Pen-Based Interaction Techniques For Organizing Material on an Electronic Whiteboard

Thomas P. Moran, Patrick Chiu,* William van Melle

Xerox Palo Alto Research Center, 3333 Coyote Hill Road. Palo Alto, CA 94304

{moran,vanmelle}@parc.xerox.com, chiu@pal.xerox.com

ABSTRACT

This paper presents a scheme for extending an informal, pen-based whiteboard system (the Tivoli application on the Xerox LiveBoard) to provide interaction techniques that enable groups of users in informal meetings to easily organize and rearrange material and to manage the space on the board. The techniques are based on the direct manipulation of boundaries and the implicit recognition of regions. The techniques include operations for shrinking and rearranging, structured borders that tessellate the board, freeform enclosures that can be split, fused, and linked, and collapsible annotations. Experience with using these techniques, the results of a user test, some design trade-offs and lessons, and future directions are discussed.

KEYWORDS: whiteboard metaphor, pen-based systems, freeform interaction, implicit structure, emergent structure, structural grouping, informal systems, recognition-based systems, list structures, meeting support tools, gestural interfaces, user interface design

1. INTRODUCTION

Our goal is to provide computational support for small, real-time "working meetings" in which groups interact to generate and organize ideas (creating and assessing new ideas and perspectives, discussing them, organizing them, negotiating about them, and so on). Group interaction in such situations is informal, freewheeling, and fluid.

Groups often use whiteboards to provide a shared visual surface to record the ideas they are working with and to preserve a shared context. When manipulation demands are high, people sometimes use a tackboard, magnetic board or tape and paper in order to be able to move items around. Sometimes people use projected computer applications, such as word processors or spreadsheets, to gain the advantages of both editing and saving the created material. However, these kinds of applications are not very well suited to meeting situations, because they force users to create and work with formalized representations. The overhead of using such representations inhibits the very processes they are meant to support [13]. One of the big challenges for current HCI design is to create systems to support *informal interaction*.

Pen-based systems that allow scribbling on wall-size displays or notepads can support whiteboard or shared notebook metaphors for working with informally scribbled material [4]. The free, easy, and familiar expression permitted by such systems makes them a promising class of tools to support informal interaction in meetings [15]. We are exploring pen-based techniques usable by people without any special drawing skills, i.e., most people who go to meetings.

We have been working for several years on a program of research to provide computational meeting tools based on a whiteboard metaphor. Our idea is to allow freehand/freeform creation and manipulation of materials on the whiteboard and to provide facilities to easily and noncommittally organize and structure the materials as needed by the group.

What makes this program of research possible is the *LiveBoard* [3], a large, shared, pen-based, rear-projected, electronic display. This provides a whiteboard-size interactive display that allows us to experience and experiment with "group-computer interaction" in a way not possible with smaller displays, such as workstations, or non- or semi-interactive, front-projected displays.

We have developed a software application, called Tivoli [11], that simulates whiteboard functionality on the LiveBoard.[1] Tivoli provides basic pen-based scribbling and erasing interaction, plus facilities for editing materials by pen-based gesturing and wiping techniques.

We extended Tivoli to allow "implicit structuring" [9] of the material on the board.[2] Material is created on the board in a

1. Tivoli is a research prototype used at PARC [8]. It is written in C++ and runs under Unix and X Windows. The LiveBoard is a PC-based commercial product from Liveworks, Inc. Tivoli is run on the LiveBoard through an X server on the LiveBoard. (A PC-based commercial product from Liveworks, called MeetingBoard, is an extended early version of Tivoli.)

2. We use the term "board" in this paper to refer to the visible work surface provided by Tivoli on the LiveBoard display.

Appears in the Proceedings of the UIST'97 Symposium on User Interface Software and Technology.

* Current address: Patrick Chiu, Fuji Xerox, Palo Alto Laboratory, 3400 Hillview Avenue, Palo Alto, CA 94304. At the time of the work reported in this paper, Patrick Chiu was with Liveworks, Inc., A Xerox Company, working at Xerox PARC.

freeform manner, which means that anything can go anywhere without constraint. However, the user can indicate to the system by certain gestures that the material is to be regarded temporarily as having a certain structure, such as handwritten text. Tivoli will then apply editing operations in accordance with the conventions of that structure, such as opening and closing space where needed when moving words around (similar capabilities are provided by [1]).

We have now further extended Tivoli by developing a set of simple and natural techniques for helping users spatially organize material on the board. This paper presents, in Section 2, the motivation, approaches, and design principles for these new techniques. Section 3 describes the techniques themselves. Finally, in Sections 4 and 5, we discuss our experience with using the techniques, a user test we conducted, some lessons, and future directions.

2. PRINCIPLES, GOALS, APPROACHES

2.1. General Design Principles

Tivoli is designed to be used in a group setting. The board is large enough for a small group to see, so it can function as a shared work surface (see Figure 1). To serve this setting, Tivoli's user interface design is guided by several principles. Some important ones are:

Social Perceptibility Principle. The interactions at the board must be understood by the whole group, not just the person at the board. Several techniques support this goal. The activity on the board, such as drawing and erasing, is *localized* to be near the pen, where people are focused. *Gestures*, which are interpreted pen strokes, are used in preference to remote buttons. And complex changes to material are *animated* so they can be understood [9].

Openness Principle. An important feature of informal group interaction is that it is open to unanticipated directions; we don't want to inhibit this. Tivoli always allows freeform scribbling, so that anything can be expressed. A subtle variation on this is the decision to have a separate mode for gesturing. Most pen-based systems interpret some ink strokes as gestures, but in Tivoli we did not want to foreclose the use of any kinds of ink strokes by predefining them to be gestures.

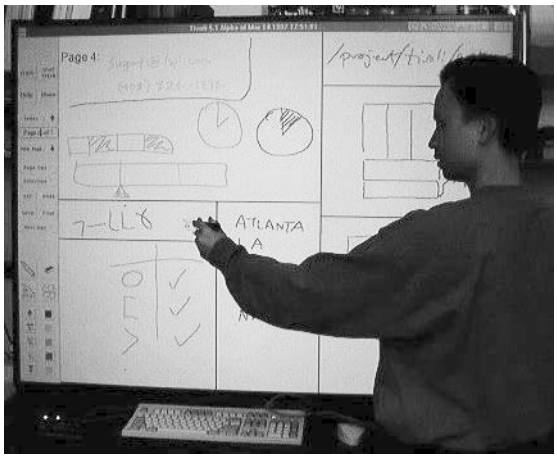

Figure 1. The LiveBoard and Tivoli is meant to be seen by a group of people.

Emergence Principle. Ideas do not arise well formed. At first there are expressions of fragments of thoughts. Once there is some rough material to work with, interpretations gradually begin to emerge as they are discussed. Users need to be able to change their minds and explore different interpretations and arrangements. The implicit structure technique in [9] and the organizational techniques in this paper support this style of informal group interaction.

Agility Principle. Group interaction needs to be fluid and rapid. Thus the user at the board needs to be nimble to keep up with the group, so the board can be an integral part of the working out of ideas and not just a recording medium. A key feature supporting agility is Tivoli's Undo feature, which enables users to "perform"[3] at a rapid pace without fear, knowing they can back up if they have to. The tool should be fun to use, encouraging it to be "played," and thus the materials created to be "played with."

2.2. Goals for Organizing Material

We have set two basic goals for the design of organizational tools: (1) to use the board space efficiently and (2) to group materials on the board.

Using Space Efficiently. The board is never big enough. Tivoli has multiple, arbitrarily-large, scrollable pages, providing an unlimited capacity for material. More important is how much material is *visible at the same time* on the board.

Grouping Materials. Scribbles and other kinds of objects on the board are grouped to make categories visible and to allow subsets of material to be treated in similar ways. Structured operations, such as rearranging a list, can then be confined to a grouped subset of material. Thus the groupings should not only be visible to the users, but also must be recognizable by the system.

2.3. Other Approaches to Organizing Material

There are several approaches to providing organizational capabilities, based on different metaphors:

Frozen Groups Approach. Many drawing applications allow selected objects to be "frozen" into a group, so they can be operated on as a unit. This is handy, but there are several problems: the groups are not visually apparent, the elements in a group cannot be manipulated, and no structural interpretation can assigned to a group.

Overlapping Regions Approach. Material can be grouped by enclosing items in opaque regions, which can be overlapped to save space, thus hiding some material. A region can be made smaller, showing only part of the material, which can be made visible by scrolling. The most common manifestation of this approach is in window systems, which are too rigid to use to organize material. The VIKI application [7] uses this approach to managing information on a 2D surface. Its basic objects are "notecards," which can be put into "collections," which are manipulated like nested windows. VIKI

3. A concept of Fred Lakin's (see http://www.pgc.com).

is fairly agile, but it is purely text-based, and its complex overlapping could be confusing in a meeting setting.

Translucent Overlays Approach. In this approach, proposed by Kramer [5], sketched material is grouped into separate translucent layers, like layers of "onionskin" paper used by architects and designers. The material on successive layers can be seen gradually fading as the layers get deeper. A layer can be assigned an interpretation, and operations can be restricted to the material on the layer. The set of layers have a complex structure of both nesting and depth ordering, and many operations are needed to manage layers and their structure: create, delete, dissolve, clear, change extent, lift, release, etc. This is an expressively powerful approach, and has many techniques similar to those in Tivoli. It does seem to structure material more rigidly than Tivoli, and hence be less agile. The complex layer structure would be difficult to manage and understand in a group setting, especially for people who are not trained in design sketching techniques.

Infinite Zooming Approach. The Pad++ application [2] suggests an intriguing approach (although Pad++ is presented as visualization system). The model would be of a surface in which one creates material at any scale and one can zoom in and out quickly to see an overview and to see details. Many issues would have to be worked out, such as how to rearrange material at different scales or a great distance apart at a given scale (multiple zoom foci might be needed). While potentially powerful, this approach could be very confusing in a group setting, because navigating through this zoomable space seems to be too complex.

2.4. Tivoli's Single Surface Approach

We have decided to stay with a single surface approach in Tivoli, because it is conceptually and visually the simplest to understand. All organizing is done by creating and editing material on the one-and-only surface of the board. Several techniques are provided. Material is grouped by dividing the surface into visually distinct regions by creating and manipulating boundaries. Space is managed by rearranging, shrinking, and collapsing material on the surface. Facile rearranging allows material to use available space; shrinking condenses the space material takes, and it can be applied selectively; and collapsing allows less important material to be hidden.

2.5. Focus on Boundaries

The "material" on the Tivoli board consists of graphical objects: ink strokes, gesture strokes, characters, and icons of various types (see [11] and [9] for more details). Subsets of these graphical objects can be dynamically selected for various purposes, such as for structure operations [9]. There is no explicit concept of a *region* of the board in Tivoli. We did not want to introduce such a concept, because all the system really needs is to identify subsets of graphical objects in order to carry out operations. Boundaries, on the other hand, are natural in Tivoli, since it is easy to create strokes and gestures and have them be interpreted as boundaries.

The main design principle for grouping in Tivoli is to provide the user with *directly manipulable boundaries* and to dynamically compute *implicitly interpreted regions*. This works as follows: When the user selects some graphical objects on the board, Tivoli looks for boundaries and chooses the set of boundaries that define the minimal region around the selection; that is, the region is implicit in the selection. Any subsequent operations on the selected objects are confined to objects contained in the implicit region. For example, consider Figure 2, which shows 9 objects (`A` to `I`) and two kinds of borders. In the left half are straight borders defining three rectangular areas. In the right half are two intersecting loops. These halves are geometrically different but structurally the same. If the user selects `E`, then the implicit region is the middle rectangle or the football-shaped area of the intersecting loops, and the contained set is `{D,E,F}`. If the user, selects `B` and `E`, then the implicit region is the composite left and middle rectangle or the area of the left loop, and the contained set is `{A,B,C,D,E,F}`.

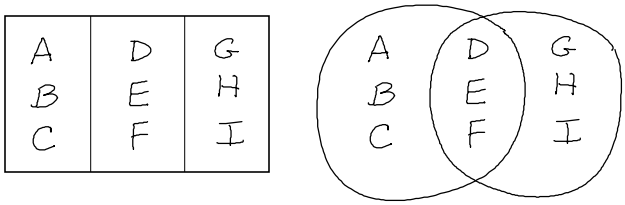

Figure 2. Regions are implicit. Tivoli computes them dynamically by choosing borders to define a region around selected material (see text).

We have introduced various kinds of concrete boundary objects into Tivoli. Some boundaries are just ink strokes, and some are created as special objects by gesture strokes. Some are structured and straight, and some are arbitrarily shaped. Boundary objects are persistent; but temporary selection enclosures can also be considered as boundaries, since they are treated in similar ways.

3. INTERACTION TECHNIQUES

We now describe the interaction techniques for organizing material on the board. There are several categories of techniques: We begin with basic Tivoli editing techniques (Section 3.1). The main techniques have to do with boundaries for defining regions. There are two kinds: structured rectilinear borders (Section 3.2) and freeform enclosures (Section 3.3), including links between enclosures. Lastly, there are techniques for collapsible annotations (Section 3.4).

3.1. Basic Selection and Editing

Operations in Tivoli are triggered by *gestures*, which are interpreted strokes.[4] There are two modes of operations: *freeform operations*, which alter material in a literal, unconstrained way, and *structured operations*, which alter material according to the rules of the structure they are in. In this paper, "structure" means list structure.[5] For example, consider a vertical list of five items, noted abstractly as `[1#2#3#4#5]`, and the action of dragging the 4th item as

4. Tivoli has a *pen mode* which indicates whether strokes are literal ink or interpreted gestures. We have experimented with different ways to change the pen mode: soft buttons on the board, buttons on the pen, and double tapping on the board.

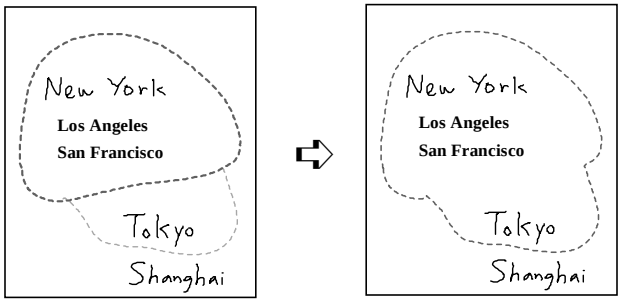

Figure 3. A freeform selection loop (dark grey dashed line) is altered by a **bump** gesture (light grey) to include new material.

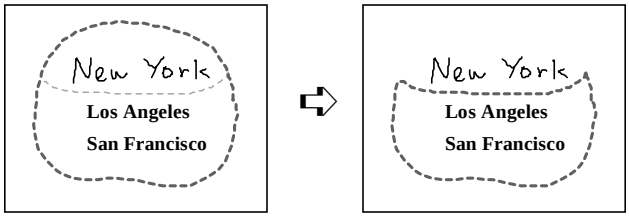

Figure 4. A freeform selection loop is altered by a **bite** gesture to exclude some material.

follows: `[1#243###5]`. In the freeform case, the 4th item is left where it is, even if it overlaps the other items. In the structured case, the list is cleaned up by creating and deleting spaces: `[1#2#4#3#5]`.

Creating Selections. Most operations in Tivoli require that a subset of material be selected first. There are two kinds of selections—*freeform* and *structured*—which determine the mode of the subsequent operations on the selected material. A freeform selection is indicated by a **loop** gesture, resulting in a selection loop (the familiar "lasso" technique). A structured selection is indicated by **bracket** or **L** gestures. The legs of the **bracket** or **L** are extended as far as they can[6] and a selection rectangle is created.

Altering Selection Boundaries. Selection loops and rectangles are temporary boundaries. They can be reshaped by *alteration gestures*. To alter a freeform loop, the gesture must begin and end on the loop. The loop is altered by replacing the segment of the loop between the contact points with the new gesture stroke (see Appendix Note 1 for details). The alteration gesture can be either outside of the loop (a **bump**, as in Figure 3) or inside (a **bite**, as in Figure 4). The material selected is recomputed by what is enclosed in the reshaped loop. To alter a selection rectangle, an **L** or **bracket** gesture is used. The new gesture is matched up with the corresponding part of the selection rectangle (e.g., an "**L**" with the lower-left corner of the rectangle), and the rectangle is reshaped so the matched part is at the location of the gesture (see Figure 5). The selection is recomputed.

5. In [9] we described several kinds of structures: words, lists, tables, and outlines. However, we found that users in meetings almost always used only the list structures. Thus, in order to simplify the user interface, we have disabled all but the list structures.

6. They extend to a border, as we shall see.

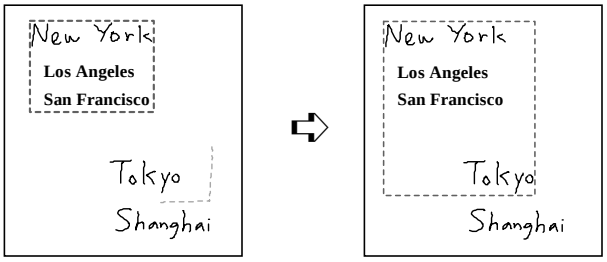

Figure 5. A structured selection rectangle is altered by an **L** gesture to include new material.

Generic Operations. The basic editing operations are move (a **right-wedge** gesture or a **drag**), copy (a **left-wedge** gesture, like a "C"), and delete (a **pigtail** gesture). These are generic in that they are either freeform or structured operations, depending on the type of selection. Further, if there is no selection, the **right-wedge** and **pigtail** are interpreted as operations to manipulate the spacing of list items. A **right-wedge** causes space to open up in a list at the point of the wedge, so that a new item can be written in. A **pigtail** causes the space between items at the pigtail to be compressed. The open-up-space operation has proved to be especially useful.

Scaling Operations. The simplest technique for managing space is to shrink material to allow more to fit on the board. Thus we provide operations to shrink (by a **spiral-inward** gesture) and expand (by a **spiral-outward** gesture) selected material. If the selection is structured, the surrounding material is moved to accommodate the new size of the selected material. For simplicity, the shrink/expand operation is by a fixed factor of $\sqrt{2}$, i.e., two successive shrinks cause the width and height of material to be half the original size.

3.2. Structured Borders

In the earlier version of Tivoli [9], we interpreted long ink strokes as borders. We found that users almost always drew straight vertical and horizontal borders. This made sense, because the structure operations were rectilinear.[7] Thus, we decided to provide straight borders that tessellate a Tivoli page into rectangular regions. The regularity of the tessellation structure allow us to provide more powerful features.

Structured Border Model. A structured border is either vertical or horizontal. A border may cross other borders, and each end of the border may dead-end into another border or extend indefinitely. This border model creates a *rectilinear tessellation*: every region defined by these borders is rectangular (possibly unbounded), and the page is completely tessellated by these regions. Adding and deleting borders must preserve the tessellation property. Border junctions are either crossings (+) or T-junctions (see Figure 6). In the case of a T-junction, the terminating border is said to be a *dependent* of the border it dead-ends into. If a border is deleted, then all of its dependents must also be deleted, in order to preserve the rectilinear tessellation.

7. For example, consider a tapering region containing a list. Some items in the list may not be able to be moved because they are too wide to fit into the narrower parts of the region.

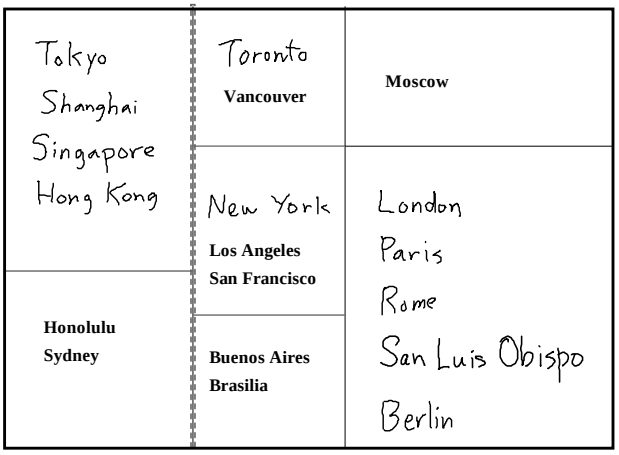

Figure 6. When a structured border is selected (vertical dashed line), its dependent borders are highlighted (light grey).

Creating Borders. A border is created from a **horizontal-line** or **vertical-line** gesture by extending the gesture until it hits another border or extends indefinitely. A border is a graphic object that looks like a straight ink stroke, but it has special behaviors. For one thing, the erase pen mode does not affect it, to protect users from inadvertently deleting it.

Selecting Borders. A **tap** on a border selects the shortest *segment* of the border, determined by the border crossings nearest the point of the **tap**. The full extent of a border is selected by a **double-tap**. When a border or a segment is selected, its dependent borders are *highlighted* (Figure 6 shows a selected border with three dependents). Operations on the selected border can affect its dependents.

Deleting Borders. A border or border segment can be deleted by a **pigtail**, which causes all its dependents to be deleted as well (in Figure 6, this would cause the leftmost five regions to become one region).

Moving Borders. A border or border segment may be moved by a **drag**. However, the movement is constrained; it is stopped when it bumps into any material (in Figure 6, the selected border cannot move beyond the "g" in "Hong Kong" or the "N" in "New York"). This protects the user from inadvertently "reassigning" material to another region (which instead is done by moving the material itself). When a border is moved, its dependents are adjusted to preserve the T-junctions (in Figure 6, the length of three horizontal borders would be adjusted).

Creating Spatial Gaps. A **horizontal-line**[8] gesture does not have to be perfectly horizontal for it to be recognized by the system (after all, the user is drawing freehand). Sometimes the user draws a recognized **horizontal-line** where there is no horizontal spatial gap for a border without intersecting some material. Originally, we would just abort the border creation and require the user to move the material to create the gap, which is tedious. So we changed the operation on **horizontal-**

8. This discussion also applies to the **vertical-line** gesture.

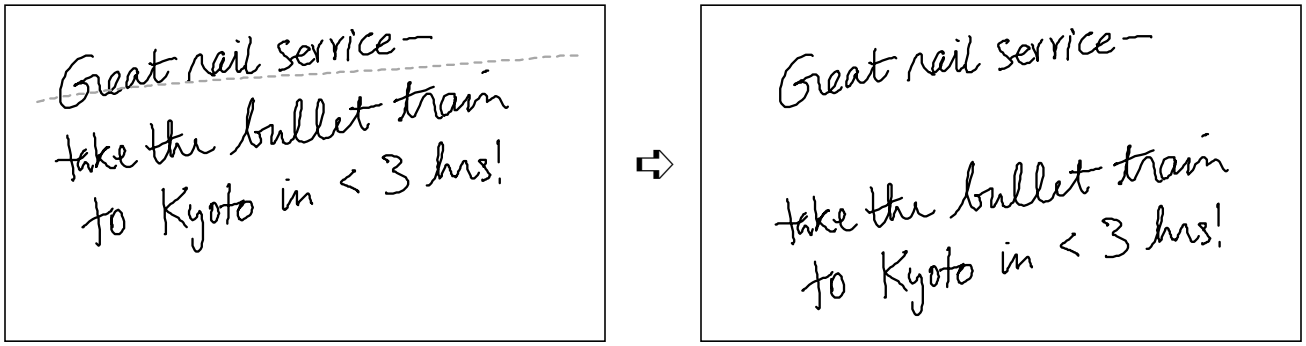

Figure 7. A **horizontal-line** gesture creates a horizontal spatial gap, allowing a border to be created or structured selections to be made.

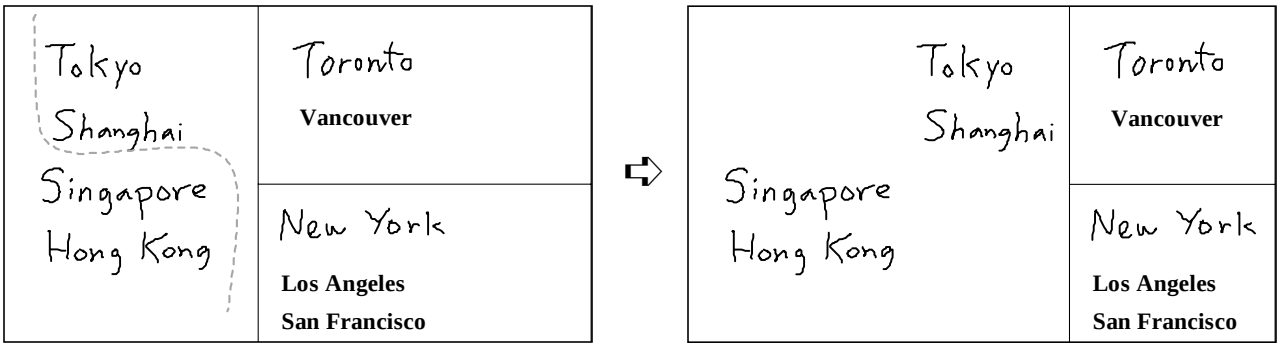

Figure 8. A **wiggly-line** gesture from top to bottom causes material to move to the right to create a vertical spatial gap (in effect, two columns).

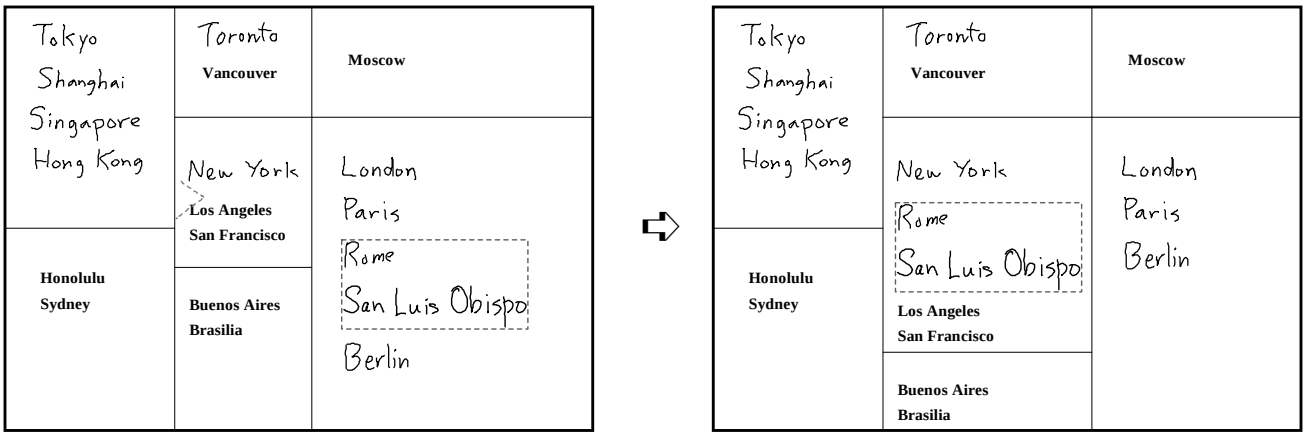

Figure 9. A structured move of material into a region causes it to be expanded and adjacent regions to be pushed rightward and downward. The move depicted here was triggered by a **right-wedge** gesture in the center region, but the user could also have just dragged the selection into place.

line to move material to create a horizontal gap if there is no gap (see Figure 7). Then a second **horizontal-line** is required to create the border. The reason for this is that the gap-creating operation proved to be useful without the border; it is used to create gaps between list items, so that messy lists can be manipulated with the structure operations. Finally, we became more liberal with our recognition and allowed a **wiggly-line** that goes from border to border, which allows more "dramatic" movement of material to help in organizing it. For example, Figure 8 shows a list being rearranged into two separate columns.

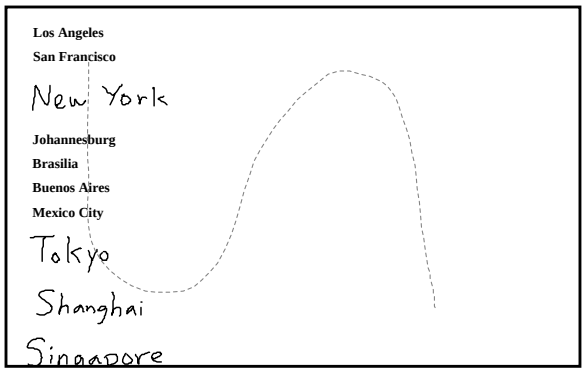

Figure 10. A **large-sideways-S** gesture causes a long list to be reformatted to fit onto the board. The resulting multicolumn list is shown in Figure 11.

Automatic Expansion. A structured move of material into a rectangular region will cause the region to expand, if needed, to accommodate the material. Expansion is accomplished by pushing borders and regions rightward and downward (as shown in Figure 9). (See Appendix Note 2 for details.) This expansion regime is easily understandable and has proved to be very useful to give the user freedom to move material as needed without worrying about what fits where. On the other hand, if the user wants to keep the size of a certain region fixed, the borders of this region can be changed to *anchored borders*, which are not automatically moved.

Flow Borders. A *Flow border* is another type of border. It is a vertical border (shown as a dashed line) that allows material to automatically "flow through" it, thus providing for multicolumn lists. When flow borders divide columnar regions, the material in those regions is treated as a single list. For example, structurally deleting items in the leftmost column

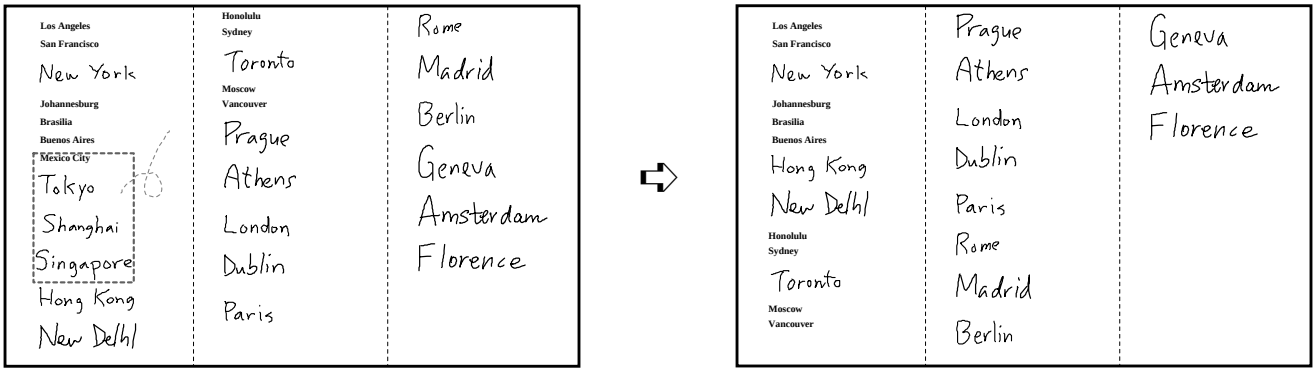

Figure 11. Vertical flow borders allow items in a multicolumn list to be regarded as a single list. Deleting items (**pigtail** gesture) in column 1 causes items to flow leftward.

causes items from other columns to flow leftward to fill in the space (as shown in Figure 11).

Shrink-to-Fit Operation. One of the most useful operations for space management uses flow borders. Consider a long list created freehand resulting from a brainstorm, during which the page was scrolled upward as items were created. People usually write on the board at a size convenient for writing, which is much larger than is needed for viewing. The shrink-to-fit operation reformats the long list into a multicolumn list that fits onto the board, so the whole list can be seen and manipulated. (See the Appendix Note 3 for the algorithm.) This operation is triggered by a **large-sideways-S** gesture. For example, Figure 10 shows the visible part of a long list. The shrink-to-fit operation reformats it to look like Figure 11.

3.3. Freeform Enclosures

Structured borders are powerful but somewhat rigid. When quickly trying to group a few items, users may not want to be confined to rectangles, and they may not want to tessellate the whole page surface. Freeform enclosures are more appropriate for this purpose.

Enclosure Model. A freeform enclosure is just an ink stroke that forms a loop of any shape. It behaves as any other stroke (e.g., it is erasable), except for certain operations that check for enclosures. An enclosure defines its interior to be a region, and the material enclosed is thus grouped. For example, if an enclosure contains a list, structural operations on the list will be confined to the enclosure. However, note that an item or a list can expand outside the confines of the enclosure. Unlike structured borders, the shape of enclosures is not automatically altered, because they are freeform and do not conform to any rules. From the outside, an enclosure is just another graphical object (e.g., a column of enclosures can be manipulated as a list of items). Enclosures can be nested. Enclosures can also intersect, making different regions possible (as in Figure 2).

Creating, Selecting, and Moving Enclosures. An enclosure is created just by drawing it. An enclosure and its contents can be selected by a **loop**, but it is quicker to just **tap** on the enclosure stroke, which causes a selection loop to be automatically created. A selected enclosure (plus its contents) is moved by a **drag**.

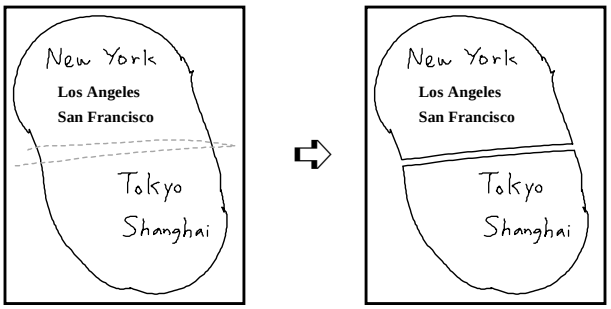

Figure 12. A **back-and-forth** gesture splits a freeform enclosure into two enclosures.

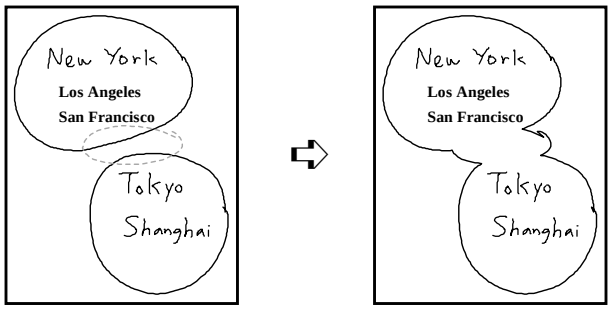

Figure 13. A **loop** gesture fuses multiple enclosure into a single enclosure.

Altering, Splitting, and Fusing Enclosures. An enclosure is a boundary, and can thus be altered with the same gestures that alter selection loops, **bite** and **bump**, as in Figures 3 and 4. Further, an enclosure can be *split* into two enclosures by a **back-and-forth** gesture (see Figure 12). Two or more enclosures can be *fused* into a single enclosure by a **loop** gesture that cuts through both enclosures (see Figure 13).

Links Between Enclosures. Sometimes it is useful to draw a link between enclosures to indicate some kind of relationship between them. A *link* is simply an ink stroke whose end points touch two enclosures.[9] This extends the enclosure model to include a *node-link structure*. This structure is supported as follows. A **tap** on an enclosure (node) not only selects the enclosure, but also highlights any links from the enclosure (as shown in Figure 14). When a selected enclosure is moved, the links are reshaped to preserve the link connectivity. This reshaping also preserves the original shape characteristics of the links (see Figure 14; also see Appendix Note 4 for the algorithm). If one does not want a particular link to be changed when an enclosure is moved, it can be "unhighlighted" by a **bite** gesture on the selection loop where the link crosses it. A link's connection points can be manually changed by selecting one of its end points (by a **tap**) and dragging the end point to another place; the link is automatically reshaped.

3.4. Collapsible Annotations

The final organizational technique is to deal with annotations that users often make on the board. The problem with annotations is that they take up space and they often "mess up" a structure, making structured operations unworkable. One way to deal with this problem is to provide a technique for collapsing annotations so they don't take up space but still are accessible.

Collapsible Annotation Model. Applying a *collapse operation* to selected material results in the material being replaced by a small *container icon* (a graphic object subject to the normal Tivoli editing operations). A container icon is a small box with a unique integer in it (see Figure 15), and we usually talk about it metaphorically as a "footnote."[10] It is to be

9. Actually, only one end point need touch an enclosure; the other may dangle. This allows incomplete diagrams.

10. There are also operations to display all the contained materials at the bottom of each page as footnotes or on a separate page as endnotes.

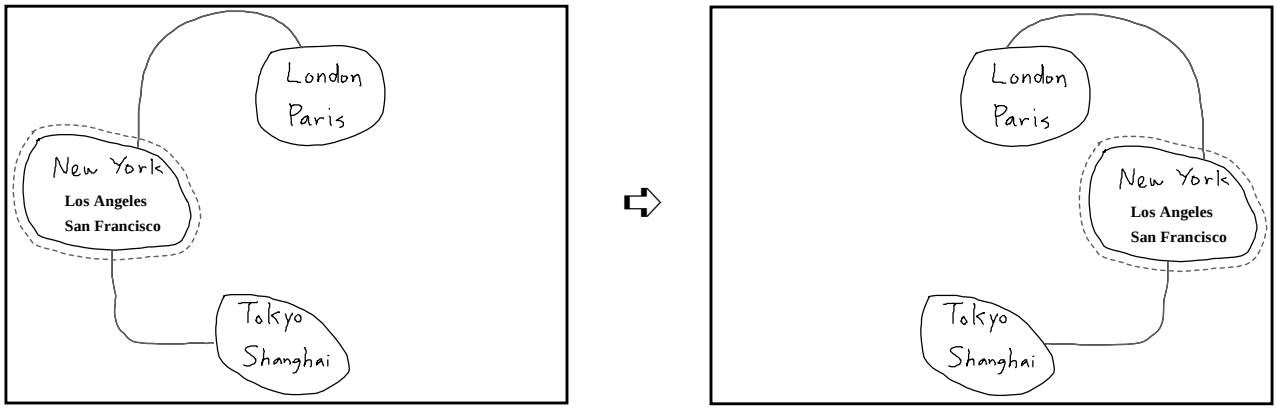

Figure 14. When an enclosure node is moved, links are reshaped to maintain the connections.

thought of as "containing" the collapsed material. Applying a *display operation* to a container icon causes the contained material to be displayed on an opaque *overlay* (as in Figure 15c).[11] An overlay can be dragged around or dismissed. The overlay itself cannot be edited; but the material on the overlay can be *pasted* back onto the board, where it is fully editable (and re-collapsible if desired).

Gestures. The collapse operation is triggered by a **balloon** gesture, as shown in Figure 15a (the original version of the container icon looked like a tiny cartoon balloon). A **tap** on a container icon selects it. **Double-tap** on a container icon displays its contents on an overlay. **Drag** moves an overlay, and **tap** dismisses it. The **paste** button on the overlay causes the material to "drop" from the overlay onto the board—visually, the overlay frame seems to just disappear—and the container icon is deleted.[12]

Structured Annotations. Material can also be collapsed and put back in a list structure. When the collapse operation is applied to a structured selection, the list items selected disappear, the space is closed up, and the container icon is placed at the rightmost point of the list item above the selection. The **insert** button on an overlay puts the material back onto

11. This is the only object in Tivoli that is allowed to opaquely overlap other objects on the board.

12. The user is warned if there is already material underneath the overlay. Semi-transparency would be useful here.

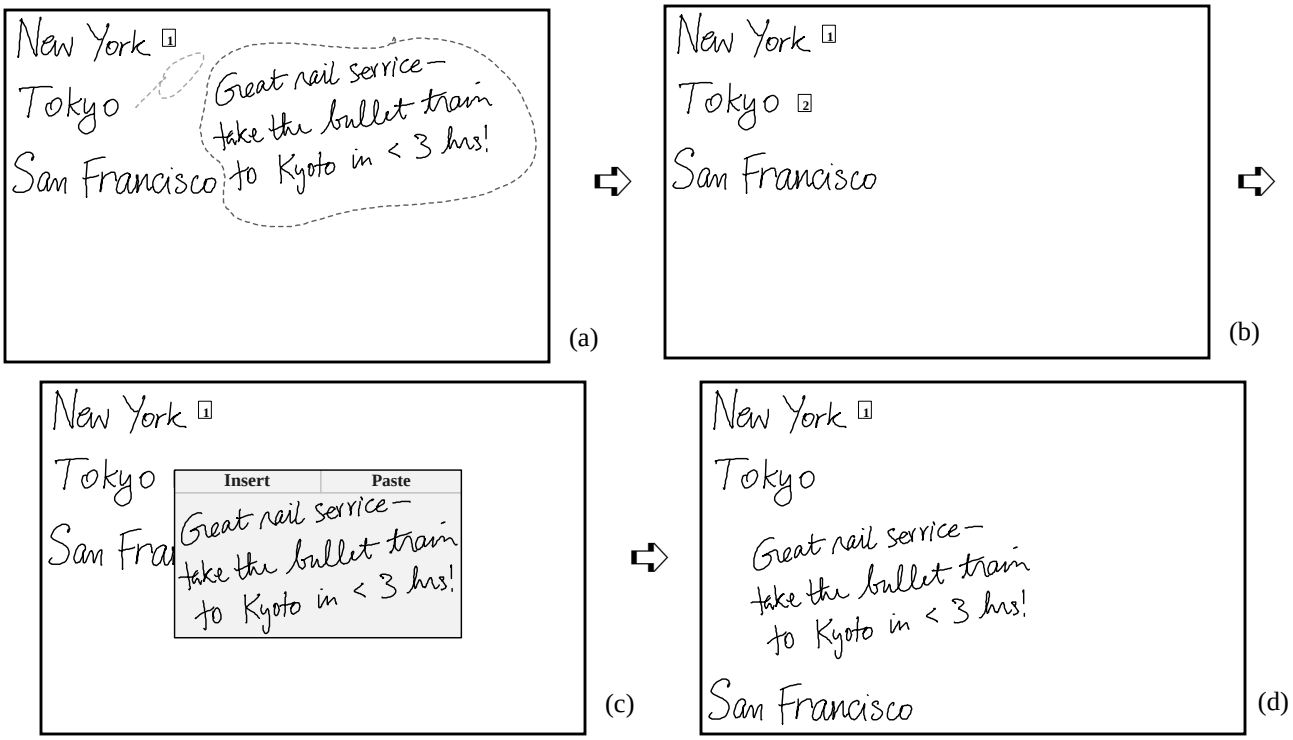

Figure 15. A **balloon** gesture (a) collapses the selection into a "footnote" icon (b). A **tap** on the footnote icon displays its contents in an overlay (c). **Insert** inserts the contents into the list (d).

the board as an inserted list item, i.e., space is opened up for the item. Using these operations on indented lists provides a primitive outlining facility. Collapsing and putting back are independent operations, and the freeform and structured versions of these can be intermixed, as is seen in Figure 15d.

4. EXPERIENCE, TESTS, AND TRADE-OFFS

The design, development, testing, and refinement of these techniques was an iterative process over a period of two years. Observations of meetings and experiences with the basic Tivoli facilities provided a continual stream of goals for designing meeting tools. We tested early versions of these techniques on ourselves. Robust versions were included in regular Tivoli releases as they became ready, so they could be tried out in various meetings. The structured border facilities and a somewhat specialized version of the collapsible annotation techniques are in regular use in our long-term meeting-support case study reported in [8].

User Tests. We wanted to test the learnability and usability of the suite of interaction techniques by novice users. We developed an explain/demo/practice training sequence and a set of test tasks exercising all the techniques. We ran three pilot users to debug the test procedures and the user interface (e.g., these tests led us to adjust the precedence among the gesture recognition rules). We then ran eight test users: seven non-technical administrative people at PARC and one student intern. None had any LiveBoard experience (except one who did a similar test for us three years earlier). There were eight individual 90-minute test sessions, consisting of training, tasks, and debriefing. The training was at the user's own pace; the user had to understand and demonstrate the use of the techniques. The test tasks simulated the user being a meeting scribe who was asked to rapidly make a long series of changes to material on the board, which required the use of all the techniques described in this paper.

Main Test Results. The basic result is that the techniques are understandable, learnable, and usable by non-technical, novice users. All users were able to learn all the techniques to their own level of comfort in an average of 43 minutes.[13] They found the techniques understandable and usable. They all were able to perform all the scribe-editing tasks. Most commented that the experience of using the LiveBoard was enjoyable. Different users found different features more compelling for their own tasks—list editing, node-link manipulation, annotations—but the overall favorite feature was shrink-to-fit.

Design Trade-Offs. Users encountered a myriad of small difficulties that they easily worked through, but no obvious changes were suggested. Rather, the tests highlighted several design trade-offs we had made. The ink-vs-gesture pen mode and the freeform-vs-structured operation mode, and the generic operations (e.g., **line** to create gaps and borders) all served their purposes, but at the cost of users having to be vigilant. Undo allowed easy recovery from lapses of vigilance. Our use of gestures means that users have to recall them (vs. recognizing items in a menu). We gave the test users a crib sheet for reference, which was helpful. Prompting techniques, such as marking menus [6], can only partially help, because many of our gestures must be drawn in a spatial context to indicate their meaning. In these decisions we usually traded some simplicity for openness and agility. The basic trade off is that users need to invest some learning time to attain the level of performance we aim to provide; this not a walk-up-and-use interface.

13. Breakdown of the learning time: basic Tivoli techniques, 19 min.; borders, 9 min.; enclosures, 10 min.; and annotations, 5 min.

5. CONCLUSION AND FUTURE WORK

There are several features we understand how to design, but have not yet implemented. For example, dragging a structured border should allow the user to flexibly shrink a region. At present a border stops moving when it bumps into material. What it should do is to continue to be dragged, but to indicate how the material in the region must be reshaped to accommodate the new border position: first empty spaces will be squeezed out, then the material will be shrunk, then multiple columns will be created. Another example is that freeform enclosures should adjust their shape when inserted material does not fit.

One intriguing idea that needs investigation is the notion of *implicit spatial boundaries*. The techniques presented here involved explicit boundaries (structured borders and freeform enclosures). While these do work well, there are situations when it is "obvious" to everyone that material is visually grouped, and it seems tedious to have to explicitly create borders. Open spaces on the board can be regarded as implicit borders (consider, e.g., Figures 6 or 8 without the borders). There are visual processing techniques for tracing out the implicit borders in empty spaces[14] and for identifying spatial groupings by proximity (e.g. [14]). Such techniques need to be explored to see if they match user perceptions well enough to be more helpful than harmful.

In summary, we have presented a suite of techniques for organizing materials on a pen-based electronic whiteboard. These techniques provide a wide range of functionality for grouping and rearranging material on the board and for using the limited space of the board most effectively. The techniques have been implemented and integrated into the Tivoli application, user tested, and made reliably usable. These techniques are not for walk-up users. They must be learned. But once learned, they allow the user to be facile enough to support a fast-paced meeting situation.

ACKNOWLEDGMENTS

We would like to thank our colleagues in the Collaborative Systems Area (CSA) of PARC for many discussions and suggestions, especially Chuck Hebel, Beverly Harrison, and Ron Mann. We thank Jim Mahoney and Eric Saund for discussing and exploring many issues of graphics and recognition with us. Finally, we thank our enthusiastic pilot and test users and members of CSA for putting up with an ever-changing whiteboard in our meetings.

14. Jim Mahoney, personal communication.

REFERENCES

[1] *aha! InkWriter Handbook* (1993). Mountain View, CA: aha! software corporation.

[2] Bederson, B. B., & Hollan, J. D. (1994). Pad++: A zooming graphical interface for exploring alternate interface physics. *Proceedings of UIST'94*. New York: ACM.

[3] Elrod, S., Bruce, R., et al. (1992). LiveBoard: A large interactive display supporting group meetings, presentations, and remote collaboration, *Proceedings of CHI'92*. New York: ACM.

[4] Gross, M. D., & Do, E. Y. (1996). Ambiguous intentions: a paper-like interface for creative design. *Proceedings of UIST'96*. New York: ACM.

[5] Kramer, A. (1994). Translucent patches—dissolving windows. *Proceedings of UIST'94*. New York: ACM.

[6] Kurtenbach, G. (1993). The evaluation of an interaction technique based on self-revelation, guidance, and rehearsal. Ph.D. Thesis, University of Toronto.

[7] Marshall, C. C., Shipman, F. M., & Coombs, J. H. (1994). VIKI: Spatial hypertext supporting emergent structure. *Proceedings of ECHT'94*.

[8] Moran, T. P., Chiu, P., Harrison, S., Kurtenbach, G., Minneman, S., & van Melle, W. (1996). Evolutionary engagement in an ongoing collaborative work process: a case study. *Proceedings of CSCW'96*. New York: ACM.

[9] Moran, T. P., Chiu, P., van Melle, W., Kurtenbach, G. (1995). Implicit structures for pen-based systems within a freeform interaction paradigm. *Proceedings of CHI'95*. New York: ACM.

[10] Saund, E., & Moran, T. P. (1994). A perceptually-supported sketch editor. *Proceedings of UIST'94*. New York: ACM.

[11] Pedersen, E., McCall, K., Moran, T. P., & Halasz, F. (1993). Tivoli: An electronic whiteboard for informal workgroup meetings. *Proceedings of INTERCHI'93*. New York: ACM.

[12] Saund, E., & Moran, T. P. (1994). A perceptually-supported sketch editor. *Proceedings of UIST'94*. New York: ACM.

[13] Shipman, F. M., & Marshall, C. C. (1993). Formality considered harmful: experiences, emerging themes, and directions. Technical Report, Department of Computer Science, University of Colorado.

[14] Shipman, F. M., Marshall, C. C., & Moran, T. P. (1995). Finding and using implicit structure in human-organized spatial information layouts. *Proceedings of CHI'95*. New York: ACM.

[15] Wolf, C., Rhyne, J., & Briggs, L. (1992). Communication and information retrieval with a pen-based meeting support tool. *Proceedings of CSCW'92*. New York: ACM.

APPENDIX: MISC. IMPLEMENTATION NOTES

1. Altering Enclosures. When an enclosure is altered by a gesture, the gesture's end points break up the enclosure into two pieces. For a **bump** (Figure 3) it is clear what to do—the enclosure always expands. However, for a **bite** (Figure 4), it is ambiguous which of the pieces should be retained, especially when the **bite** divides the enclosure into somewhat equal pieces; users' perceptions vary on this issue. Our implementation keeps the piece with the longer arc length. We are exploring various other criteria: (a) keep the piece that merges best with the **bite** in terms of minimizing the angle change at the two contact points, (b) keep the piece so that the resulting enclosure has the largest area, (c) keep the piece so that the resulting enclosure contains the most material.

2. Automatically Expanding Structured Regions. The principle is to preserve the highest level border structure. Connected colinear border segments should remain connected. In the expanded region, the right and bottom borders along with their connected colinear borders determine the neighboring regions to be pushed out; these in turn determine the next neighboring regions; and so on. For example, in Figure 9, the vertical border to the right of the destination must be moved, and since it is colinear with the border above it (to the right of Vancouver), both are moved in concert. The border below the destination must also be moved, but it is free to move independently.

3. Fitting a List on the Board. The material on the Tivoli page is first interpreted as a list by grouping the objects into list items. The number of columns, *N*, for formatting the visible board is computed by finding the *N* that maximizes the shrink factor (i.e., minimizes the degree to which the material is scaled down). For a specific *N*, the shrink factor is determined by the height of the list and the width of the widest list item. Finally, flow borders are created, the material is scaled by the shrink factor, and the list items laid out into the columns. (See Figures 10 and 11.)

4. Reshaping Links. When an enclosure is moved, the end point of its link is also moved to preserve its attachment to the enclosure; and all the segments comprising the link must also be moved to preserve the shape of the link. We found that simply scaling the link segments has the drawback that it sometimes causes drastic distortions. Instead, we employ an additive (rather than multiplicative) method where the displacement of the enclosure's move is additively distributed among the link segments. (See Figure 14.) If the resulting link intersects the enclosure, other steps are taken to remove the unsightly intersections. First the attached endpoint is flipped to the other side of the enclosure. If it still intersects, then the link is reshaped as above. If it still intersects, the part of the link crossing the enclosure is truncated, so that no intersection remains.