Fluid Interaction with High-resolution Wall-size Displays

François Guimbretière, Maureen Stone, Terry Winograd
Computer Science Department, Stanford University, Stanford, CA 94305-9035
Tel: 1-650-723-2780 Email: francois@cs.stanford.edu

ABSTRACT

This paper describes new interaction techniques for direct pen-based interaction on the Interactive Mural, a large (6'x3.5') high resolution (64 dpi) display. They have been tested in a digital brainstorming tool that has been used by groups of professional product designers. Our "interactive wall" metaphor for interaction has been guided by several goals: to support both free-hand sketching and high-resolution materials, such as images, 3D models and GUI application windows; to present a visual appearance that does not clutter the content with control devices; and to support fluid interaction, which minimizes the amount of attention demanded and interruption due to the mechanics of the interface. We have adapted and extended techniques that were developed for electronic whiteboards and generalized the use of the FlowMenu to execute a wide variety of actions in a single pen stroke. While this techniques were designed for a brainstorming tool, they are very general and can be used in a wide variety of application domains using interactive surfaces.

CR Categories: H.5.2 User Interfaces - Graphical user interfaces, Input devices and strategies, Interaction styles, Windowing systems; **I.3.6 [Graphics] Methodology and Techniques** - Interaction techniques;

Keywords: Large displays, interactive wall, FlowMenu

1. MOTIVATION

When people work collaboratively with a large collection of information, they often put it on a wall, where it is easy to view, annotate, and organize. In professional design settings, a project often has a dedicated room where related materials are spread out on every available surface. Drawings, photographs, diagrams, printed text, charts, spreadsheets, even bits of real objects can all be stuck on the wall. People can rearrange, annotate, and refine the collection to analyze it, create a design, or solve a problem (figure 1).

Technology to stick stuff on physical walls is well developed. Papers can be annotated by writing directly on them, or by applying Post-It^{tm} notes. As a project evolves, it is easy to rearrange the contents of the walls, regrouping and replacing items as needed. For these reasons, walls are the primary medium used during brainstorming, where ease of use and flexibility are at a premium.

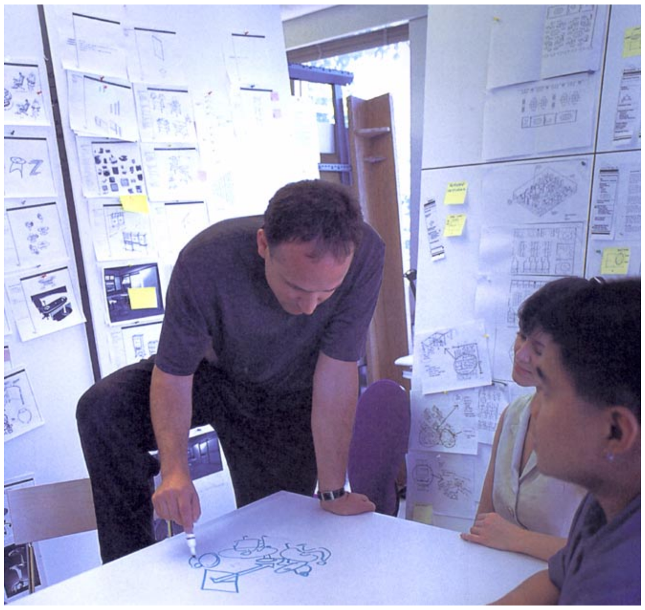
Figure 1: This design studio at IDEO illustrates the use of physical walls for displaying and working with large quantities of information. Note the diversity of information posted on the wall, from sketches to photographs to Post-It notes, and the abundance of printouts of digital media (courtesy of IDEO)

But a physical wall has limitations. Digital information needs to be printed out to be posted, and then it can't easily be updated. While designers like to use simple technologies such as scribbling and posting notes during the idea generation phase, they dread the post-brainstorm phase where ideas have to be sorted out, transcribed and archived. Handwritten materials are difficult to transfer after a meeting to any kind of permanent storage, and even more so to searchable digital archives.

Until recently, limitations of computer display screens have put the "stick it on the wall" approach out of reach for digital information. The advent of large, high-resolution displays is now making it possible to leverage this simple yet powerful metaphor. As computer displays move beyond the desktop to become commonplace on the walls around us, we have the opportunity to create new and better ways of

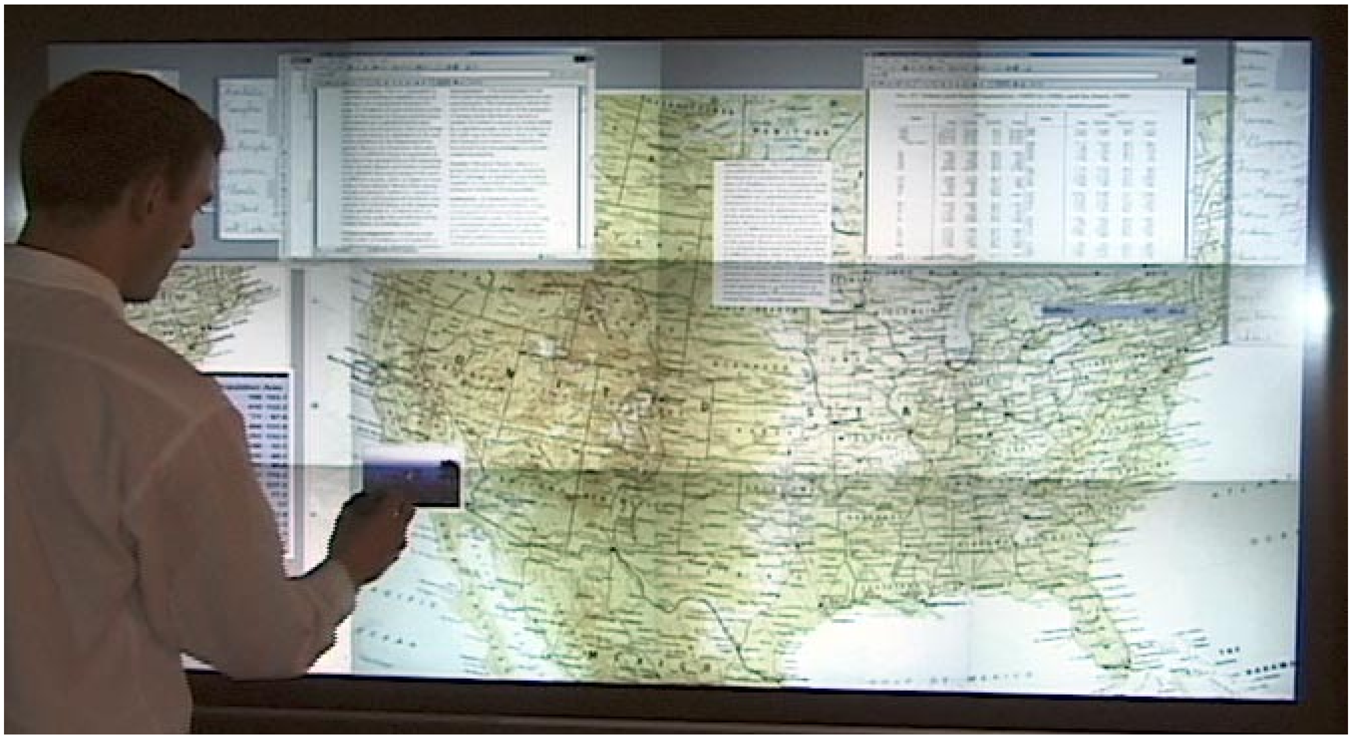

Figure 2: On this interactive wall, users can work with high resolution images, application windows, hand drawn material and information structures such as lists (upper right). The pen is used for all interactions including sketches and annotations directly on graphical objects or on transparent overlapping sheets.

interacting with a wide variety of information and applications. It is not enough to simply move existing GUI interfaces onto larger displays. Walls afford different kinds of interactions than workstations for several key reasons: The large visual display area can be used to work with large quantities of simultaneously visible material; interaction is directly on the screen with a pen-like device or touch, rather than with a keyboard and an indirect pointing device; and people often work in groups at a wall together, interleaving social and computer interactions.

We have developed an "interactive wall" interaction metaphor, in which hand-drawn marks, running applications, 3D visualizations, information structures, and images can be "stuck on the wall" and manipulated with a pen. This metaphor bridges the gap between the casual use of a whiteboard and the more rigid but more powerful data manipulation of a desktop (figure 2).

We have tested our wall-interaction design with several professional product design groups engaged in brainstorming tasks. Their needs and critiques motivated many aspects of our design. We chose brainstorming as our driving example because it provides a strong test for the fluidity of interaction. Successful design brainstorming depends on unhampered dynamic social interaction and the easy intermixing of different media. We believe that our solutions are general enough to be applied to many other application domains, ranging from radiology to construction planning and management, and are working with groups in those and other domains.

2. BASIC DESIGN GOALS

The design of the interactive wall was shaped by the nature of its use: users engage in up-close (arms length) direct interaction with complex visual materials in a setting of real-time multi-person collaboration. This led to three primary design goals:

- **High Resolution**. An interactive wall needs to be able to display large quantities of materials that include images and application windows at an appropriate resolution for a user standing at the board (a resolution comparable to workstation monitors). To provide a smooth wall-like environment, it needs to be a flat, continuous surface, not interrupted by physical seams. In the future, large, high-resolution flat panels will become available. For our experiments we made use of a 6' by 3.5' tiled back-projection display with 64 dpi resolution, which we call the Interactive Mural (see section 5 for implementation description).
- **Clean Screen**. There is a striking difference between the visual look of a paper-based project wall and a standard GUI screen. The GUI intersperses the user's content with a profusion of visual "widgets" for control: window boundaries, title bars, scroll bars, tool bars, icon trays, view selection buttons, and many more. A whiteboard or project wall, on the other hand, provides a uniform blank surface whose content is the user's marks or posted pieces of paper. To capture the feel of a wall we want to reserve the visual space for content, with an absolute minimum of visual distraction associated with interaction mechanics. We organize our display as an arbitrary

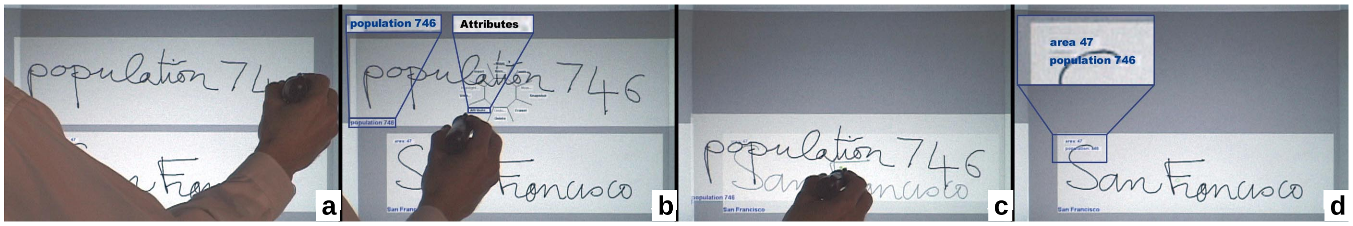

Figure 3: The sequence of actions that apply a population attribute to the sheet "San Francisco": The parameter/value pair is written on the board (**a**); FlowMenu is invoked and "item... → Attribute" chosen. Handwriting recognition is performed in the background (**b**); Continuing the pen motion, the sheet is dragged and dropped onto the target (**c**); the attribute for the target sheet is updated (**d**). (insets added for clarity)

collection of opaque and translucent "sheets," which can be created, drawn on, and moved independently. Sheets have only content on them, without visual affordances for actions.

- **Fluid Interaction**. Along with the visual clutter of GUI interfaces, there are continual interruptions to the flow of activity. Dialog boxes pop up, windows and tools are selected, object handles of various kinds appear and are grabbed, and so on. To some degree this is inevitable to provide complex functionality. The traditional workstation GUI is oriented toward facilitating the speed of highly differentiated actions done by experienced users whose attention focus is entirely on the current computer activity. But users of a wall display are often engaged in simultaneous conversation with people in the room, so their focus on the board is episodic. They do not have the additional cognitive capacity to cope with complex state differences, or have a high tolerance for having their attention distracted to the interaction with the board instead of with the people.

An interactive wall interface needs to provide more functionality than a whiteboard, but this need not call for widgets, modes, dialogs, and the other apparatus of complex interaction.

3. DESIGN PROBLEMS AND SOLUTIONS

The design goals of Section 2 have implications for all aspects of the interface. This section describes some new interaction techniques that were developed to achieve the goals in our three primary areas.

3.1 Flow and Go

Our system use FlowMenu [8] as the main command mechanism. The FlowMenu provides a clean partitioning between command and content creation: Press the meta-button on the top of the pen to invoke the pop-up FlowMenu, and in a single stroke select a command, specify a character -string parameter and perform a direct manipulation task.

Tests with users have shown that the basic FlowMenu action (select and continue moving) provides an easily learnable and fluid affordance for commands in which the parameters are determined by the sheet where the menu is initiated and by a physical path after menu choice (e.g., for moving a sheet). It also works well for built in parameter sets (e.g., a menu-based set of zoom factors) and simple numeric parameters. However, entering long text strings using the QuikWriting [29] capabilities in FlowMenu proved cumbersome, and few people have been able to use it effectively. Therefore, we have augmented it with a technique we call "Flow and Go," which combines handwritten character input with FlowMenu selection and object motion.

Since we are dealing with an information wall, not a whiteboard, it is necessary to provide for text parameters to commands. In the simplest case, we need a name to call up images, models, or connections to remote computers. More complex cases insert parameters into existing content. Our current application for brainstorming supports several operations that require such parameters. We expect this list to be expanded as new applications are developed:

- **Posting new computational objects**. To bring up a new application desktop, a JPEG image, or a 3D model, the user names the object and indicates the type of sheet to create.
- **Entering text into running applications**. When an application desktop is displayed as a sheet on the wall, the user can interact with it using the pen (which is interpreted as left-button mouse positioning) and with handwritten text replacing keyboard entry.
- **Attribute specification**. Any sheet can be assigned arbitrary attributes with their associated values using handwritten text present on the board. These are used for ordering in containers (such as scatterplots) and for recording information for later use (e.g., votes on alternatives).
- **Container parameters**. Some information structures, such as graphs, have alphanumeric parameters, such as labels and ranges on axes need to be modified during the meeting.

We do not want wall-interaction to require a physical keyboard, and virtual on-screen keyboards are disruptive and hard to use. We wanted to combine the immediacy of handwritten text entry with the FlowMenu in a way that avoided the interposition of dialog boxes and prompts, which hinder fluid interaction. Flow and Go combine FlowMenu and Drag and Drop to create a convenient way to provide text and other complex parameters.

The user starts by writing a parameter in an open area, which creates a new sheet containing the text. By default the system, applies handwriting recognition to all strokes (details in section 4.1). The recognized text forms the parameter. Then, the user activates a FlowMenu over that text sheet and selects a command. In the simple case, such as calling up an image, the text sheet is replaced with the new content and attached to the cursor for easy positioning.

If the text is a set of parameters to be inserted, such as an attribute definition, issuing the FlowMenu command attaches the text sheet to the cursor, where it can be smoothly dragged and dropped onto the target. The text sheet vanishes, and the parameters are applied to the underlying item.

Although this sequence is a bit awkward to describe, it has a very natural feel. You write the entry, choose what to do with it and, as needed, drag and drop it where it belongs. There are two separated operations (writing the text and then applying it), but they do not involve a modal state like a dialog box. The entered text is just another sheet on the screen and can be used at any time as part of a command.

A simple syntax is used for multi-parameter commands. For example, to set the attribute "population" of "746" (indicating thousands) to the sheet containing the name "San Francisco," the user writes "population 746" as shown in figure 3a). He then uses the FlowMenu on it to select the command "Attribute" (figure 3b) and drags it onto target sheet (figure 3c). The attribute tag updates accordingly (figure 3d).

3.2 ZoomScapes

It is hard for any electronic display to compete with a real wall in the quantity of information that can be posted. Until we have large, high-resolution "wallpaper" displays, we can expect computer-based displays of feasible cost to be smaller than a wall, and visibly lower resolution than what can be printed on paper. The trade-off, of course, is that a computer can dynamically reallocate visual space, scale, summarize, and do other operations that make a smaller space more effective.

In keeping with a wall metaphor, the overall display consists of a collection of opaque or translucent rectangular "sheets," which can be individually created, deleted, moved, and scaled, and which can overlap on the screen. They can contain ink marks, images, active applications, and 3D models. Each sheet can be individually scaled to an arbitrary size using numeric parameters applied via the FlowMenu.

In addition, we have created a general, location-based scaling mechanism we call ZoomScapes, which allows us to scale items by simply moving them into the appropriate region. This is similar to the automatic scaling at the edges of the display provided by Flatland [22, 23], but has been implemented as a general transformation that can be flexibly mapped to the surface of a display. We have also explored the implications of moving large objects and collections of objects across such a landscape, and have developed algorithms for creating a smooth and natural behavior.

A ZoomScape is a surface on which each point has an associated scaling value. As sheets are moved around on this surface, their size changes according to their location. Sheets are always active so unlike [15] we opted for a distortion free metaphor. To implement a ZoomScape appropriately, we must address the following problems:

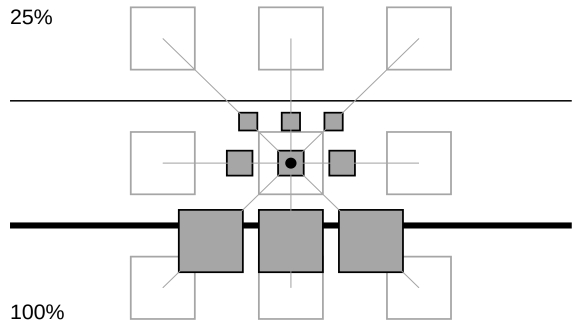

Figure 4: In addition to scalling individual sheets, ZoomScape gradually deforms the geometry of a group of sheets as they are moved from a 100% (bellow thick line) to a 25% area (above thin line) through a transition area. The light boxes show where the sheets would be in the "reference geometry" that is used to calculate the deformation (see text). The dot represent the cursor

- A sheet is not a point, but covers a rectangular area. From what point should the scale factor be taken? What happens when large items are positioned such that they overlap several scaling regions?
- In moving collections of sheets as a group, how and when do individual members of the group scale to maintain a smooth and invertible transition?
- The motion of a sheet across a scaling boundary can create an abrupt jump in scale, which is disconcerting to the user.

Through an iterative series of prototypes, we developed solutions and guidelines to these problems. The resulting implementation has a smooth, natural feel and is one of the most popular parts of our design.

The most natural scale point for moving single objects is the cursor point rather than some standard point on the image (e.g., its center). This point is directly associated with the user's hand, and therefore with the cognitive sense of where the object is being moved. Therefore, the scale of an object is not determined by its position alone, but is also a function of the action that moved it. This means that the way objects that straddle a boundary scale depends on how they were positioned. Somewhat to our surprise, users find this a feature. Large objects can be positioned in the thumbnail region at the top of the screen such that they hang into the main part of the display, effectively increasing the size of the scaling region. Conversely, a large object positioned in the main part of the display can fill the entire board, overlapping the scaling region without being reduced.

Moving a selected group of sheets into a scaling region is more complex than positioning a single sheet. For single sheets, it is sufficient to scale the entire sheet as the cursor passes into a scaling region. For groups of sheets, however, this can cause problems. Consider a group of sheets that cover a large area. If the entire group is treated as a single large object, members of the group may disappear off the edge of the board before scaling is applied, then abruptly jump back in again as the cursor passes into the scaling region. If the user stops without crossing into the scaling region, sheets can be left in the region (or off the edge of the screen) without scaling, an unexpected result.

To solve this, we separately manage the scale of individual elements and the group geometry, defined as the set of vectors between the dragging point and the center of each sheet in the group. Each sheet in the group scales about its center as it passes into the scaling region. The distance between the sheet and the cursor is also scaled, smoothly distorting the group geometry as it passes through the scaling region (figure 4).

The implementation for the group distortion is as follows: For each vector $\overrightarrow{AO}$ from the initial drag anchor point A to a sheet center O, we compute its direction $\vec{u}$ and L_{ref} its "reference length", which is the integral over $\overrightarrow{AO}$ of the shrinking factor *shrink*(*x*,*y*) (the inverse of the scale factor at that point:

$$L_{ref} = \int_{\overrightarrow{AO}} shrink(t)dt$$

L_{ref} can be thought of as the length that would result if the entire vector were in a unit-scale region. As the anchor point is moved, the location of the center of each sheet is adjusted to keep the direction $\vec{u}$ of the vector and the reference length L_{ref} constant. At each move, we calculate the distance that will preserve the value of the integral, using Newton's approximation method. The scale factor for the sheet itself is determined by the resulting location of its center point. This approach is very general and can accommodate various ZoomScape geometries, including those with non-rectangular regions and non-linear ramp functions.

ZoomScape configurations that use continuous scaling changes (e.g., small at the edges, large in the center) or provide ramps between areas of different scaling create smooth scaling motion. We found that in general it is better to have several uniform regions rather than a continuous change, so that small motions within a region do not cause unintended scale change. For our experiments on brainstorming, the screen is divided into two uniform areas with a small ramp area between them. Sheets moved into the top 1/5 are automatically scaled to 1/4 of their size while objects on the lower part of the screen are full size. Users seem to find it natural to push less-used data to the top of the board, where it remains visible throughout the room.

Although the interaction of ZoomScapes with containers and groups required experimentation and is complex to describe, the experience in using it is that things simply move "where they should". We have learned that as long as automatic scaling activities are controlled in a smooth monotonic way by the user's motions and are continually visible, the underlying algorithms are not directly perceived by the user, and the changes feel natural. In general, the experience of using the ZoomScape involves no explicit planning or action formulation is to manage items on the screen. They are simply moved and the scaling follows naturally.

4. OVERALL DESIGN

In order to provide users with the experience of interacting with a wall of information, we needed a coherent overall design that encompassed the creation, incorporation and manipulation of visual materials. In addition to the new techniques described in Section 3, we have addressed many other aspects of interaction, each with the goals of keeping the screen clean and the interaction fluid.

4.1 Stroke interpretation

A user can draw a stroke with a pen for several different purposes (We define "stroke" as putting the tip of the pen on the board, moving it through any path, and then lifting it). Using our system, the user can invoke the FlowMenu anywhere on the surface, by pressing the meta-button located on the top of the pen. The effect of the resulting command stroke will depend of the menu selected. The effect of button-up strokes depends on the front most sheet at the point where the stroke begins. In an active application window, a stroke is like a left-button mouse click or drag. On a 3D object, it rotates the object. On any 2D graphical object (bitmap graphics as well as hand-drawn) the stroke creates a mark of digital ink. As in FlatLand, our system automatically merges 2D graphical objects linked by a stroke. This feature make it easy to create the equivalent of a mixed media collage. If a stroke begins on the wall background, the result depends on its overall path. If its bounding box intersects an existing 2D graphical object sheet, it is interpreted as a mark on that sheet, and the sheet is extended to cover it. Otherwise a new drawing sheet is created and the stroke creates a mark on it. As in Flatland, the bounding box for a collection of marks extends slightly to the right and downward from the geometrical bounding rectangle, so that subsequent strokes intended as handwriting will be merged with the previous ones.

The net effect of context-dependent stroke interpretation is that the most common operations on all the types of materials can be done by using the pen directly without issuing commands or using special affordance areas.

Mark strokes are sent to the Calligrapher character recognizer [27] each time the user pauses. The ink is not modified or removed and recognition does not interfere with further interaction. The recognized text appears as it is computed in the lower left corner of the sheet. A dictionary is used, and in practice the recognition has been surprisingly robust. In order to avoid meaningless attempts to interpret drawings as character data, simple heuristics are applied to turn off recognition when the stroke geometry is not text-like. If desired, the user can use the FlowMenu to specifically initiate a drawing sheet that it is not to be recognized as text, but since the character recognition is non-intrusive, users generally do not bother to do this.

4.2 Image origination

Whiteboard-like content (hand-drawn sketches and handwriting) is generated with strokes as described in the previous section. Non-stroke content can be added to the wall through a number of channels:

- The wall display is one element in a room that incorporates multiple devices, including other large screens and a wireless network of laptops and PDAs. Any of these devices can run a small applet onto which the URL for an image file can be drag-and-dropped, causing a sheet with the image to appear on the wall.
- A digital camera (Ricoh RDC-7) mounted on the ceiling above a table can be used to create an image of any paper document or small objects laid on the table. The field of view is 21”x15” with a resolution of approximately 100 dpi. Two physical crop marks are positioned on the corners of the area of interest within the view field. The scanner is triggered through a FlowMenu command at the wall, or through other devices in the room. When a scan is triggered, a place-holder sheet for the image appears immediately on the wall, and can be annotated, moved, and otherwise manipulated during the time it takes for the image to be captured and transmitted (which can be ten seconds or more with the current equipment). Although in many ways this is equivalent to a flatbed scanner, the direct imaging from the tabletop has proved to be particularly appealing to designers.
- Using a FlowMenu item and naming any computer on the network that is running a VNC server [32], the wall can display a live image of its desktop. The pen operates as a pointing device (with left-button down) when it is touched within the sheet, and text input is entered into the remote desktop using Flow and Go.
- FlowMenu commands for new 2D and 3D objects accept a text parameter and render the corresponding object in a new sheet. The parameter can be a short name, which is interpreted as an entry in a local database.
- Finally, the FlowMenu provides a snapshot command that enables the user to select (by corner dragging) a rectangular region of 2D images, stroke sheets, and VNC sheets, and create a snapshot of it. This makes it easy to use VNC to browse or generate information of interest, and then stick results on the wall.

4.3 Containers

In a stroke-oriented system for whiteboard-like use, structures can be inferred from stroke geometry [18, 19, 12, 22, 23]. For a wall-interaction metaphor with mixed materials, we use automatically interpreted structure at the lower level (e.g., inferring that a collection of strokes is a character string), but use explicit structuring with “containers” for higher level structures such as lists and graphs. A container can be created via a FlowMenu command over a sheet or a collection of selected sheets. The sheets are automatically moved and aligned in accordance with the container type. The simplest container is a list (see upper right of Figure 2), in which items are left-aligned in a vertical column. Items can be added to the list by dragging them into its area, where they will be inserted at the point closest to the pen location when they are dropped. Items can be moved out of the list or to a different position in the list with the standard move command since menu invocation over any element acts on that element as it would irrespective of the container. Each container type has a distinct visual look that includes a “background” area in which menu commands can be applied to the container as a whole.

When a container such as a list is moved to a different scale area in the ZoomScape, all of the items are correspondingly scaled together, even though they may individually be in different ZoomScape regions. For example placing the top of the vertical list in the scaled down area at the top of the screen will scale the list one fourth and let the list flow downward in effect extending the scaled down area. Since the list is still active, the user can drop or remove items from it. At first this seemed like an unintended effect, but in practice it has turned out to be both comprehensible and useful, extending the benefit of the ZoomScape to produce compacted material that can cover more of the board area.

In addition to lists, we have implemented a “scatterplot” container: a 2-dimensional graph onto which sheets are positioned according to the value of numerical attributes. The plot is created with a FlowMenu command, and its axis labels and ranges are added using Flow and Go onto the corresponding areas of the graph geometry. Attribute values can be assigned to any sheet with Flow and Go by writing an attribute-value pair and then using the Set Attribute menu command to drag this value onto the sheet. The resulting attribute-value pair is displayed in the upper left corner of the sheet contents (figure 3). When a sheet with attribute names corresponding to the axes is moved onto a scatterplot, it is incorporated and positioned according to the values. If the sheet is missing an axis attribute, a random value within the range is assigned. As with lists, once a sheet has been added to a scatterplot, it moves and scales with that container, but can still be accessed (including marking on it or moving it out) with the full set of operations available for any sheet of its type.

4.4 Storage and logging

One of the key advantages of an electronic wall over a conventional physical wall is that material can be saved, restored, viewed elsewhere, and searched electronically. We provide two different forms of information capture:

- **Logging**. Each action on the wall is entered in a time-stamped sequential log, along with contextual information that makes it possible to replay the sequence in forward or reverse. This log is the basis for an infinite undo facility. Using the FlowMenu as a knob, the user can undo actions one by one, and redo them back in the same order, to previous state of the wall. The log can be used later to restore the session to its state, except for dynamic states of remote desktop connections. The restored state includes the full history, so it is possible to undo actions from the restored session and return to intermediate state.
- **Visual snapshots**. A snapshot of the entire screen can be saved as a Microsoft Word file, using the MSOffice drawing format, with text art fields for the interpreted character strings. This file can be printed or viewed on a workstation, and standard tools (e.g., cut and paste) can

be used to repurpose, rearrange, add, and modify items. Currently this snapshot does not contain images from 3D models or VNC screens.

5. IMPLEMENTATION

To achieve arms-length viewing resolution (64 dpi) while providing a large seamless space (6' by 3.5'), we constructed a tiled back-projection display using twelve 1028x768 digital projectors with precision-aligned non-overlapping tiles. Although some visual artifacts result from the tiling (see Figure 2), the experience is that of a unified graphic space. We anticipate that within a few years, displays of at least this size and resolution will be available commercially. Input is provided by a wireless Ebeam digital pen [4], modified to include a push button switch. This input device is relatively inexpensive, and could be replaced by any pen-tracking system with sufficient precision and area coverage.

The 9 megapixels graphics canvas is rendered as a single OpenGL surface, using the WireGL protocol [9] and a distributed rendering API [10] that can take advantage of a cluster of 32 Linux PCs connected with a 1GB/sec local network to render complex 3D models. This configuration was available as part of research on high-resolution high-throughput graphics in our laboratory. We do not expect configurations of this complexity to be generally available, and our basic interaction principles were not designed to require them. In fact our system was demonstrated running on a single PC driving a SmartBoard [35].

The software is written in C++ on top of a toolkit we developed called MilleFeuille, which handles the basic graphics operations, pen tracking, stroke marking, and the composition of behaviors on overlaid layers. We use the EventHeap [5] to provide a uniform blackboard-like communication structure for other devices, such as those used for scanning or transferring images to the wall. An off-the-shelf handwriting recognizer [27] is used for text recognition, VNC [32] is used to display GUI desktops from servers, and QSplat [34], a resource-dependent renderer is used to display 3D models.

6. RELATED WORK

This research builds on a number of areas of previous work on wall and whiteboard display systems.

6.1 Wall-interaction metaphor

Large wall-based information spaces are common in non-computer settings. Mixed mode approaches integrate paper-based materials with an information system, as in the Insight Lab [14], which uses barcodes to associate paper items with on-line information, Collaborage [20], which uses camera capture of paper-based materials posted on a wall and the The Designers' Outpost [11] which track Post-It notes and their relationships on the screen. Although there is great potential for bits-plus-atoms mixtures in future environments, our experiments have concentrated on new kinds of interaction with fully electronic representations of information. There have been a number of large digital walls developed for display without close-up interaction [6] and interactive digital walls have been used for tasks such as tape drawing [2], with specialized interaction techniques

In interaction style, our work is most similar to whiteboard interfaces, which have emphasized informal interaction, clean screens, and fluid interactions with stroke-based materials. Tivoli [28] was the first well-developed electronic whiteboard system, and we have adopted its basic stroke capture metaphor and concepts for containers (lists, etc.) as loosely structured geometrical composites. Tivoli used GUI-like controls outside the whiteboard area, multiple screens for visibility management, and command gestures recognized by shape. We have chosen not to duplicate those aspects, and have added others related to materials on the wall that are not based on hand-drawn marks.

Flatland [22, 23] extended the whiteboard to provide automatic grouping of strokes into "segments," with affordances for moving, merging and performing other manipulations on segments as a unit and for giving them domain-specific and application-specific behaviors. We have adopted Flatland's approach to segmentation for stroke data, extending it to an environment that includes other kinds of materials along with handwritten input.

The key difference from previous whiteboard systems is our integration of multiple kinds of media onto the wall, which is structured as a collection of independent overlapping rectangular translucent or opaque sheets rather than a single surface with marks. This integrates the common GUI windowing metaphor, with its overlapping rectangles, and a whiteboard metaphor, which provides a surface for strokes. It captures the feel of a wall, on which overlapping information can be posted and moved around and hand-based drawing can easily be added. Unlike GUI windows, the sheets do not carry elaborate action-affordance borders, since the FlowMenu is used to manage them. We have not tried to provide non-rectangular sheets or complex sheet management operations, as Kramer did for his translucent patches [12], but have designed for simplicity of drawing and annotating.

6.2 Visibility management

The predominant GUI metaphor achieves high visual capacity through the illusion of overlaid rectangular areas, which can be easily moved and can be opened and closed, but in which the material has a fixed size. Most digital whiteboard interfaces stay close to the physical whiteboard metaphor by having a fixed-size screen with no visibility management, along with a mechanism for multiple screens, selected using a thumbnail display. Flatland has independent segments, but does not let them overlap or open and close. It manages space with a scaling technique based on filling up the board. As a segment is moved, other segments move out of the way. Segments are automatically squashed as they bump into the border of the board. Since we allow overlapping sheets, we have not used Flatland's space management techniques for moving segments, and have generalized the automatic scaling mechanism to ZoomScapes.

We retain the whiteboard's overall spatial metaphor of a fixed-size single surface (in our case a wall rather than a whiteboard) with objects that can be individually scaled, rather than presenting an unbounded virtual surface with zoom-based navigation, as in Pad++ [1]. Global zooming can hinder the usability of the interface with a group, since actions can disrupt viewing in areas away from the object of interest, especially if semantic zoom techniques cause objects change to their visible representation when a zoom-scale change is made.

Like the Perspective Wall [15], ZoomScapes leverage the fact that users maintain contextual information at the periphery of their work area or focus area. While most Focus+Context techniques put an emphasis on controlling the distortion, ZoomScape imposed a fixed distortion and put an emphasis on an easy and safe manipulation of underlying pieces of information.

6.3 Command interaction

Interaction designs for small pen-based devices have explored a number of ways to replace conventional keyboards and pointing devices. In addition to conventional GUIs, they have incorporated radial menus [17], marking menus [13], handwriting and gesture recognition [3] and character entry pads [7, 26, 29, 16]. Pen interaction for large displays has primarily been developed for whiteboard systems, and has focused on the use of gestures as part of stroke interpretation.

Three aspects of command interaction need to be provided: command activation, item manipulation, and the entry of text and parameters. These are in addition to direct content interactions such as drawing, rotating a model, interacting with an application, etc.

Command activation: Whiteboard systems such as Tivoli [28] use gestural marks for commands, along with GUI widgets on some areas of the board. Flatland provides radial menus for command activation, augmented by the use of gestural marks. Others allow the choice of commands to be determined by the physical way in which the stroke is made, such as choice of devices [35] and the posture of the gesturing hand [33]. We have chosen to provide a single uniform mechanism. All commands are activated with the FlowMenu [8]. The absence of mnemonic gestures (e.g., X or zigzag for erasing, caret for inserting) is made up for by the uniformity of a simple and straightforward conceptual model, which helps initial learning. As more practice is gained, more and more complex menu-selections are remembered as gesture.

Item manipulation: In general, pens work well for selection operations (individual and group), and for item motion. Standard GUI techniques need to be adapted due to the lack of modifier buttons and keys, and each system has solutions to this problem, combining in some way the use of gestures, widgets (such as handles) menus, and temporal modes. Our solution is uniform, in that every operation is initiated with a FlowMenu selection and applied only to the ongoing stroke. A single selection, for example, is done by choosing a “select” menu item and lifting the pen. A group selection begins with a “select group” menu choice followed (in a continuous motion) by a lasso path surrounding the items. Items are moved by choosing a “move” item, at which point the object under the menu becomes attached to the pen and follows until the pen is lifted. This uniformity allows us to avoid visual affordance handles (such as title bars or Kramer's “pearls” [12]), to avoid the use of temporal modes, and to have a simple conceptual model for what strokes do (all non-meta-strokes interact directly with the contents, and all meta-strokes initiate a FlowMenu).

Text and Parameter entry: Small-screen pen-based systems have devoted a good deal of effort to text entry. Techniques include virtual keyboards, handwriting recognition, special character recognition [7, 26], and continuous radial selection [29, 16]. Commercial whiteboard systems provide virtual keyboards and some form of handwriting recognition. In order to introduce the mix of media for wall-interaction, we needed to provide for pen-based character input. We initially provided only the character selection mechanism of FlowMenu, based on QuikWriting [29]. User testing showed that for short sequences (e.g, a few digits to specify a scaling value) this was acceptable, but for anything longer, the mechanism was cumbersome even for experienced users. In addition, we wanted to provide character recognition on stroke-created content, in order to process the wall contents for subsequent use, such as search. We incorporated a commercial handwriting recognition engine [27], and developed the Flow and Go technique, as described in section 3.1.

6.4 Integration with other devices

The Interactive Mural is one part of a larger environment that combines multiple devices into an integrated workspace, which includes several other large displays, special devices such as the overhead scanner, and integrated software connecting laptops and PDAs [5]. Our goals in this research focused on interaction with a single large display, but it was done with an eye towards how that display participates in the larger environment of an interactive workspace. For example, a person using a laptop with a wireless LAN connection can use a browser-based applet to drag-and-drop an image onto the wall. This is one step towards the kind of multi-device integration that has been explored in a number of projects with techniques such as tangible-based Pick and Drop [30, 37], PDA-based tool palettes [31, 21], and multi-modal pen and voice interaction [25].

7. DISCUSSION

7.1 User response

We have conducted a pilot user study with five groups of designers from IDEO and Speck Design. Each team was invited to perform a brainstorming session on a topic of their choice (actual working sessions) and then were asked to fill out questionnaires on specific features and general user satisfaction. The system was also used during the early design of the Chrysler Design Awards 2001.Designers have reacted very positively to the wall-interaction metaphor. The capability to easily scan paper documents and the use of ZoomScape for visibility management were rated as the most valuable features. Facilitators were eager to make use of the digital access after sessions, avoiding the tedious process of recording what was on a wall before taking it down. We do not yet have any longitudinal data on how the materials are used after capture. The ZoomScape feature was well received and initial user testing showed that it was used without hesitation by almost all of our pilot groups. We have yet to explore "landscapes" other than our simple top/ bottom division and to see how they can be used for different purposes.

The greatest perceived problems resulted from difficulties with the pen input device and performance issues such as the long delay in getting an image from the camera. There is also a general performance decay when the board begins to have more than a hundred items. We were careful to design all operations to give immediate feedback and to allow multi-tasking, so delays do not block the user from going on. Overall performance issues are different for our OpenGL model (which actively redisplays the contents of every sheet at screen-refresh rate) than they are for traditional pixel buffers and backing stores. The implementation needs to take this into account as it is developed for larger walls and longer sessions.

We found that we had not provided users with sufficient training time. Our goal was to build an interface that was easily learnable but did not need to be self-revealing. We planned an hour of instruction and practice for first-time users (the facilitators of the brainstorming sessions). The goal was to have all aspects of the board be memorable (or self-revealing) after a short exposure. To a large degree this succeeded, but some aspects, such as the use of containers, were not remembered well enough to be used effectively in a single session. In our next round of studies we will look at mastery of the system over multiple sessions.

7.2 Interaction goals

We set out to meet three basic goals in our interaction: Integration of high resolution materials, clean screen, and fluid interaction. All of these received high marks from our users, and we feel that we have gone a long way towards meeting them.

High resolution was used in two key ways. First, it enabled the incorporation of materials such as Web browsers (and snapshots taken from them) with a visual quality closer to that of physical pictures than to the typical large-pixel projection screen. Second, the sessions all made heavy use of the ability to scale hand-drawn text and sketches to 1/4 size. They retained full readability, allowing people to write and draw at a large, physically comfortable size and then easily reduce materials to an effective viewing size that gave a large virtual screen space.

The "clean" look we achieved was clearly appreciated by our visually sophisticated user group. We were able to almost completely avoid the use of "decorations" associated with control. The one small exception was the background areas of list containers: a vertical bar along the left side that is in the list but not in any item. This enabled the user to apply the full range of operations to the list as a whole, or to any item within it. In a way, this is like having a background (as the overall screen does), which has its own set of potential operations distinct from those of the items within it. As we develop more kinds of containers we will explore this question further. We may also need to modify the look to accommodate operations such as the "scrolling" of materials within a sheet that provides a virtual portal onto a larger visual field (as conventional windows do). We want to do this in a way that does not violate the overall wall metaphor or add visual clutter.

Fluid interaction is an elusive quality to measure, but our observations and discussions show that we have made significant progress in that direction. There are no temporal modes that require users to keep in mind what they are doing beyond a single stroke. No dialog boxes are needed to accomplish standard operations such as specifying an attribute and its value. From our user studies, we have identified a number of details that can be improved, but overall the facilitators were able to use the system while maintaining focus on the group and process. We can attribute this both to the basic interaction techniques, such as FlowMenu and ZoomScapes, and to a variety of design details that were guided by the quest for fluidity.

The use of handwritten character recognition for Flow and Go commands has been effective even with the limitations of character recognition accuracy. Although all text strokes are given to the recognizer, accuracy is not critical except for command parameters, which tend to be relatively short. We currently use a standard English dictionary to guide the recognizer, and are exploring ways to use specific context (such as the list of available models, a file directory, or a personal dictionary) to enhance its recognition for command strings. In our current implementation, recognized text cannot be edited. Although the rewriting of short strings is generally as easy as modifying them with edit commands, there is clearly a need to add an editing facility.

8. CONCLUSIONS

The wall-interaction metaphor brings together the advantages of the graphical user interface with those of whiteboards. It allows users to work with computer based materials, including arbitrary applications and 3D models, with the ease of informal writing, sketching, and space management. In implementing this metaphor we have:

- demonstrated the feasibility and appeal of interacting directly with digital materials on large surfaces.
- developed and demonstrated the use of design criteria that are suited to the use of these surfaces in group activities such as brainstorming
- introduced new interaction techniques that follow these criteria and provide in a widely applicable way for a number of interaction aspects including visibility management (ZoomScape), and command invocation with parameters (Flow and Go). These techniques allow users to achieve the effects of complex commands with simple fluid interactions.

While this techniques were demonstrated here in the context of the brainstorming tool on a large, high performance display wall, we believe that they address generic needs. With little or no modification they can be applied to a wide variety of application domains and interactive surfaces like tablet computers and PDA.

9. ACKNOWLEDGEMENTS

The authors wish to thanks Ian Buck and Matthew Eldridge from the WireGL group at Stanford for their ongoing technical support, Phil Weaver and Yao Ding from EFI for their gracious support in adding a button to our pen. David Kelley and many designers from IDEO as well as David Law from Speck Product Design provided invaluable support and feedback. Thanks to Brian Lee for porting the system to a single PC configuration. This work was sponsored by the DOE VIEWS program (contract B504665).

10. REFERENCES

1. Bederson, B.B., and Hollan, J.D. Pad++: A Zooming Graphical Interface for Exploring Alternate Interface Physics. *In Proc. UIST '94*, pp. 17-36.
2. Buxton, W., Fitzmaurice, G., Balakrishnan, R., and Kurtenbach, G. Large Displays in Automotive Design. *IEEE Computer Graphics and Applications*, 20(4), pp. 68-75.
3. Carr, R., and Shafer, D. The Power of Penpoint (1991).
4. Electronics for Imaging, http://www.e-beam.com/
5. Fox, A., Johanson, B., Hanrahan, P., and Winograd, T. Integrating Information Appliances into an Interactive Workspace. *IEEE Computer Graphics & Applications*, 20(3) (May/June 2000).
6. Funkhouser, T., and Li K., (eds.). Special issue of *IEEE Computer Graphics and Applications.* Onto the Wall: Large Displays, 20(4) (Jul/Aug 2000).
7. Goldberg, D., and Richardson, C. Touch-Typing with a Stylus. *In Proc. CHI'93*, pp. 80-87
8. Guimbretière, F., and Winograd, T. FlowMenu: Combining Command, Text, and Parameter Entry. *In Proc. UIST '00, pp. 213-216*
9. Humphreys, G., Eldridge, M., Buck, I., Stoll, G., Everett, M., and Hanrahan, P. WireGL: A Scalable Graphics System for Clusters. *In Proc. SIGGRAPH '01*.
10. Igehy, H., Stoll, G., and Hanrahan P. The Design of a Parallel Graphics Interface. *In Proc. SIGGRAPH '98*, pp. 141-150.
11. Klemmer, S.R., Newman, M.W., Farrell, R., Bilezikjian, M., and Landay J. The Designers' Outpost: A Tangible Interface for Collaborative Web Site Design. *In Proc. UIST* '01.
12. Kramer, A. Translucent Patches: Dissolving Windows. *In Proc. UIST '94*, pp. 121-130.
13. Kurtenbach, G., and Buxton, W. Integrating Mark-Up and Direct Manipulation Techniques. *In Proc. UIST ' 91*, pp. 137-144.
14. Lange, B.M., Jones, M.A. and Meyers, J.L. Insight Lab: An Immersive Team Environment Linking Paper, Displays, and Data. *In Proc. CHI '98*, pp. 550-557.
15. Mackinlay, J.D., Robertson, G.G., and Card, S.K.The Perspective Wall: Detail and Context Smoothly Integrated. *In Proceedings of CHI '91*, pp. 173-179
16. Mankoff, J., and Abowd, G.D. Cirrin: A Word-Level Unistroke Keyboard for Pen Input. *In Proc. UIST '98*, pp. 197-206
17. Momenta User's Reference Manual. Momenta.
18. Moran, T., Van Melle, B., and Chiu, P. Spatial Interpretation of Domain Objects Integrated into a Freefrom Electronic Whiteboard. *In Proc. UIST '98*, pp.175-184.
19. Moran, T., Van Melle, B., and Chiu, P. Tailorable Domain Objects as Meeting Tools for an Electronic Whiteboard. *In Proc. CSCW '98*, pp. 295-304.
20. Moran, T., Saund, E., Van Melle, W., Gujar, A.U., Fishkin K.P., and Harrison, B.L. Design and Technology for Collaborage: Collaborative Collages of information on Physical Walls. *In Proc. UIST '99*, pp. 197-206
21. Myers, B.A. . Using Multiple Devices Simultaneously for Display and Control. Special issue of *IEEE Personal Communications*, Smart Spaces and Environments, 7(5), (Oct. 2000), pp. 62-65.
22. Mynatt, E., Igarashi, T., Edwards, W.K. and LaMarca, A. Flatland: New Dimensions in Office Whiteboards. *In Proc. CHI '99*, pp. 346-353
23. Mynatt, E., Igarashi, T., Edwards, W.K. and LaMarca. Designing an Augmented Writing Surface. *Computer Graphics and Applications*, 20(4) (Jul/Aug 2000) 55-61.
24. Nunamaker, J.F., Dennis, A.R., Valacich, J.S., Vogel, D.R., George, J.F. Electronic Meeting Systems To Support Group Work. *CACM*, 34(7), (Jul 1991), pp. 40-61
25. Oviatt, S.L., Cohen, P.R., Wu, L.,Vergo, J., Duncan, L., Suhm, B., Bers, J., Holzman, T., Winograd, T., Landay, J., Larson, J., and Ferro, D. Designing the User Interface for Multimodal Speech and Gesture Applications: State-of-the-Art Systems and Research Directions. *Human Computer Interaction*, 15(4), pp. 263-322
26. Palm OS. http://www.palmos.com/
27. Paragraph Corp. Calligrapher. http://www.paragraph.com/
28. Pedersen, E.R., McCall, K., Moran, T., and Halasz, F. G. Tivoli: An Electronic Whiteboard for Informal Workgroup Meetings. *In Proc. InterCHI '93*, pp. 391-398.
29. Perlin, K. QuikWriting: Continuous Stylus-Based Text Entry. *In Proc. UIST '98, pp.215-216*.
30. Rekimoto, J. Pick-and-Drop: A Direct Manipulation Technique for Multiple Computer Environments. *In Proc. UIST* '97, pp. 31-39.
31. Rekimoto, J. A Multiple-Device Approach for Supporting Whiteboard-Based Interactions. *In Proc. CHI '98*, pp. 344-351
32. Richardson, T., Stafford-Fraser, Q., Wood K.R., and Hopper, A. Virtual Network Computing. *IEEE Internet Computing*, 2(1), Jan/Feb 1998, pp. 33-38.
33. Ringel, M., Berg, H., Jin, Y., and Winograd, T. Barehands: Implement-Free Interaction with a Wall-Mounted Display. *In Proc. CHI '01 extended abstracts*, pp. 367-368
34. Rusinkiewicz, S., and Levoy, M. QSplat: A Multiresolution Point Rendering System for Large Meshes. *In Proc. SIGGRAPH '00*, pp. 343-352
35. Smart Technologies. http://www.smarttech.com.
36. Stefik, M.J., Foster, G., Bobrow, D.G., Kahn, K., Lanning, S., and Suchman, L. Beyond the Chalkboard: Computer Support for Collaboration and Problem Solving in Meetings. *CACM*, 30(1), (Jan 1987), pp. 32-47.
37. Streitz, N.A., Geißler, J., Holmer, T., Konomi, S., Müller-Tomfelde, C., Reischl, W., Rexroth, P., Seitz P., and Steinmetz R. i-LAND: an Interactive Landscape for Creativity and Innovation. *In Proceedings of CHI '99*, pp. 120-127.