To support quick orientation, an algorithm that preserves the relative spatial layout of the original drop targets will be used to place the proxy targets in the area of immediate reach. (*support for easier interaction with remote drop targets*).

- Proxy targets will be fully interactive and equal in size to the original. To allow for a maximum number of simultaneously displayed proxy targets at the same time, these targets consist only of the user interface components on which a drop can actually take place (e.g. only the file tree from an explorer window). Furthermore, the technique introduces a suspend mode for drag&drop operations to facilitate interaction with complex drop targets without burdening the user to maintain contact with the screen or additional hardware requirements[4] (*support for complex drop targets*).

- No assumptions will be made on whether there is pen or touch input (*support for direct input techniques, i.e. pen- and touch-input*).

- Only single taps (clicks) will be necessary to operate the technique (*minimal requirements on available hardware*).

- The technique will be invoked automatically once the user drags an object (*minimal overhead to invoke the drag&drop operation*).

[3]To identify possible drop targets, hard-wired solutions (i.e. mapping drop targets and drag sources at compile time) are possible (and in fact this approach has been used for the implemented prototype). However, there are also more flexible solutions based on the semantic annotation of user interface components [17]

[4](contrary to Kobayashi and Igarashi, who use a throwing metaphor [14], the pause is triggered via a simple drop on empty screen space in drop-and-drag)

- No visual connections to the original targets are used (*minimal interference in multi-user settings*).

Thus, drop-and-drag enables the user to conveniently interact with distal targets, reducing the source-target distance in visual and motor space alike, while preserving the full set of possible interactions with these targets.

Proxy Features

Created by cloning the visuals of the drop target, a proxy represents an alternative view of the drop target, similar to a ScaleView portal[5] with the drop target as its remote area. Additionally, all actions performed within the proxy are delegated to the actual drop target. If the visuals of the drop target change due to such a delegated action, the visuals of the proxy are updated as well. Thus, a proxy enables a user to access all functionality provided by the actual drop target. This includes common actions as clicking buttons, expanding and collapsing of trees or invoking menus, as well as more rare actions such as panning a map component to the desired location.

Once the user starts to drag an object, his/her options to complete the drag&drop operation using the appearing proxies are twofold:

- He/she can drop the dragged object on one of the proxies directly (given the target does not require any additional interaction),
- or he/she can pause his/her interaction by temporarily dropping the dragged object next to the proxies. All proxies will remain visible and a resume button on each of them will then be enabled. Then, the user can perform the necessary changes to the state of the target object (without the burden of maintaining the drag&drop operation) and continue his/her drag&drop interaction afterwards. This is done via the resume button which causes the next click to be interpreted as a drop. As the user might want to nullify the paused operation instead of completing it, another button is added to allow the user to cancel the whole interaction, causing all proxies to disappear.

An exemplary proxy target, using a file tree is shown in the steps 2 to 6 of the walkthrough in Figure 2.

Proxy Placement

After a dragging motion is started and all drop targets accepting the currently dragged object are determined, the locations of the corresponding proxies need to be computed. The layout algorithm that we employ for this purpose was designed with three requirements in mind:

1. A proxy should always be within the user's immediate reach.
2. Spatial arrangement, alignment and proximity between the actual drop targets should be preserved.
3. In order to reduce the need for reorientation, the proxy should be placed on the line of sight between the source and the actual target, allowing the user to look for it in the original direction.

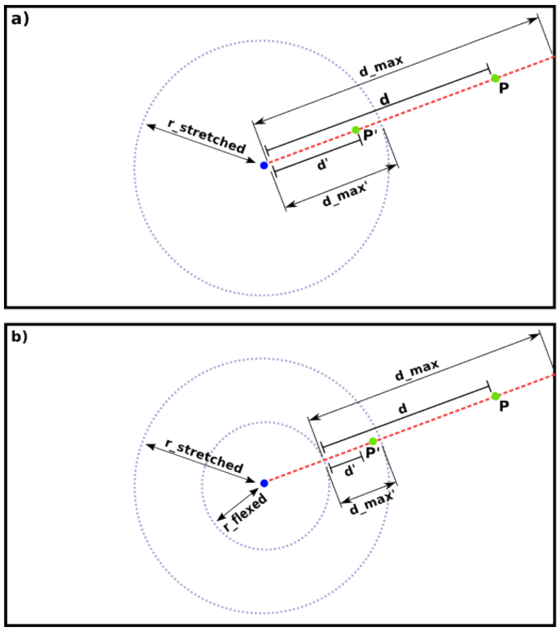

Figure 1. a) The initial version of the proxy positioning algorithm - proxy targets are projected nearer to the user, regardless of the original target's position.
b) The refined version of the algorithm places proxy targets directly over the original targets if they are located within the reach of the user's flexed arm. All other targets are projected into the area between the user's flexed and stretched arm's radius. This compensates for user's moving the dragged objects too fast and overshooting close-by proxy targets.

We implemented these requirements through an algorithm that projects the location of a drop target into the screen space the user can currently reach, i.e. the screen space defined by a circle with the user's arm length as its radius and the origin of the drag&drop operation as its center. Our first version of this algorithm handled all targets the same way:

A proxy is positioned such that $d/d_{max} = d'/d_{max'}$ holds true as illustrated in Figure 1(a). ($r_{stretched}$ denotes the distance defined by the user's stretched arm; d_{max} is the maximum distance from the drag source location to the edge of the screen; d'_{max} denotes the maximum projected length and is therefore equal to $r_{stretched}$. d is the distance between the drag source and the original (potential) drop target P. d' is the distance of the new proxy target position P', calculated by projecting the ratio d/d_{max} onto d'_{max}.)

However, preliminary tests revealed issues with this implementation. Users often expected proxies to appear with a reasonable distance to the origin of the drag&drop operation. When dragging very fast, they overshot the proxy before it became visible and then failed to notice it immediately, as they focused on the original drop target. Furthermore, users frequently dragged over a short distance before dropping on empty space. This allowed them to take full advantage of drop-and-drag, i.e. they could interact with the proxy without being forced to keep on dragging the object. Nearby proxies often interfered with this strategy by covering the relevant screen space.

To avoid these issues, we modified the algorithm by distinguishing between two kind of drop targets: those within the reach of the flexed arm and those beyond this distance. If a target is located within the reach of the user's flexed arm, its proxy is positioned right above it. All other drop targets are projected into the ring between the length of the user's stretched arm ($r_{stretched}$) and the length of his/her flexed arm (r_{flexed}), as illustrated in Figure 1(b). (d_{max} is the maximum distance from the drag source location to the edge of the screen minus $r_{stretched}$, d is the distance between the drag source and the original drop target P minus $r_{stretched}$, P' is the location of the projected drop target proxy. d' is the distance of the new proxy target position P', calculated by projecting the ratio d/d_{max} onto d'_{max}, which denotes the difference between the user's stretched and flexed arm's radius.)

This minimizes the amount of proxies located within very close proximity. As a result it is very unlikely that the user overshoots a proxy or that close-by proxies interfere with the intention to pause the drag&drop operation. At the same time it is still assured that every proxy is within immediate reach.

Our current algorithm can be applied to an arbitrary number of drop targets simultaneously. For a high number of large potential targets, this may lead to occlusion problems. This and other possible limitations of drop-and-drag are discussed later in the next section.

Exemplary drop-and-drag interaction

To illustrate the features offered by drop-and-drag, Figure 2 shows an exemplary sequence of actions, using drop-and-drag to move a folder from a newly attached portable flash drive to a folder currently not visible in the file explorer view of the user's computer:

1. The file explorer showing the user's computer is located in the upper left of the screen - the flash driver's file explorer has just opened in the lower right corner.
2. The user initiates a dragging gesture on the folder to copy the file - a drop target proxy, which shows the file tree part of the "computer-file explorer", is displayed close to the starting point of the dragging gesture.
3. As the the node/folder on which the user wants to drop the dragged folder, he/she pauses the drag&drop operation by dropping the dragged object on empty screen space.
4. Now he/she can conveniently interact with the file tree displayed by the proxy and expand the corresponding node.
5. By pressing the resume button at the lower left edge of the proxy, the user indicates that he/she wants to actually drop the dragged object.
6. Subsequently, he/she selects the now visible drop location. As a result, the dragged folder is copied to the folder

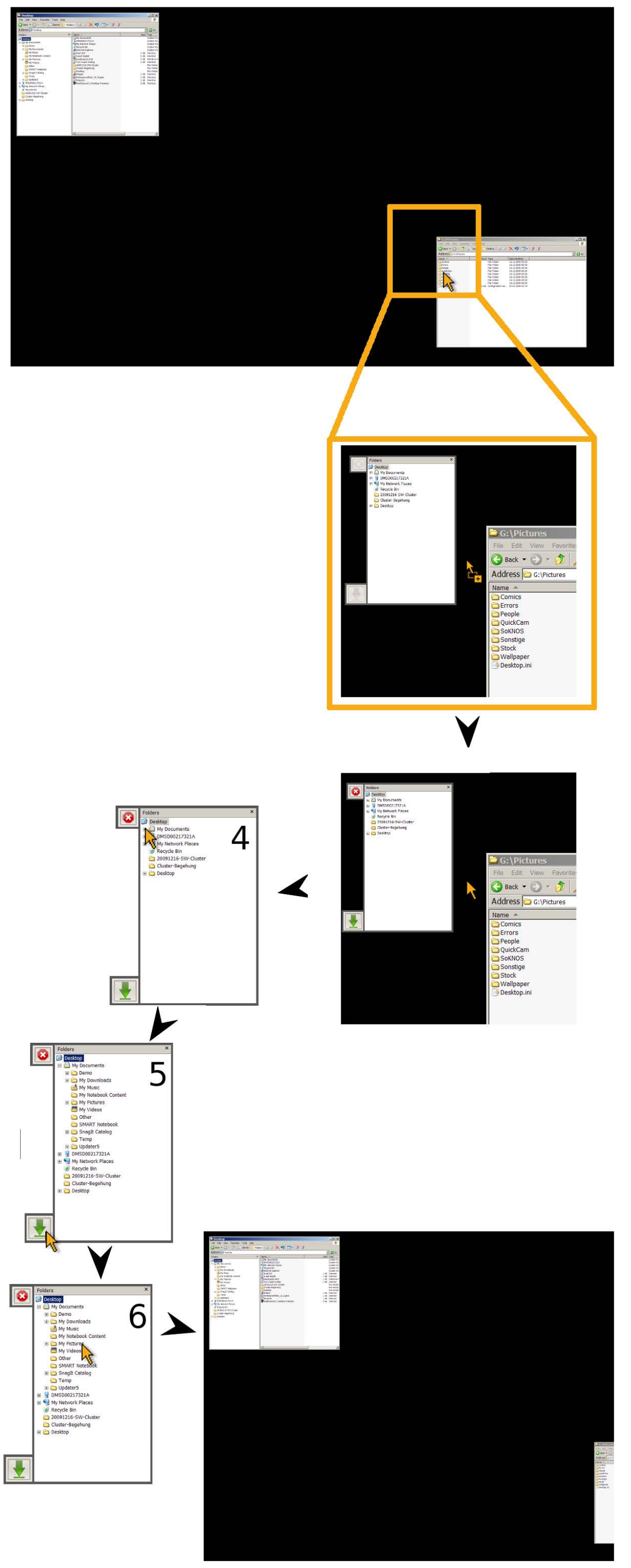

Figure 2. An exemplary drop-and-drag operation where pictures are copied from a flash drive to a sub-folder of "My Pictures".

within the file tree and the desktop returns to its initial state.

7. However, the state of the original file tree branch is now expanded as the state changes were forwarded by the drop target proxy.

EVALUATING DROP-AND-DRAG

To assess the general performance of drop-and-drag and to investigate possibilities for future improvements, we conducted a controlled experiment comparing drop-and-drag to traditional drag&drop (later, in the discussion section, we elaborate on the advantages and limitations of this choice). Participants of this study had to perform drag&drop operations using both techniques in two different scenarios involving complex and simple drop targets. We had five hypotheses:

H1 With increasing distances, drop-and-drag becomes significantly faster than traditional drag&drop.

H2 The effect of H1 is more distinct in case of complex drop targets.

H3 The average error rate of drop-and-drag does not exceed the error rate of traditional drag&drop. An operation is considered erroneous, if it is aborted or if the dragged object is dropped on a target, that was not in the correct state.

H4 For more distant targets drop-and-drag will be less error-prone than traditional drag&drop.

H5 drop-and-drag will exhibit a better user acceptance on large touchscreen devices.

Apparatus

The user study was performed on a Barco[2] display wall with a screen width of 300 cm and a height of 150 cm. The screen was placed 87 cm above the ground and was operated at a resolution of 4200 x 2100 pixels. Single touch input was provided by an array of infrared sensors, fabricated by SMART technologies [21]. Inherent to this technique for recognizing touch input two side effects affected our experiment:

First, a "touch" can be registered without physical contact to the screen. This sometimes confuses users who are unfamiliar with the device, as unintentional clicks are performed. Second, the accuracy of the touch input depends on the angle between the screen surface and the user's finger. The flatter the angle, the more likely other body parts, e.g. other fingers, the wrist or the elbow, are registered by the infrared sensors as well, resulting in a cursor position that does not correspond with the one intended by the user (cf. Figure 3). This problem most often occurs when acquiring distant targets, as the user tries to maximize his/her reach by aligning his/her finger with his/her arm. As a consequence, this behaviour frequently leads to very imprecise and sometimes unintentional touch input.

In order to avoid errors that are a mere result of this touch-recognition technique, participants were made aware of these issues. They were advised to touch the display perpendicularly if possible.

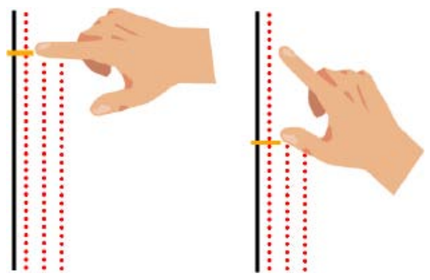

Figure 3. If the finger is not approaching the display surface perpendicularly, an unintended cursor position may occur.

Participants

Twenty-eight researchers and students from our lab volunteered to participate in our experiment. Five of them were female, twenty-three were male. Their ages were between twenty-three and forty. All but three participants were right handed. However, two of the three left-handed participants stated that they use the mouse with their right hand and were therefore treated as right-handed during the course of the experiment. None of the participants had any prior experience using drop-and-drag. They rated their experience in using touchscreen devices from 2 to 4 on a 5 point scale (1 no experience at all, 2 below average, 3 average, 4 above average, 5 extensive experience) with a mean value of 2.6.

Design

A within subject design with repeated measures and three independent variables was used. These variables were technique (traditional drag&drop, drop-and-drag), target distance (very close, close, medium, far) and target complexity (simple, complex). The target distance was not defined by static values, but rather in relation to the participants' arm length. 'Very close distance' targets were positioned at a distance of approximately 0.5 times the participant's arm length, 'close distance' targets at 0.9 times the arm length, 'medium distance' targets at 1.6 the arm length and 'far' targets at 2.5 the arm length. In case of 'simple' drop targets, the dragged object could be dropped without any prerequisites. 'Complex' targets, however, consisted of a panel containing two tabs. In order to perform a successful drop, the user was required to switch to the second tab (cf. Figure 4). In case of the traditional drag&drop technique, this forces a participant to reach for the target and select the second tab prior to the start of the actual drag interaction. By contrast, drop-and-drag enables the user to start the drag, perform a temporary drop anywhere on the screen, switch the tab on the displayed proxy and complete the operation via the resume button and a subsequent click on the actual target panel. Simple drop targets were 225 pixels wide and 210 pixels high. Complex drop targets were 24 pixels higher to accommodate for the additional tabs necessary to provide the tabbed pane. Thus, the area relevant to the drag&drop operation, i.e. the actual drop target, retained its size. Consequently, the size relevant to the index of difficulty according to Fitts' law remains constant throughout the experiment as well. In summary the experiment consists of 1792 samples (28 participants x 2 techniques x 2 target complexities x 4 distances x 4 repetitions).

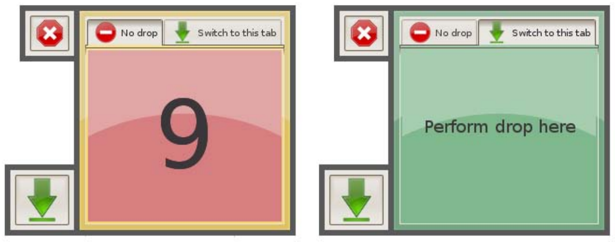

Figure 4. The two states of a complex drop target: In the left part of the figure, the target does not accept any drops. The right side shows the target, where the correct tab is selected and drops are accepted.

Test Procedure

Before performing any drag-and-drop operations, the participants were required to configure some user specific parameters before proceeding to the actual testing phase. Those parameters were:

- The height at which the participant could comfortably touch the screen while standing in front of it,
- the preferred hand, and
- the length of both the flexed as well as the stretched arm.

After completing the configuration phase, the screen layout was adapted accordingly. If the participant was right (left) handed, the source of all drag-and-drop operations was placed at the left (right) edge of the screen. This was done to generate sufficiently distant targets, but should not affect validity - for more realistic scenarios, the placement algorithm will adapt to the possibly range in the other direction. On the basis of the length of the participant's flexed and stretched arm, four sets of four targets each were generated (cf. Figure 5), one set of targets for each distance ('very close distance', 'close distance', 'medium distance' and 'far distance'). Consequently, only eight targets were within the user's immediate reach. The remaining targets required the user to walk towards them, as they had a distance of 1.6 to 2.5 times the user's stretched arm length to the drag source.

Users were asked to accustom themselves with the screen layout and the different set-ups for about five minutes in order to avoid learning effects during the experiment affecting the results. Subsequently, four test runs, i.e. every combination of both drag-and-drop techniques and complex or simple drop targets, were conducted for each participant. Every test run consisted of at least sixteen drag-and-drop operations, one for each target. In case of an error, i.e. the dragged object was not dropped on the drop target, the operation had to be repeated. For each step in each test run there was only a single valid drop target. Furthermore, the participants were asked to look for the target first, i.e. before they initiated the logged drag&drop operation by activating the source with a button click.

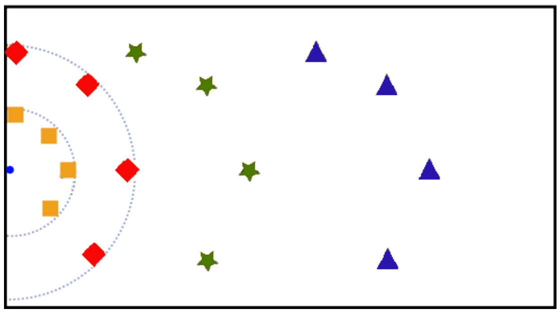

Figure 5. A schematic screen layout for a right handed person. Each set of targets is depicted by a different symbol: target 1-4 by a square, target 5-8 by a pentagon, target 9-12 by a star, and target 13-16 by a triangle. The reach of the participant's flexed and stretched arm is depicted by the two dashed circles.

To avoid the impact of exhaustion or learning effects on the results, the order of the test runs was permuted over the experiment. Additionally, the single operations during each of the test runs were performed in random order.

Results

To test our hypotheses, we analyzed task completion times, error rates and user ratings for the two experiment scenarios. Additionally we collected qualitative feedback during and after the test runs that gave hints to possible improvements of drop-and-drag.

Task Completion Time

Two-way repeated measures ANOVA tests revealed that distance significantly affected task completion times independently of technique and in both scenarios (simple targets: $F(3, 216) = 34.00, p \ll 0.01$; complex targets: $F(3, 216) = 34, 01$, $p \ll 0.01$). While the technique alone did not significantly affect the task completion times (simple targets: $F(3, 216) = 0.1$, $p = 0.76$; complex targets: $F(3, 216) = 3.02$, $p = 0.08$), a strong technique $\times$ distance interaction (simple targets: $F(3, 216) = 6.67$, $p < 0.01$; complex targets: $F(3, 216) = 17.25$, $p \ll 0.01$) indicates that drop-and-drag and traditional drag&drop are affected differently by changes in distance.

Distance	Simple Targets	Complex Targets
very close	$F(1, 54) = 15.88$ $p < 0.01$	$F(1, 54) = 29.58$ $p \ll 0.01$
close	$F(1, 54) = 4.80$ $p = 0.03$	$F(1, 54) = 15.98$ $p = 0.01$
medium	$F(1, 54) = 0.01$ $p = 0.91$	$F(1, 54) = 0.07$ $p = 0.79$
far	$F(1, 54) = 5.73$ $p = 0.02$	$F(1, 54) = 13.94$ $p \ll 0.01$

Table 2. Results of one-way repeated measures ANOVA tests. The nominal factor was technique (traditional drop&drag, drop-and-drag). The results show significant differences between techniques for 'very close', 'close' and 'far' distances.

The results of additional one-way repeated measures ANOVA tests presented in Table 2 support this: there was a significant difference between techniques for targets at ’very close distance’, ’close distance’ and ’far distance’. Only for targets at ’medium distance’ no significant difference was found (this can be attributed to the fact that drop-and-drag starts to outperform traditional drag&drop around this distance).

These findings are reinforced by an analysis of the task completion times of all participants. As illustrated in Figures 6 and 7, traditional drag&drop remains faster than drop-and-drag for targets in the user’s immediate reach; this is the case for targets at ’very close distance’ and ’close distance’.

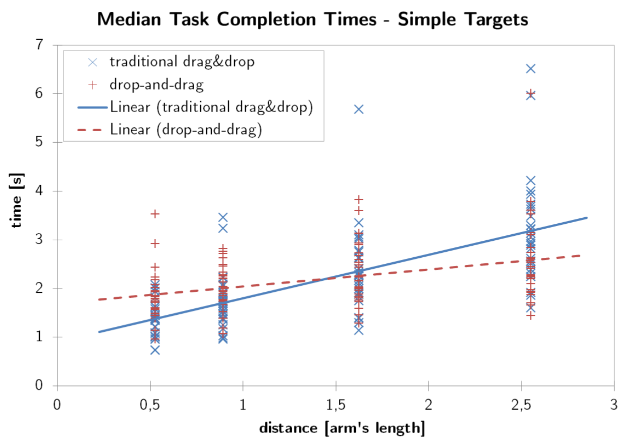

Figure 6. Task completion times of test runs with simple targets versus target distances in relation to arm length.

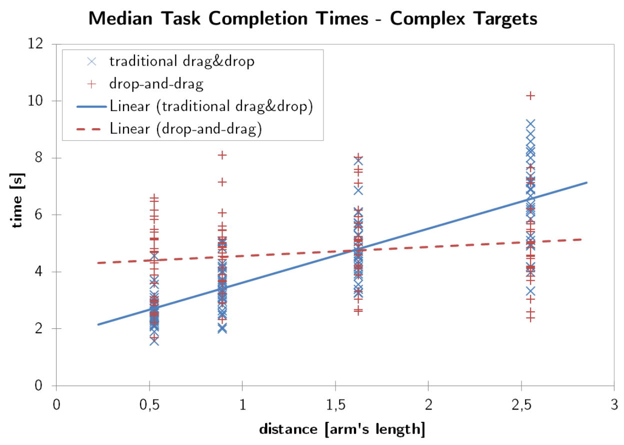

Figure 7. Task completion times of test runs with complex targets versus target distances in relation to arm length.

At a distance of approximately 1.6 times the user’s arm length, i.e. the distance of the ’medium distance’ targets, both techniques perform equally fast. When exceeding a distance of about 2 times the arm’s length, which was the case for the ’far distance’ targets, drop-and-drag clearly outperforms traditional drag&drop. For these targets drop-and-drag was on average 32% faster. Overall, these findings support the hypothesis that drop-and-drag becomes significantly faster than traditional drag&drop with increasing distances (*H1*).

Furthermore, we hypothesized that this effect is supposed

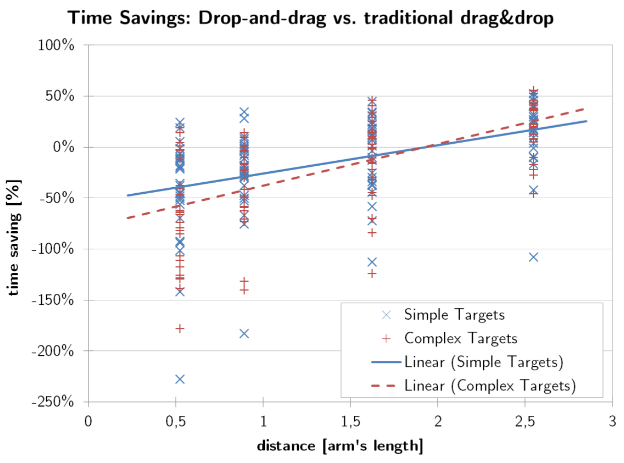

Figure 8. Relative time savings of drop-and-drag versus target distances in relation to arm length.

to be observed more clearly when drop targets require additional interaction to successfully complete the operation. Although drop-and-drag begins to be faster at approximately the same distance, regardless of target complexity, Figure 8 shows that the time savings when using drop-and-drag with complex targets exceed those for simple targets. We therefore consider our hypothesis *H2* confirmed.

Error Rates

Although error rates were expected to rise with increasing distance, no such correlation was observed. Overall error rates were relatively low. For simple targets, drop-and-drag performed better than traditional drag&drop; for complex targets, traditional drag&drop was slightly less error-prone (cf. Table 3). However, paired t-tests showed that both differences are not significant, due to a very high variance within our data coupled with a relative low overall error rate. The most frequent error when using drop-and-drag with complex targets was related to resuming a paused drag&drop operation. It seemed that despite the time the participants were given to familiarize themselves with drop-and-drag, some of them did not acquaint themselves with the concept of pausing and resuming an operation before completing some of the timed tasks. Summing up, *H4* is rejected by our results, *H3* is confirmed for the overall error rate, but the results for complex targets indicate room for improvement.

Targets	Drag&drop	drop-and-drag
Simple	8.57%	3.58%
Complex	10.62%	12.71%
Overall	9.60%	8.15%

Table 3. Mean error rates

Subjective Satisfaction

After completing all four test runs, the participants answered a short questionnaire in order to capture their subjective satisfaction concerning traditional drag&drop and drop-and-drag. Participants were asked whether the task was easy to fulfill using the different techniques and whether the respective techniques provided reasonable support - their ratings were captured on a five point Likert scale ("1 being "full disagreement" and 5 "full agreement").

While both techniques were perceived equally difficult in case of simple drop targets (the average rating was 3.4 for traditional drag&drop and 3.8 for drop-and-drag), drop-and-drag outperformed traditional drag&drop when operating on complex drop targets (average rating of 3.5 versus 2.4). Furthermore, only four of the 28 participants would still prefer traditional drag&drop when interacting with a very large touch-screen. This supports our expectation that drop-and-drag exhibits a better user acceptance due to the benefits it offers (*H5*). Further evaluation will however be necessary to test whether this hypothesis also holds against other supporting techniques for drag&drop (see next section).

DISCUSSION

While the the results of our experiment show that drop-and-drag performs well, especially for targets in medium (1.5 × user's arm length) and far (2.5 × user's arm length) distance, there is also a number of limitations in both the evaluation and our current design itself.

Proxy size and placement

As noted when elaborating on our design in design section, our current design will work for an arbitrary number of potential drop targets, where each target results in a proxy in equal size. For a reasonably large number of targets this may lead to occlusion problems: Important information could be hidden from the user and he/she might be unable to access essential interaction elements such as buttons or menus. However, an approach that scales down the proxy targets size (like The Vacuum [4]) may also be challenging, especially when touch input is used and thus precision is limited by the human finger. More refined techniques may include fish-eye view [9], but additional investigation should clarify the typical number of targets involved in drag&drop operations, first. Also, an analysis of the context users need to identify the proxy target (e.g. the surrounding explorer window for a file tree) could help to determine the necessary size.

Visual connections and interference

We deliberately decided not to include any visual connections between proxy targets and original targets, as we expected them to cause visual interference in multi-user settings. However, we noticed during our evaluation that some participants would reorient during the drag&drop operation, even though proxy targets were placed on the line of sight between drag source and original target. Thus further investigation on the interference caused by visual connections is necessary for multi-user settings. It may be reasonable to compromise and introduce lightweight connection visuals like the rubber-bands of push-and-pop[7].

Evaluation against standard drag&drop

While the evaluation of our technique against standard drag&-drop allowed a basic assessment of its performance, it allows comparison to state of the art techniques only to a very limited degree. Even though there is no support for complex drop targets in push-and-pop[7], we plan to re-implement this technique for future evaluations as, besides this, it meets most of our other requirements for ideal drag&drop techniques for large touchscreens very well.

CONCLUSION & OUTLOOK

When designing drop-and-drag, we aimed for a technique which makes drag&drop operations on large touchscreen displays both faster and easier, especially when more complex state manipulations are required. The findings of the controlled experiment show that this goal was met.

Although traditional drag&drop exhibits faster performance when for targets within immediate reach, the ability of drop-and-drag to efficiently bridge large distances, even when additional interaction complicates the drag&drop operation, is apparent (time-savings of up to 32% compared to traditional drag&drop). Furthermore, user satisfaction was improved notably: During their sessions multiple participants stated they felt more comfortable with drag&drop operations that use drop-and-drag rather than traditional drag&drop.

Yet, there is room for improvement: Qualitative feedback from some study participants indicated a number of refinements of our technique. Two participants suggested that employing animated transitions when creating the drop target proxies could improve the usefulness of drop-and-drag. Adding a rubber band as in push-and-pop would also be an option, but was not proposed by participants. Further on, additional research will be necessary to determine whether these visual cues cause interference in multi-user settings.

More controversial was the way the drag operation is resumed. While one participant explicitly stated that he liked the possibility to complete the operation with two clicks, two others stated that they would prefer to resume the operation by another drag interaction, which would originate from a scaled-down version of the original source, placed at the current position of the resume button. The latter suggestions might also offer an improvement to the unexpectedly long time it took for some participants to get accustomed to pausing and resuming drag&drop operations.

Furthermore, as pointed out during the discussion of the limitations of drop-and-drag, future work will need to include refinements to the the current positioning algorithm which may have occlusion problems for a larger number of targets.

Finally, while drop-and-drag did outperform the traditional technique, a more extensive comparison will also need to include state-of-the-art approaches like push-and-pop[7], at least for the case of simple drop targets.

ACKNOWLEDGMENTS

We would like to thank all those participating in our experiments for their time and effort. Also we would like to thank both our colleagues and the reviewers for giving valuable advice on how to improve this paper and the presented interaction technique.

The work presented in this paper has been partly funded by the German Federal Ministry of Education and Research.

REFERENCES

1. R. Ball and C. North. Effects of tiled high-resolution display on basic visualization and navigation tasks. In *CHI '05: Extended Abstracts of the SIGCHI conference on Human factors in computing systems*, pages 1196–1199. ACM, 2005.
2. BARCO. Barco control rooms. `http://www.barco.com/controlrooms/`. last accessed: 2010-05-11.
3. P. Baudisch, E. Cutrell, D. Robbins, M. Czerwinski, P. Tandler, P. T, B. Bederson, and A. Zierlinger. Drag-and-pop and drag-and-pick: techniques for accessing remote screen content on touch- and pen-operated systems. In *INTERACT '03: Proceedings of the IFIP TC13 Conference on Human-Computer Interaction*, pages 57–64. Springer, 2003.
4. A. Bezerianos and R. Balakrishnan. The vacuum: facilitating the manipulation of distant objects. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 361–370. ACM, 2005.
5. A. Bezerianos and R. Balakrishnan. View and space management on large displays. *IEEE Computer Graphics and Applications*, 25(4):34–43, 2005.
6. X. Bi and R. Balakrishnan. Comparing usage of a large high-resolution display to single or dual desktop displays for daily work. In *CHI '09: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1005–1014. ACM, 2009.
7. M. Collomb, M. Hascoët, P. Baudisch, and B. Lee. Improving drag-and-drop on wall-size displays. In *GI '05: Proceedings of the Graphics Interface conference*, pages 25–32. Canadian Human-Computer Communications Society, 2005.
8. M. Czerwinski, G. Smith, T. Regan, B. Meyers, G. G. Robertson, and G. Starkweather. Toward characterizing the productivity benefits of very large displays. In *INTERACT '03: Proceedings of the IFIP TC13 Conference on Human-Computer Interaction*, pages 9–16. Springer, 2003.
9. G. W. Furnas. Generalized fisheye views. In *CHI '86: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 16–23. ACM, 1986.
10. J. Y. Han. Low-cost multi-touch sensing through frustrated total internal reflection. In *UIST '05: Proceedings of the ACM symposium on User interface software and technology*, pages 115–118. ACM, 2005.
11. M. Hascoët. Throwing models for large displays. In *HCI '03: Proceedings of the British HCI Group Annual Conference*, pages 73–77, 2003.
12. K. Hawkey, M. Kellar, D. F. Reilly, T. Whalen, and K. M. Inkpen. The proximity factor: impact of distance on co-located collaboration. In *GROUP '05: Proceedings of the SIGGROUP conference on Supporting group work*, pages 31–40. ACM, 2005.
13. A. Kahn, J. Matejka, G. Fitzmaurice, G. Kurtenbach, N. Burtnyk, and B. Buxton. Toward the digital design studio: Large display explorations. *Human-Computer Interaction*, Volume 24(1 & 2):9–47, 2009.
14. M. Kobayashi and T. Igarashi. Boomerang: suspendable drag-and-drop interactions based on a throw-and-catch metaphor. In *UIST '07: Proceedings of the ACM symposium on User interface software and technology*, pages 187–190. ACM, 2007.
15. Microsoft. Microsoft surface product information. `http://www.microsoft.com/surface/`. last accessed: 2010-05-10.
16. C. Müller-Tomfelde and C. Schremmer. Touchers and mousers: commonalities and differences in co-located collaboration with multiple input devices. In *CHI '08: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1149–1152. ACM, 2008.
17. H. Paulheim. Ontology-based Modularization of User Interfaces. In *EICS '09: Proceedings of the SIGCHI Symposium on Engineering interactive computing systems*, pages 23–28. ACM, 2009.
18. J. Rekimoto. Pick-and-drop: a direct manipulation technique for multiple computer environments. In *UIST '97: Proceedings of the ACM symposium on User interface software and technology*, pages 31–39. ACM, 1997.
19. G. Robertson, M. Czerwinski, P. Baudisch, B. Meyers, D. Robbins, G. Smith, and D. Tan. The large-display user experience. *IEEE Computer Graphics and Applications*, 25(4):44–51, 2005.
20. D. M. Russell, J. P. Trimble, and A. Dieberger. The use patterns of large, interactive display surfaces: Case studies of media design and use for blueboard and merboard. In *HICSS '04: Proceedings of the Hawaii International Conference on System Sciences*, 2004.
21. SMART Technologies. Smart technologies. `http://www.smarttech.com/`. last accessed: 2010-05-10.
22. D. S. Tan, D. Gergle, P. Scupelli, and R. Pausch. With similar visual angles, larger displays improve spatial performance. In *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 217–224. ACM, 2003.
23. T. Tsandilas and R. Balakrishnan. An evaluation of techniques for reducing spatial interference in single display groupware. In *ECSCW'05: Proceedings of the European Conference on Computer Supported Cooperative Work*, pages 225–245. Springer, 2005.