DGP-TR-2004-002

July 21, 2004

Interaction and Visualization Techniques for Very Large Scale High Resolution Displays

Anastasia Bezerianos, Ravin Balakrishnan

Interaction and Visualization Techniques for Very Large Scale High Resolution Displays

Anastasia Bezerianos, Ravin Balakrishnan
Department of Computer Science
University of Toronto
anab | ravin @dgp.toronto.edu
www.dgp.toronto.edu

ABSTRACT
Interaction at close proximity with wall sized interactive displays presents interesting interface challenges in that not all parts of the display are easily visible or reachable by a user without significant physical movement. We address this challenge by developing interaction and visualization techniques for bridging distances and organizing content to facilitate easy access to all displayed information. These include techniques for bringing proxies of further-away parts of the screen toward the user for viewing and interaction; portal widgets that support visualization of, and interaction with, alternate views of portions of the virtual canvas; and transient storage of items in unused portions of the screen. Interaction with proxy items in the relevant widget areas is either mediated by, or functionally identical to, direct interaction with the original items on the main virtual canvas, allowing for seamless transitions between the two for a fluid overall user experience.

Keywords: large displays, interaction techniques, distance reaching, alternative views

INTRODUCTION

Interaction with large format displays have long been of interest to the research community, with much of the early research focusing on single whiteboard sized displays [15, 16]. More recently, the rapidly decreasing cost of projectors have spurred research in the construction of much larger wall sized displays by tiling multiple projectors to form a single virtual image [1, 7, 17]. These multi-projector large displays are particularly interesting from an interaction perspective in that the high resolution provided by the tiling of multiple projectors enables users to view high quality imagery even when they are up-close to the display. In contrast, single projector systems at that scale would not be suitable for up-close interaction as the image would appear too pixelated. With a few notable exceptions [2, 9, 21], much of the research in this area has focused on the issues surrounding hardware and projector registration [1, 7, 17] and rendering over clusters [10]. If we are to use these displays in the highly interactive manner for which they are well suited, we need to address the interaction challenges that arise due to their ability to display vast quantities of data over a very large spatial canvas.

Unlike interaction on a desktop or even a small whiteboard sized display where almost all displayed items are within arms reach of the user, data on wall sized displays often reside farther away, or in an unreachable location (e.g., higher than the user can reach). From a visualization perspective, it can be difficult for a user to view all parts of the screen at equal clarity, since some of the display will appear in the user's peripheral vision. As a result, if existing user interfaces are mapped onto displays of this scale for up-close interaction, they would at the very least require the user to walk around the display to accomplish even simple tasks, or they may be unusable altogether when, for example, the user can't reach the top of the display to operate an application's menu bar. Admittedly, one could always operate such a display from afar, using a mouse and a keyboard, but we believe that such an approach does not fully leverage the potential benefits that can accrue with up-close direct interaction.

In this paper, we explore the design space of very large scale interaction, and present the design and implementation of a set of interaction and visualization techniques that attempt to address some of the challenges. Although the design of our techniques is driven by our focus on direct up-close interaction with high resolution wall sized displays (Figure 1), the techniques could also be beneficial for smaller or lower resolution systems.

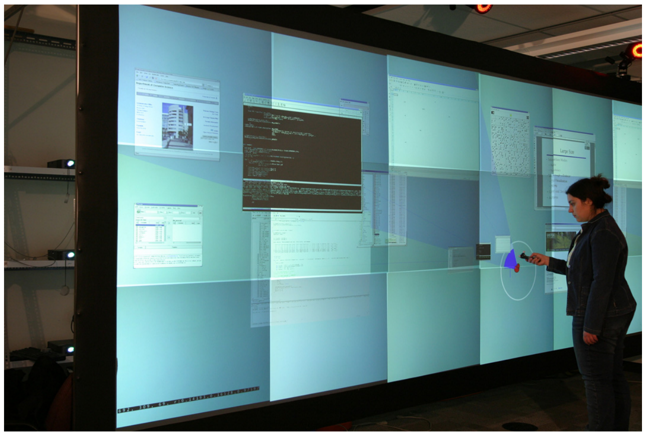
Figure 1. High resolution large scale interaction.

RELATED WORK

Tivoli [16] is one of the first applications developed for a whiteboard sized display. The main focus was on content structuring for meeting tasks, but it does identify potential problems arising from directly applying existing interfaces to larger displays. Flatland [15] is another application for whiteboard displays that concentrates on content management, but presents ways to create free space and overlap data in a flipchart metaphor.

Swaminathan and Sato [22] discuss various configurations for creating large displays and identify problems that arise due to scale, including pointer movement and control challenges over large distances. They propose using a dollhouse metaphor, where a small scale model of the display and its contents is used to specify pointer movement in the large display. Guimbretière et al [9], in addition to techniques for content creation and placement, introduce ZoomScapes which are regions of the screen with different zoom levels. Objects crossing ZoomScapes are scaled correspondingly around the center of movement of the user in a continuous manner. Baudisch et al [3] present focus plus context screens, where users can perform detailed tasks on a small high resolution screen, while at the same time visualizing contextual peripheral information via a surrounding large low resolution screen. Streitz et al [21] describe techniques for connecting multiple displays together and moving objects between them.

There has also been significant research in the area of reaching across distances in large interaction surfaces. The pick-and-drop [18] and take-and-put [8] techniques provide ways of moving content from one location of a screen to another or to a different screen entirely. The shuffle [8] and flick [25] techniques provide ways of sending content quickly to a remote location, covering a specified distance or reaching the edge of the interactive surface respectively, while the throwing technique [8] is similar but user configurable. The drag-and-pop and drag-and-pick [2] techniques allow the user to move an icon or cursor towards potential targets at the far reaches of the screen via proxies that are brought close to the current cursor position.

More recently, several papers have described interesting techniques that allow users to define alternative views of work areas. Scalable Fabric [20] enhances existing window managers by using an area around the main focus of the user to scale down windows and create user defined task related groups. Selection of a task group in this area brings it into focus, replacing the previous group occupying the main screen. WinCuts [23] augments window managers by allowing users to acquire and interact with alternative views of arbitrary regions of existing windows.

In short, our survey revealed significant research in the area of large display interaction, with much of it focusing on relatively small whiteboard sized large displays. Our work, which focuses on up close interaction with much larger and higher resolution displays, builds upon much of this previous work in interesting ways.

SYSTEM HARDWARE and SOFTWARE

Our interface is prototyped on a back projection screen tiled with 18 projectors at 1024x768 resolution in a 6x3 tiling (Figure 1). The screen is 16' wide and 6' high, with resolution of 6144x2304 pixels. The projectors were driven by a cluster of 18 workstations. Software was written in C++ with Chromium (http://chromium.sourceforge.net) providing graphics rendering over the cluster. Our techniques are designed for use with a touch sensitive display, with finger tapping providing a single button event. However, our screen is not yet touch enabled, so as a temporary measure we currently use a handheld wireless single button tracker whose position in absolute coordinates across the screen is tracked using a camera-based Vicon motion tracking system (http://www.vicon.com). Although our tracker could provide 6-dof position and orientation for the handheld tracker in 3D space, we restricted all our techniques to only 2-dof x,y screen coordinate positions operable with a single button since our ultimate goal is for these techniques to be as widely applicable as possible on standard touch enabled displays.

INTERACTION and VISUALIZATION TECHNIQUES

Overview

We present six techniques as a first step in dealing with the transition from manipulating information on desktop scale displays to very large wall-sized displays. Our techniques are designed to address fundamental issues in space management and remote access for interaction at this scale.

We assume a use scenario where a single user is working up close to the display, using a touch enabled screen (or a suitable replacement as in our prototype), without easy access to keyboards or other input devices for command input. To support this usage style, our prototype uses context sensitive marking menus [11] that popup at the user's input location for all command input. Our system has one global marking menu for selecting between techniques, and additional context specific marking menus attached to objects and interaction widgets. We animate all transitions in our interface in order to assist the user in maintaining context as they move from one operation to the next.

Vacuum Tool

The Vacuum tool enables quick access to items on areas of the screen that are either difficult or impossible to reach by bringing them to the user for viewing and manipulation. Inspired by the drag-and-pop technique [2], the tool acts as a "vacuum cleaner", bringing towards it items that reside inside an arc of influence centred about the widget and spanning the canvas.

Widget Design and Base Functionality

In designing the vacuum tool, we made two important design choices. First, we opted to allow the user to interactively control the parameters of the tool, including its position on the screen, the angle of the arc, and its start and end extents. Second, if the user moves the tool around, the effect of the tool is dynamically updated with items moving into the arc being brought toward the tool, and items leaving the arc removed from the tool's control.

In our current implementation, the Vacuum tool is designed as a circular knob, with three selectable parts (Figure 2). The entire tool can be moved on screen by clicking and dragging on the pin icon at its centre. This pin icon appears often in many of our widget designs, employing a "pinning" metaphor as a consistent indicator that the widget can be repositioned. Around the center a coloured arc is drawn, defining the vacuum's area of influence. The angle of the arc can be changed by simply clicking and dragging on it. A small white wedge at one edge of the arc allows the user to change the start extent of the arc. The actual area of influence of the vacuum tool on the rest of the screen is represented by a semitransparent overlay that sweeps out from the tool's centre.

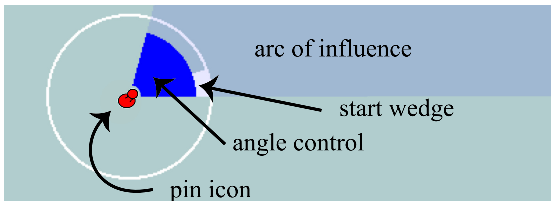

Figure 2. Vacuum widget. The user can click and drag on: the pin icon to reposition the widget, the start wedge to change the start extent of the arc, and the angle control to change the arc's angle of influence.

Apart from displaying and controlling the tool itself, we also need to represent the results of its effect on the virtual canvas as it pulls items inside the arc of influence toward the tool centre and thus the user. We initially experimented with simply displacing the vacuumed items entirely from their original locations on the canvas to the tool's centre. However, we found this to be disorienting and instead took the approach of bringing a *copy* of the vacuumed items to the tool. As items enter the vacuum's influence, we animate the movement of a copy towards the tool's centre, while a ghost image of the item remains at its original position. When an item leaves the vacuum's influence, the reverse animation takes place. This approach allows the user to maintain a sense of overall context of where everything is on screen, while being able to interact with vacuumed items at closer proximity.

In large scale displays it is important for the user to be able to keep track of state changes and for the canvas to be resistant to changes not initiated by the user. As such, when the vacuum tool is dismissed (in our prototype implementation, the tool is invoked and dismissed through the system wide marking menu), all copies of vacuumed items are faded away and the actual items reactivated in the positions they held prior to the vacuum tool's use. We do, however, allow the user to click on the vacuumed copies to keep them around as shortcuts after the vacuum tool is dismissed. These shortcuts are then grouped for future use. In general, the space reorganization and transformation caused by the vacuum tool is valid only for the duration of the tool's current invocation, unless the user performs an explicit action with the copied item(s).

Layout of Vacuumed Items

An important aspect of the design of the vacuum tool is in the representation of the vacuumed items when they are close to the tool's centre. The vacuum tool can collect many items from a large area of the screen, but must display copies of the items within the much smaller region near the tool's centre. For even a moderate number of items, it is impossible to display these copies at full size near the tool's centre without significant overlap in layout. We thus explored two different ways to layout the vacuumed copies around the tool .

In the first layout, called the *stacking vacuum* (Figure 3 & Figure 4), copies of the vacuumed items are stacked in full size, one on top of another with a slight offset, near the tool's centre. Items are stacked in the order in which they are vacuumed (i.e., beginning with items closest to the starting extent of the tool's arc of influence). This layout has the advantage of preserving the original size of the items, enabling easy perception of the copies. However, manipulation of the copies is hindered by the overlapping objects. Items have to be first selected and brought to the front, before interaction can take place. Note that we deliberately offset the items slightly in the stack in order to allow users to select items deep in the stack.

In the second layout, called the *scaling vacuum* (Figure 5 & Figure 6), copies of the vacuumed items are scaled down in size and displayed around the tool's centre, preserving the relative spatial relationships of items to one another. A semitransparent line connects the centers of the copies to the centre of the original items. This line virtually passes through the center of the tool as well, so all three points are aligned. This property of the scaling vacuum allows for the copied items to be selected with cursor movements that are identical in direction, but smaller in magnitude, to the movement required for selecting the original item. The advantage of the scaling over the stacking vacuum is that no additional overlap is introduced between vacuumed items, apart from overlap present in the original items' layout. The disadvantage is that the copies are significantly smaller, and thus harder to perceive and interact with in detail without first clicking on them to expand their size.

Discussion and Refinements

The stacking vacuum breaks down when the number and size of the vacuumed items is large. We could improve this by scaling down the stacked items, but this too has its limits. While crowding is less severe in the scaling vacuum, it too can get difficult to navigate as the space around the tool's centre gets populated with numerous copies of items.

Two refinements could alleviate these problems. A *semantic sensitive vacuum* that only vacuums items with certain characteristics (e.g., user interface elements) could reduce the object space significantly. A *spiral vacuum* is another alternative where a virtual knob would limit the outward extents of the arc of influence. As the knob is turned, the extents are changed, with the metaphor being one of the vacuum spiralling out from the tool's centre.

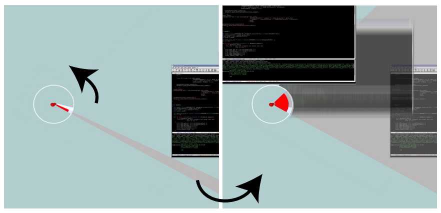

Figure 3. Close-up of stacking vacuum. As the arc is increased to include the window item (left image), a copy of the item is brought to the tool's centre (right image). The arc is colored red to differentiate it from the blue colored arc of the scaling vacuum. Motion blur in right image illustrates the animated transition that occurs during the item's movement.

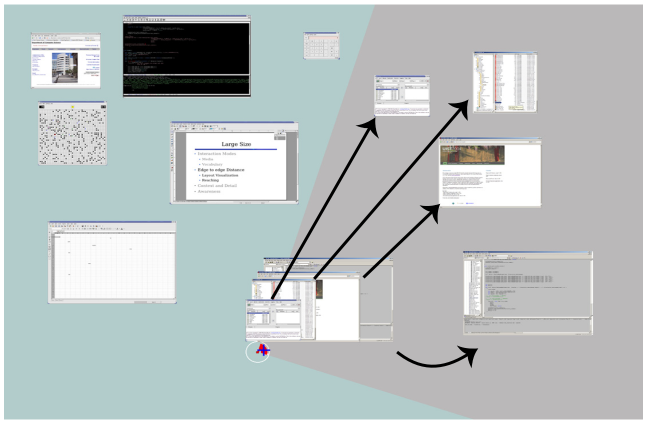

Figure 4. Full screen view of stacking vacuum. Black arrows added to illustrate relationships between original items and stacked vacuumed items, and are not part of the interface.

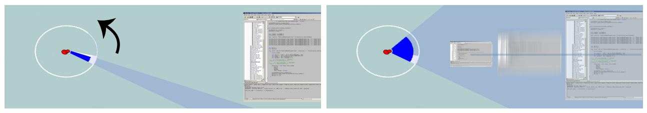

Figure 5. Close-up of scaling vacuum. As the arc is increased to include the window item (left image), a copy of the item is scaled down and brought to the tool's centre. (right image). Motion blur in right image illustrates the animated transition that occurs during the item's movement.

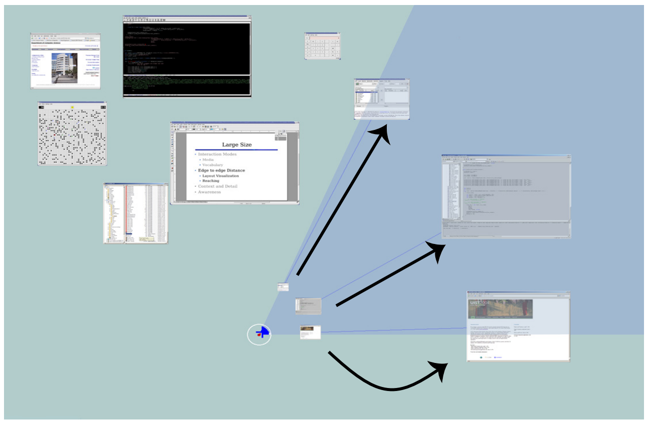

Figure 6. Full screen view of scaling vacuum. Black arrows added to illustrate relationships between original items and scaled vacuumed items, and are not part of the interface.

The idea of an arc of influence is also present in the drag-and-pop technique [2]. However, in that technique the angle of influence is fixed and the technique's behaviour different from our vacuum tool in that items of interest are brought to the user as she initiates a drag action towards a remote item. In contrast, the vacuum tool supports persistent vacuuming, and is intended to be used more as a tool that is invoked and used for a short period, rather than the very transient nature of the drag-and-pop technique. Furthermore, the vacuum tool can enable the vacuuming of any type of item, and subsequent viewing and manipulation of the vacuumed copies, whereas the drag-and-pop technique is designed specifically for drag and drop actions into desktop icons and as such does not have to be too concerned with the layout issues we addressed in our work. In a sense, the vacuum tool can be thought of as a generalized superset of the drag-and-pop technique.

Edge Reaching Tool

It is very likely that many existing applications will be run on large scale displays, with little change to their interface design. As noted in [16], when applications designed for desktop scale displays are run full screen on even moderately sized large displays such as whiteboards, the typical arrangement of user interface elements on the borders of the application window can pose major usability problems. On very large scale displays, the user may simply not be able to reach the top of the screen to access the menu and tool bars, may not want to walk all the way to either vertical edge to access tool palettes typically found at those locations, and may find it rather inconvenient to have to bend down to reach the icons at the bottom of the screen.

While the vacuum tool can enable easy reaching of items on the edges of the screen, its design was as a general purpose tool for access to selectable parts of the screen. Given the special and extensive use of edge regions by many applications, we have designed an *edge reaching* tool specifically suited to accessing edges of the screen.

The edge reaching tool divides the screen into a grid. For our 16' x 6' display, we have found a 3x2 grid to be appropriate for this tool. The tool acts as an interactive thumbnail of this grid. When the user clicks on parts of the thumbnail grid, scaled down copies of items within that section of the large screen slide down to surround the tool (Figure 7). These are scaled down to a predefined size, but the user can adjust the scale by pulling at the yellow bands that represent the edges of the screen. We could also restrict selections to items within the edge of the regions (i.e., yellow bands), making it a true edge-reaching tool. If semantic selections are enabled, then only the appropriate items (e.g., user interface elements) are brought to the tool.

Both the vacuum and edge reaching tools are similar in that they create shortcuts of items for the user. The edge reaching tool and scaling vacuum preserve the layout of the region of the screen brought to the tool, allowing particularly easy access to items in familiar screen layouts. The stacking vacuum however is more exploratory in nature. The stacking of items tends to facilitate attending to it peripherally until something attracts the user's attention. Although the edge reaching and scaling vacuum tools appear to be applicable in similar situations, the vacuum is more flexible in allowing fine tuning of the area of interest. However, although the edge reaching tool statically predefines areas of interest, multiple discontinuous areas can be active at a time (Figure 8), a feature the vacuum supports only if multiple instances of the tool are created.

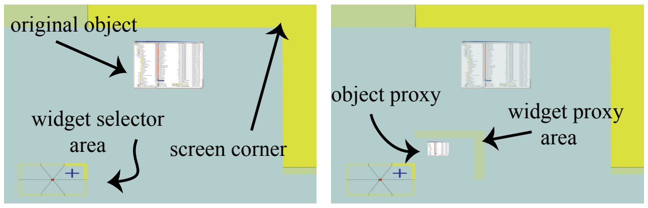

Figure 7. Using the edge reaching tool. (a) An area on the widget is selected. (b) Corresponding screen area is scaled down, with animated transition, into a proxy surrounding the tool. Items in the proxy can be used just like the originals.

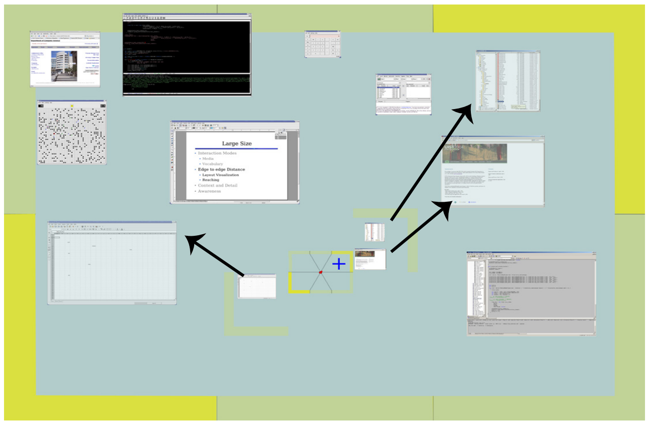

Figure 8. Full screen view of the edge reaching tool, with two regions selected. Black arrows are for illustration only.

Canvas Portals

In any display, portions of an effectively infinitely large virtual canvas can be invisible to the user depending on which part of the canvas is depicted on the screen at any given time. In large scale displays, this problem is somewhat mitigated in that a larger part of the virtual canvas can be shown on the display at any one time. However, the trade-off here is that some of this larger display is not easily visible when the user is up close to the screen. To explore solutions to this problem, we have developed *Canvas Portals*, which are widgets that provide alternate interactive views of the canvas (Figure 9).

Widget Design and Base Functionality

The basic functionality of a canvas portal is that of a magic lens [6] to a sub area of the virtual canvas. It appears as a window-like widget on screen, and has three user controllable parameters: the focal point of the portal on the canvas, the scale factor of the portal which controls how much of the canvas around the focal point is mapped to the portal, and the position of the portal itself on the canvas.

A canvas portal can be repositioned on the canvas by clicking and dragging on its pin icon (Figure 9). The scale factor is adjusted via a FastSlider [13] invoked from the canvas portal's marking menu. The scaling transformation is always centered around the portal's focal point. We provide two ways of altering the focal point. The most direct and precise method is to click directly on the desired location on the canvas, after first selecting the "change focal point" item from the canvas portal's marking menu. This method, however, can be rather inconvenient if the desired focal point is on parts of the screen that are difficult to reach, or impossible to achieve if the point of interest is on parts of the virtual canvas that is currently not being displayed on screen. As such, we provide a second technique where the focal point is selected at a coarser granularity through a thumbnail representation of the entire virtual canvas (Figure 9). This thumbnail is attached to the top left corner of the canvas portal, and shows an iconic representation of the focal area as well as the position of the portal itself on the larger canvas. The user simply drags the icon representing the focal area around the thumbnail to reposition the focal point. As the drag occurs, context is provided by highlighting the corresponding region on the main canvas with a semi-transparent overlay. This approximate way of changing the focal point has the advantage that the user can interact close to the canvas portal without having to move around the screen. It also allows the user to reach areas at the screen's extremities or areas of the virtual canvas not visible on screen.

While the positioning, focusing, and scaling functionality of the canvas portal builds upon previous work [6], we have designed significant additional functionality, as discussed in the following subsections.

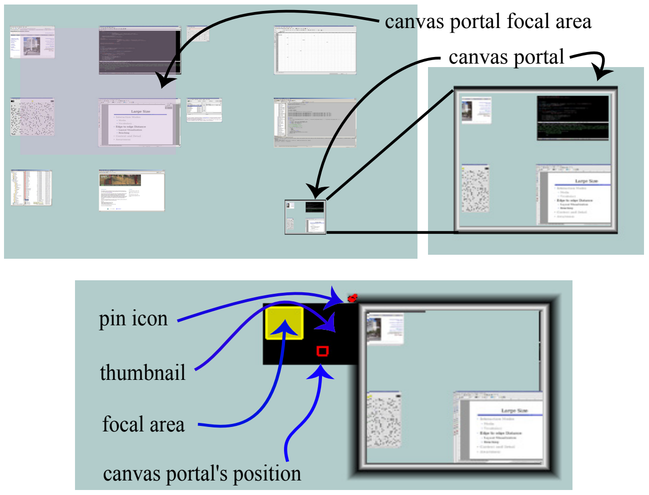

Figure 9. Canvas portal. (top) Contents of canvas portal correspond to a specified focal area on the virtual canvas. (bottom) Close-up of canvas portal showing components of thumbnail used for repositioning the focal point, as well as the pin icon used for repositioning the portal itself.

Interaction within Canvas Portals

Interaction inside the portal is equivalent to interaction on the entire screen. Every event occurring in the portal is transformed to the main canvas coordinate system. Thus the user can operate on a zoomed-out canvas portal to organize material on the main canvas, drag over large distances by increasing the control gain in the canvas portal, or get an overview of parts of the main canvas at a glance. Using a zoomed-in canvas portal the user can manipulate in detail an object that may appear small on the virtual canvas. The user can also focus the portal at remote areas where she might want to send events, for example in a full-screen application she can position the focal area of a canvas portal at a remote tool pallet.

Canvas Portals to Main Canvas Attachment

In our application we allow the user to pan the entire virtual canvas across the display screen. We therefore allow the user to "pin" a canvas portal to a particular area of the canvas, as opposed to an area of the screen. Thus, even if the virtual canvas is panned, the view of the canvas portal persists. For example, if the user focuses and pins a canvas portal on a group of windows, the canvas portal remains focused on the group as the virtual canvas is panned. Unlike WinCuts [23], canvas portals are not attached to specific windows or parts of them. This has the disadvantage that when a window is hidden behind others on the canvas it is similarly hidden in the canvas portal.

Transitioning between Canvas Portals and Main Canvas

Unlike regular magic lenses, canvas portals support the passing of objects back and forth between the portal, the main canvas, and other portals (Figure 10). If a user is moving an object on the main canvas and the center of movement (cursor pointer) crosses a canvas portal border, the object gets transitioned into the portal and continues its movement inside that canvas portal's coordinate system. The inverse also holds. The center of movement of the user is thus the center of translation and scaling of the item, as it is in ZoomScapes [9] and ScalableFabric [20]. When parts of a moving item cross a CanvasPortal boundary, a translucent rendering of these parts is cast on the other side of the border, indicating that the movement can be extended outside the current working reference frame, which may be the main canvas or a canvas portal. The seamless transitioning of items between a canvas portal and the main canvas allow for quick rearrangement of items in and out of the portals, without always requiring a refocusing of the portal as in a regular magic lens. It can also enable interesting usage scenarios, particularly when multiple canvas portals are active simultaneously. For example, after interacting in detail with an object in a zoomed-in canvas portal, the user can push it into a second zoomed-out canvas portal that acts as a temporary space for working items without cluttering their main working area.

If a user working inside a canvas portal realizes that the portal's view is one that they would like transferred to the entire display screen, a selection from the portal's marking menu will warp the entire screen's view to match that of the canvas portal, centered at canvas portal's location. In this way, the canvas portal can act as way to specify viewpoint transformations for the entire screen.

Grouping of Multiple Canvas Portals

We currently provide grouping of multiple canvas portals into a stacked (Figure 11a) or side-by-side layout (Figure 11b). Switching between the canvas portals in a stacked group can be achieved by either touching one of the constituent portals, which brings that portal to the top with an animated flipping transition; or by expanding the stack into the side-by-side layout where a new portal is selected by simply clicking on it.

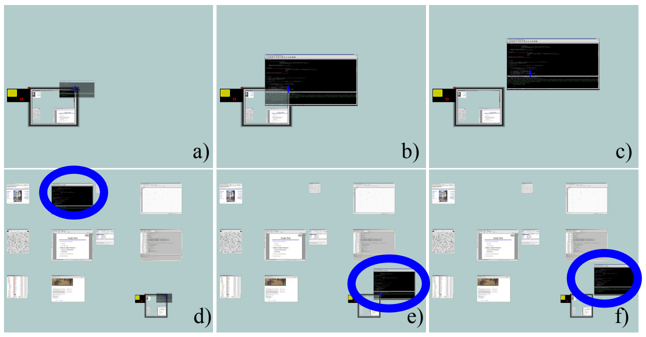

Figure 10. Transitioning items between canvas portals and main canvas. (a-c) are close-up views of canvas portals in corresponding overview images in (d-f). (a & d) An object is selected in the portal and moved beyond the top right corner of the portal. (b & e) Centre of object crosses the portal boundary, and object is displayed with the viewing transformations of the main canvas, and is moved from its original location on the main canvas to the new location. (c & f) Object is completely out of the portal.

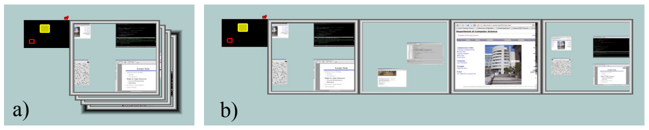

Figure 11. Grouping of multiple canvas portals. (a) Stacked layout. (b). Side-by-side layout. Clicking on any of the portals brings it to the top in the stacked layout.

Discussion and Refinements

With canvas portals the user is given the option of adjusting the input gain as desired to facilitate selection. Thus remote targets, or targets of small size, can be easily selected using zoomed-out or zoomed-in canvas portals respectively.

In our system we assume that our screen is touch sensitive and that all interaction occurs up close to the screen. However, when interacting with large scale high resolution displays, it is quite possible that users may at times want to interact from slightly further away, perhaps using different input modalities. While moving away from the screen will allow users to get a better overview of the screen without requiring widgets specifically designed for providing such overviews, the functionality provided by canvas portals can still be useful. For example, having additional views in canvas portals could allow for less or faster context switching when moving between tasks within an application or between different applications. Also, canvas portals can provide detail and overall views of the virtual canvas simultaneously, which is useful at any scale of interaction. Finally, the benefit of being able to group items and move them seamlessly across different scales and views is also realizable even when interacting at a distance.

Having multiple canvas portals active can lead to difficulty in distinguishing which canvas portal corresponds to which focal area on the virtual canvas. To mitigate this, each time a transition occurs (in the canvas portal stack or when a portal is moved, etc) we display a connecting line between the canvas portal and its focus area. Nevertheless, as the number of portals increases, several of them may point at similar areas of the virtual canvas, making it difficult to tell these portals apart. In future versions, we intend to explore labelling the portals and having a tabbed layout of labels similar to that available in some desktop window managers.

Window Portals

Window portals are a variant of canvas portals that provide quick access to, and switching between, application windows on a large scale display (Figure 12). The main overview region of the window portal is functionally equivalent to a canvas portal with a focal area covering the entire virtual canvas (i.e., it acts as a zoomed-out view of the entire canvas). When a user clicks on an item in the overview region, the item is expanded and displayed as an active object next to the overview region, and is marked with a red border to indicate its special status. The user can interact in detail with the selected object or select a new one. As the user selects different objects over time, a thumbnail representation of all previously selected objects is displayed in a timeline over the currently displayed object. Clicking on thumbnails in the timeline will turn the associated object into the currently active one. Thus, while the overview region of the window portal acts as a spatial locator of objects on the main canvas, the timeline region acts as a temporal locator for recently used objects.

Discussion and Refinements

On the Windows desktop the ALT-TAB key combination is used for alternating between windows. Given a display without keyboard, window portals provide similar spatial and temporal switching between active windows. Thus, we believe that it is applicable not only to large displays, but to smaller displays that lack an easily accessible keyboard, such as tabletPCs in a slate configuration, or whiteboard sized displays. Note that if we enable semantic filtering of items in the overview region of the window portal, we can restrict the display to for example only user interface widgets. Thus, we could create on-the-fly palettes of interface widgets that could be moved around the screen and operate multiple applications from a single locale.

Given that the window portal consists of a zoomed out view of the screen, issues relating to selecting and distinguishing between small targets arise. While semantic filtering can limit the selection space, the potential number of items (overlapping or not) can still be large and it does not address the issue of small targets to select from. Allowing dynamic zooming, or expanding targets [12], could be one potential solution worth investigating.

Overlapping windows on the canvas can also be problematic in window portals, as in any other multi window management system. Solutions to date include tabs and peeling [5], and multiblending [4]. We have implemented an alternate approach that "fans-out" a group of overlapping windows when a user clicks on any member of the group, in a manner similar to the widgets presented in [14, 24]. Items can be selected from this fanned-out display, or the group collapsed again. By default, we treat overlapping objects as a group. This enables not only the fan-out operation, but also permits moving of the entire group as a whole within the window portal. This grouping feature is currently implemented across all our widgets.

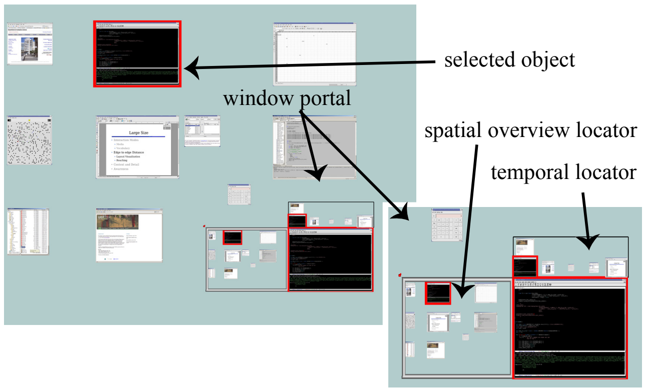

Figure 12. Window portals. (left) Window portal in the main canvas. (right) Close-up showing the overview region used as a spatial locator, the temporal locator timeline display, and the selected object. Red borders around the selected object in all views helps maintain context.

Division Bands

The vacuum tool, edge reaching tool, canvas portals, and window portals all provide alternative ways to view and access data while essentially preserving the overall view and layout of the main virtual canvas. Division bands (Figure 13) also provide alternate views of the virtual canvas, but unlike the previous tools, they temporarily disrupt the overall view of the virtual canvas. They act much like a cutting tool that virtually slices up the canvas along specified vertical and horizontal boundaries, and allow the cut portions to be dragged around to quickly reveal more or less of certain parts of the virtual canvas.

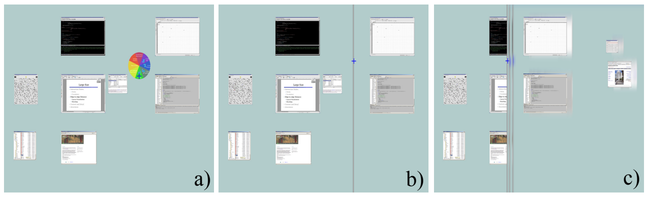

Figure 13. Single division band. (a-b) A new vertical band is created at the location of the marking menu. (c) Pulling the band to the left reveals more of the canvas on the right.

Using the global marking menu, the user specifies the cut position (the menu's invocation point) and the direction (direction of the mark used in the menu selection). We deliberately placed the cut down, right, left, and up commands in menu locations that would require selection marks in the corresponding directions, to facilitate a fluid combined specification of command and parameter. Once the cut position and direction is specified, a vertical or horizontal (depending on the specified direction) rod widget appears on screen. Based on the specified cut direction, one of the cut pieces of the canvas gets attached to the rod and a division band is thus created. For example, if the user specified "cut-right", a vertical rod appears with the right side of the canvas attached to the rod. This division band (i.e., the rod) is now attached to the user's pointer and can be moved around. If the user drags the division band left-right on screen, the attached portion of the canvas is expanded or shrunk, depending on the direction of movement. A quick flick-and-release motion of the pointer dismisses the division band. If the user releases the band without the flicking motion, then the division band remains on screen for subsequent reselection. Thus, users can use division bands to quickly drag a part of the screen towards them for viewing and/or manipulation. The ability to quickly dismiss the band with a flicking gesture allows for very transient quick views of remote portions of the screen, much like pulling on a spring loaded window-blind.

Multiple division bands of different orientations and directions can be positioned on the screen (Figure 14). Multiple pinned division bands of different orientations and directions are presented in a fixed ordering. Horizontal ones are placed on top of vertical bands since they are more likely to be followed by a permanent pinning action since we have found that they are most useful for creating new space on screen by dragging unused portions of the virtual canvas into view from the top or the bottom. In contrast, vertical bands tend to be used in a more transient manner.

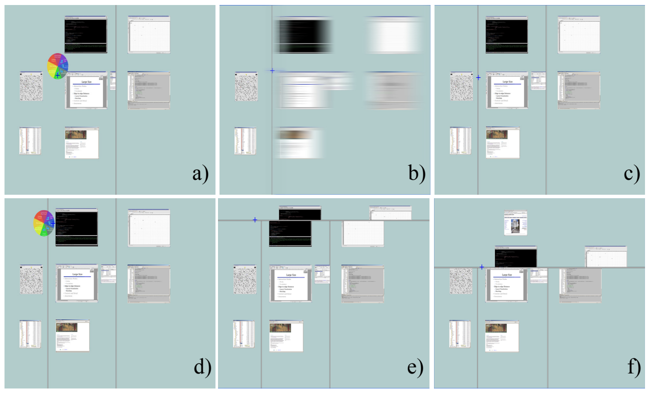

Figure 14. Multiple division bands. (a) An existing band is present. User invokes a menu to create a second band with same direction and orientation as existing one. (b) Second band is created and attached to cursor. As new band moves, its attached canvas covers the first band. (c) Second band is released and both bands are visible. (d,e) A third, horizontal, band is created. (f) Band is dragged downwards to reveal content previous hidden at the top of the virtual canvas.

Discussion and Refinements

The quick creation and dismissal feature of division bands makes them well suited for fast glancing actions at remote content, or parts of the virtual canvas not currently on screen. The pinning option enables them to also be used as persistent shortcuts to remote areas of the screen or as virtual desktops in a similar way to the flipcharts in Flatland [15].

Similar to canvas portals, our implementation allows objects to be moved between the main canvas and portions of the canvas attached to division bands.

When first created, division bands have the same zoom factor as the regular canvas, but this can be changed via an onscreen widget, allowing for the creation of a form of ZoomScapes [9]. We can have different bands each with their own zoom factors, allowing users to move items back and forth between different bands for different visualization effects. For example, objects can be stored in zoomed-out bands to save space, or brought into zoomed-in bands for detailed inspection.

A potential problem is the visual effect of virtually cutting up of the canvas. We believe this is not an issue for single user interaction, since the cutting originates from the user's own actions and is unlikely to cause confusion. However, if multiple users interact with the display simultaneously, division bands could prove to be confusing.

The Well

As observed by Guimbretière et al [9], even with very large displays users can still run out of space, and our experience corroborates this observation. However, we also observed that not all parts of the large display are viewed or used equally frequently. In particular, on our 6' high display, a user interacting at close proximity to the screen tends to focus on a region ranging roughly from waist level to just above eye level. As such, the lower part of the screen tends to be underutilized. When the lower part was used, it typically served as a place for storing objects users wanted to move out of the way. However, given that all users could do was to drag items in and out of this space, its usage was often sub-optimal and somewhat haphazard. As such, we felt that it would be useful to develop an explicit mechanism, called the *Well* (Figure 15), which allows for optimal and efficient use of this space for storage, and subsequent retrieval, of any onscreen object.

Objects are sent to the Well via a command invoked from its contextual marking menu. The object gets "dropped" vertically into the Well, and is positioned at the front of all other objects at that same vertical position. The other objects are moved further back in depth in the Well's 3D perspective rendering, thus providing a cue as to how recently objects were placed in the Well. The 3D perspective view allows for objects to be overlapped in depth while retaining some level of visibility. If the user needs to retrieve an object from the Well, she can select the object and restore it to its original location on screen using the "restore" command from the object's contextual marking menu. Alternatively, the object can simply be dragged onto any other location on screen. The entire Well can be cleared, and all stored objects restored to their original locations on screen, via a single command. The Well can be activated/deactivated at the user's discretion.

The Well provides similar base functionality as the taskbar in the Windows desktop in that objects can be minimized and subsequently returned to their original locations. It differs from the taskbar in that the minimized objects are displayed as miniatures of their original representations, rather than as text labels or icons. This enables the user to quickly recognize the objects with a quick glance, even from afar. Furthermore, the Well displays objects sorted according to recency, providing more context to the user. The DataMountain by Robertson et al. [19] similarly used a 3D perspective layout to organize items on a desktop display. Unlike the DataMountain, which was intended to be a sophisticated document management interface, the Well is used only as a temporary storage facility that works in concert with the rest of the display canvas. We also deliberately did not require users to explicitly sort or arrange objects in the Well, relying instead on an automatic arrangement in an effort to reinforce its use as a transient storage area with minimal operational overhead.

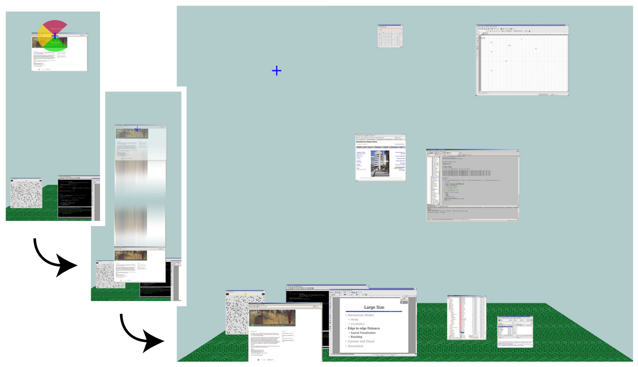

Figure 15. Well. Supports storage of items on the floor of the display, in a 3D perspective view. (a) "send to well" command invoked on an object. (b) Object moves to Well in an animated transition. (c) Object is displayed in front, pushing existing objects in the Well backwards in depth.

DISCUSSION and FUTURE WORK

Preliminary User Feedback

While we have not yet performed a detailed user evaluation of our work, we have obtained very preliminary feedback from four people who have explored using our techniques for about an hour. Two were familiar with user interface research, but the other two were naïve users. All users found the vacuum and edge reaching tools useful, and they understood their use and purpose almost immediately. Users also quickly grasped the benefits of the canvas and window portals, although given that we did not perform a task oriented study we cannot ascertain as to when users would choose to use them over the other techniques. Some users did not immediately understand the purpose or operation of the division bands, but after some explanation they found the quick glancing and spring-loaded releasing action quite useful and indicated that it helped refresh their view of the state of the entire canvas. All users understood the well technique, but did not show as much interest in it compared to other techniques. From this very preliminary feedback, it is clear that we need to improve the immediate understandability and usability of the division bands.

Extensions to the Techniques

In both the scaling vacuum and the edge reaching tool, changes in scale factor is globally applied to all selected items. It would be interesting to explore ways of performing semantic scaling, where different scale factors are applied to different classes of data, under user control.

It would be interesting to combine the flexibility of the vacuum tool with the edge reaching tool's ability to deal with discrete angles. One could imagine a configurable vacuum where the extents of the vacuum are determined not by a simple arc but by an arbitrary region that can be defined by the user by sketching the desired region or by dragging on handles on a default region.

Similarly, we could allow users to change canvas portals to use shapes other than the default rectangle. For example, more free-form shapes could be useful for selecting groups of items, or thin skinny rectangles could be useful for capturing tool or menu bars. Division bands could also potentially benefit from cuts beyond straight lines.

We have observed that horizontal division bands are used mostly to facilitate managing working and storing spaces, whereas vertical ones are most often used for quick glancing actions or fast shortcuts and their life on the canvas is more limited. While we have sorted the display of multiple division bands based on this observation, we intend to investigate patterns of use of the different direction bands in more detail and create a sorting and visualization mechanism better tailored to user behaviour.

The direct creation mechanism of division bands and their ephemeral nature leads us to believe they can become excellent snapshot mechanisms. They could be used to take state snapshots of the screen and act as a form of history of the virtual canvas. Visualization challenges of depicting and navigating groups of such "history bands" related to their spatial and temporal nature need to be solved.

Richer Information from Input Devices

Our work has focused on a usage scenario where the user interacts up close to the display with a touch sensitive screen capable of only two degree-of-freedom x-y input, with one button event. Many of our techniques, however, could be significantly extended if more sophisticated input devices or finger/hand gestures are used instead. Even a simple input enhancement, such as detecting finger hover over the surface of the screen, could expand the interaction vocabulary. For example, hover could be used to provide transient magnification of small scaled items in the vacuum or canvas portals. It could also help simplify the interactions that currently require continuous press-and-hold dragging actions, such as in division bands.

Supporting Multiple Users
Our research focused on single user techniques. The affordances of wall sized displays, however, make it inherently suitable for multi user interaction. While some of our tools could be used as is by multiple users simultaneously without interfering much with one another, others such as division bands that affect large regions of the screen have to be enhanced to support multiple users. This is one area that clearly requires significant future research.

CONCLUSION
We have presented a set of techniques aimed at facilitating direct up-close interaction with high resolution wall sized large displays. Some of these techniques, such as the vacuum and edge reaching tools, are tailored to reaching remote content on the screen and can act as exploration tools. Others, like the canvas portals, are mainly intended for layout arrangement and context switching. Nevertheless they can also be used for remote and fast reaching actions. Window portals act as shortcuts to specific items on the screen and can be viewed as a general window management tool with applications beyond large scale interaction. Division bands allow for space creation as well as a quick way to look at off-screen or hard to view content on the virtual canvas. Finally, the well technique allows for temporary storage in relatively unused parts of the screen. Overall, the ideas presented here can be viewed as building blocks for applications tailored to large displays, or as mechanisms for enhancing the usability of existing applications when they are run on large displays.

ACKNOWLEDGEMENTS
We thank Michael Wu, Daniel Vogel, Joe Laszlo, John Hancock, and members of the Dynamic Graphics Project lab(www.dgp.toronto.edu) at the University of Toronto.

VIDEO
Is available at the UIST submission site

REFERENCES
1. Baar, J.v., Willwacher, T., Rao, S., & Raskar, R. (2003). Seamless multi-projector display on curved screens. *Eurographics Workshop on Virtual Environments (EGVE)*. p. 281-286.
2. Baudisch, P., *et al.* (2003). Drag-and-pop and drag-and-pick: Techniques for accessing remote screen content on touch- and pen-operated systems. *Proceedings of Interact*. p. 57-64.
3. Baudisch, P., Good, N., & Stewart, P. (2001). Focus plus context screens: combining display technology with visualization techniques. *ACM UIST*. p. 431-440.
4. Baudisch, P., & Gutwin, C. (2004). Multiblending: displaying overlapping windows simultaneously without the drawbacks of alpha blending. To appear in *ACM CHI*.
5. Beaudouin-Lafon, M. (2001). Novel interaction techniques for overlapping windows. *ACM UIST*. p. 152-154.
6. Bier, E., *et al.* (1993). Toolglass and Magic Lenses: The see-through interface. *ACM SIGGRAPH*. p. 73-80.
7. Funkhouser, T., & Li, K. (2000). Onto the Wall: Large Displays. *IEEE CG&A*. 20(4).
8. Geibler, J. (1998). Shuffle, throw or take it! working efficiently with an interactive wall. *Extended Abstracts of ACM CHI Conference.*
9. Guimbretière, F., Stone, M., & Winograd, T. (2001). Fluid interaction with high-resolution wall-size displays. *ACM UIST*. p. 21-30.
10. Humphreys, G., *et al.* (2001). WireGL: A scalable graphics systemfor clusters. *ACM SIGGRAPH*. p. 129-140.
11. Kurtenbach, G., & Buxton, W. (1994). User learning and performance with marking menus. *ACM CHI*. p. 258-264.
12. McGuffin, M., & Balakrishnan, R. (2002). Acquisition of expanding targets. *ACM CHI*. p. 57-64.
13. McGuffin, M., Burtnyk, N., & Kurtenbach, G. (2002). FaST Sliders: Integrating marking menus and the adjustment of continuous values. *Graphics Interface*. p. 35-42.
14. McGuffin, M., Tancau, L., & Balakrishnan, R. (2003). Using deformations for browsing volumetric data. *Proceedings of IEEE Visualization*. p. 401-408.
15. Mynatt, E., Igarashi, T., Edwards, W., & LaMarca, A. (1999). Flatland: New dimensions in office whiteboards. *ACM CHI*. p. 346-353.
16. Pederson, E., McCall, K., Moran, T., & Halasz, F. (1993). Tivoli: An electronic whiteboard for informal workgroup meetings. *ACM CHI*. p. 391-398.
17. Raskar, R., Baar, J.v., & Chai, J. (2002). A low-cost projector mosaic with fast registration. *Asian Conference on Computer Vision (ACCV)*.
18. Rekimoto, J. (1997). Pick and drop: A direct manipulation technique for multiple computer environments. *ACM UIST*. p. 31-39.
19. Robertson, G., *et al.* (1998). Data mountain: Using spatial memory for document management. *ACM UIST*. p. 153-162.
20. Robertson, G., *et al.* (2004). Scalable fabric: A flexible representation for task management. To appear in *ACM CHI*.
21. Streitz, N., *et al.* (1999). i-LAND: an interactive landscape for creativity and innovation. *ACM CHI*. p. 120-127.
22. Swaminathan, K., & Sato, S. (1997). Interaction design for large displays. *Interactions*. 4(1). p. 15-24.
23. Tan, D., Meyers, B., & Czerwinski, M. (2004). Wincuts: Manipulating arbitrary window regions for more effective use of screen space. To appear in *ACM CHI*.
24. Vernier, F., Lesh, N., & Shen, C. (2002). Visualization techniques for circular tabletop interfaces. *Advanced Visual Interfaces*.
25. Wu, M., & Balakrishnan, R. (2003). Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays. *ACM UIST*. p. 193-202.