Flatland: New Dimensions in Office Whiteboards

Elizabeth D. Mynatt[1], Takeo Igarashi[2], W. Keith Edwards[1], and Anthony LaMarca[1]

[1]Xerox Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304, USA
+1 650-812-4405
[mynatt,kedwards,lamarca]@parc.xerox.com

[2]University of Tokyo
7-3-1 Bunkyo-ku, Hongo
Tokyo 113-8656, JAPAN
+81 3 3812 2111 ext. 7413
takeo@mtl.t.u-tokyo.ac.jp

ABSTRACT

Flatland is an augmented whiteboard interface designed for informal office work. Our research has investigated approaches to building an augmented whiteboard in the context of continuous, long term office use. In particular, we have pursued three avenues of research based on input from user studies: techniques for the management of space on the board, the ability to flexibly apply behaviors to support varied application semantics, and mechanisms for managing history on the board. Unlike some previously reported systems, our design choices have been influenced by a desire to support preproduction—rather than final production—work in an office setting.

Keywords: pen-based computing, whiteboards, ubiquitous computing, light-weight interaction, Flatland

INTRODUCTION

Whiteboards are ubiquitous tools in informal office work. By this description, we envision a whiteboard in a private office with four principal characteristics based on observations of whiteboards in an office setting [17]. First, the whiteboard acts as a working area and repository to support *thinking* tasks such as sketching out a paper as well as quick *capture* tasks such as jotting down a reminder. As an interface for thinking, whiteboards are often used for *pre-production* tasks where the emphasis is on understanding ideas, tasks or concepts and the production work based on that thinking is accomplished in a different arena. Examples are sketching out an algorithm that is later coded using a computer, planning out a set of tasks whose schedule is captured in a planner or email message, drafting ideas for a web page or organizing concepts that are later put into prose using a word processor. These pre-production artifacts would not even be thought of as drafts, but as what comes before a draft. Due to its large visual surface and simple input capabilities, the whiteboard can be appropriated for many pre-production activities It lacks, however, mechanisms to support specific tasks, such as managing to-do lists or organizing an outline, that be found in many desktop and PDA tools.

Second, the content on the whiteboard, whether a sketch or a reminder, has particular characteristics stemming from its creation and use. We refer to this type of content as *everyday content* and loosely define it as the continually shifting set of information in your office that you use to do the majority of your tasks. This flow of information is often incomplete, unnamed, informal, heavily context-dependent and transient. Examples are notes, to-do lists, drafts, reminders, sketches and the like that are in sharp contrast to archival material typically filed either electronically or physically. As an interface for everyday content, common whiteboards afford quick capture of material with an informal look-and-feel without the overhead of naming, filing and formatting. However, the context surrounding its creation (e.g. “What was I thinking when I wrote down “Rosebud”?”) must be remembered by the user.

Third, whiteboard material is generally clumped into various clusters on the whiteboard. Some clusters, such as an important phone number or a sketch, may be long-lived compared to other clusters, such as an illustration for a visitor or a quick calculation that is erased within a day. Often there are “hot spots” that are erased while bordering content persists. Since whiteboard content is rarely obscured and quite visible in the office, its presence acts as a reminder. Some people depend on this informal awareness. As an interface for clusters of content the whiteboard’s large visual space affords delegating portions for longer-lived content. Additionally, its vertical orientation makes it less likely to be obscured in contrast to the horizontal orientation of a desk that affords stacking and obscuring layers of content.

Fourth, in an individual office, the whiteboard functions as a personal device, but it also has a semi-public role for office visitors and informal meetings. Whiteboards are typically visible to an office visitor, and can be the focal point for an informal office discussion whether one person is illustrating ideas for the other, or multiple people are contributing to the board’s content. The placement of the whiteboard significantly affects its public role, whether it is visible and writable for the typical office visitor. Since the whiteboard can be easily erased, private material can be quickly deleted when an office visitor arrives, although the material would then be lost.

These four characteristics—thinking or pre-production tasks, everyday content, clusters of persistent and short-lived content, and semi-public to personal use—underlie our model of informal whiteboard use in an individual office. This use of whiteboards is quite distinct from the use of desktop computers, and still varies significantly from the use of personal, pen-input devices given a whiteboard's public role and large, continually visible surface.

Although the whiteboard is a flexible tool for quickly capturing input with an informal look-and-feel and its large visible surface supports parallel tasks including awareness, its utility past that point is limited. Its content cannot be saved and retrieved, or even moved out of the way. As simple strokes on a board, all input is treated the same whether it is a to-do list, a series of calculations or an illustration. Additionally, many people complain about illegible writing and the poor visual quality of drawings.

Our goal in this work is to create an augmented whiteboard called "Flatland" to better support typical whiteboard use. Our initial hardware configuration is a SmartBoard (tm) coupled with a projector. The SmartBoard is a touch-sensitive whiteboard that accepts normal whiteboard marker input as well as stylus input. Captured strokes are then projected onto the board. Given this platform and our characterization of whiteboard use, our design goals were:

- To support a low threshold for initial use while making increasingly complex capabilities available. At the simplest level, Flatland should act like a normal whiteboard where you can walk up to it and write on it. In general, its look-and-feel should remain simple and informal to support the nature of pre-production tasks.
- To provide a look and feel appropriate for informal whiteboard tasks and distinct from production-oriented tools such as a desktop computer.
- To support informal office pre-production tasks such as to-do lists and sketching.
- To support clusters of content on the whiteboard. These clusters, or segments, may be created for different purposes, at different times and by different people.
- To support the use of everyday content by creating context-aware interaction and infrastructure. As unnamed material, content could be stored and retrieved based on its salient context (spatial location on board, time of creation, people present) instead of requiring a file name.
- To support the flexible management of a dynamic whiteboard space such as freeing up whitespace for new input while maintaining the visibility of current content.
- To support a range of use from semi-public to private use, such as providing a means to get a clean board without losing current content.

Our efforts are motivated by two threads of inquiry. First, while there has been previous research in computer-augmented whiteboards, these efforts have principally focused on the use of whiteboards in meeting or classroom settings [1][13]. Our goal is to investigate a computer-augmented whiteboard design in the setting of an individual office. Second, we hope to contribute to the growing body of research in ubiquitous computing [20], augmented reality [4] and tangible interfaces [10] The whiteboard itself is a ubiquitous tool in many office environments. In our design, we attempt to extend the existing whiteboard look-and-feel with an interface whose feel and aesthetics match its role in informal office work.

This paper is organized as follows. In the following section we introduce the basic concepts in Flatland to give the reader a sense of its functionality and feel. We then discuss Flatland's design in three parts. First, we describe facilities for managing the virtual projected space on the physically constrained board. Key concepts here are automatically segmenting user input as well as mechanisms for shuffling whiteboard segments while maintaining visibility of those segments.

Second, we describe how we support different tasks on the whiteboard by allowing the user to apply different "behaviors" to input strokes. While behaviors provide specific interpretation of input strokes, the whiteboard maintains a unified appearance and feel. To support concurrent tasks and flexible reuse of content, behaviors can be composed, in parallel or sequentially.

Third, we describe how whiteboard segments are stored based on their context, and strategies for retrieving those segments. Each segment is tagged with context markers (e.g. spatial location on board, people present, time of day) that can be used as the basis for later retrieving that segment. Since time is often a key contextual clue, we provide a strategy for searching segments across time where the search "snaps" to interesting points in the timeline.

We then discuss related work, paying particular attention to other whiteboard and pen-input interfaces as well as general work in blurring the boundaries between the physical and virtual realms. We close the paper by detailing the status of the current system, summarizing the key contributions of this work, and discussing potential avenues for future work.

FLATLAND BASICS

In this section, we give a general description of Flatland. We shall describe many of these features in greater detail in later sections. As Flatland is designed for long-term use, this scenario illustrates using Flatland over a period of days.

> *On Monday, Ian walks up to his new Flatland board and jots down some quick notes using the stylus just as if he were using an old-fashioned whiteboard. Flatland automatically groups his notes into a segment and draws an informal border around them. On Tuesday, he writes down a to-do list creating another segment. On Wednesday afternoon, he sketches a map to his house for an office visitor. On Thursday, he uses the time slider to replay items that he has checked off so that he can write his status report.* (See Figure 1).

There are two modes of stylus input. The primary mode is for drawing strokes on the board. The secondary mode is activated by holding a button and is used to create meta-strokes. These meta-strokes form gestures[1] that are used for managing the board's visual layout as well as for applying behaviors to segments. The tap gesture causes a pie menu to

1. Although we took great care in designing a small gesture set, we will not discuss this process in detail due to space constraints.

FIGURE 1. Using Flatland

be displayed and directional gestures are short cuts for the pie menu (i.e. a marking menu). (See Figure 2)

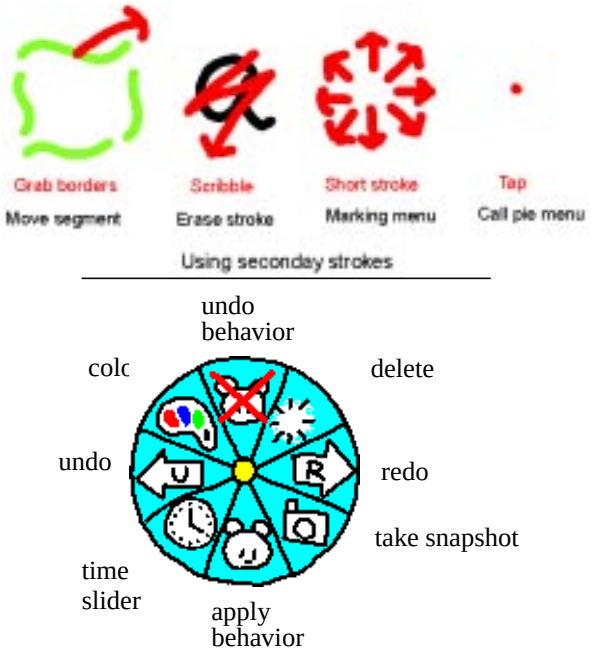

FIGURE 2. Gestures and Pie Menus

MANAGING SPACE

As a computationally-enhanced whiteboard, Flatland provides a flexible and dynamic writing surface. Since the presence of material on the whiteboard often acts as an informal reminder, we opted for strategies that allow users to acquire whitespace while still ensuring the visibility of existing content.

The basic conceptual building block here is that of a whiteboard *segment*—a cluster of content. Flatland creates segments automatically when users write on the board. The segments are not allowed to overlap and can be moved by the user or the system. Flatland also automatically shrinks segments to create more whitespace on the board.

Auto-Segmenting

Most whiteboard users manage multiple clusters of content on their whiteboard. By taking advantage of the large visual surface, they use different parts of the board for different tasks at different times. Since this process of dividing up the board is a lightweight, implicit interaction, we wanted to provide *automatic* mechanisms for generating these clusters or segments. Although users do not *need* segments to write on the board, these segments are the basic building block for managing the board's spatial layout, adding additional behaviors to the whiteboard, and retrieving content.

Given a fresh, clean board, when the user begins writing a border appears, denoting a new segment. The border grows to encompass additional strokes of input if the subsequent strokes seem to fall in the same segment. Several factors could determine into what segment strokes belong:

- Ink Density: Given a new stroke, the system could balance maximizing ink density in each segment while minimizing the number of segments on the board.
- Active Segment: If the user has been interacting with the board recently, there could be an active segment that expects subsequent input.
- Time: The system could be biased to creating a new segment if significant time has passed since input in that area of the board.
- Content: Similar content could be kept together.
- Spatial Arrangement: The system could expect subsequent input following cultural norms. For example, lists would proceed top to bottom, left to right per Western writing norms.

We explored these factors in our design, interaction mock-ups, and implementation. We opted for a simple design where existing segments are grouped into bounding boxes.[1] The bounding box for the active segment is expanded to anticipate new strokes to that segment. If new strokes cross the border of the expanded segment, they are included. Currently the extra space in the active segment only follows Western writing conventions with additional space to the right of and below existing strokes. Pen input in an inactive segment, makes it active with an expanded input area. Input to the "root" space of the board (called the *root segment)*) generates a new segment.

We opted for this simpler mechanism since it seemed to do "the right thing" most of the time and the interaction is predictable. We also provide simple facilities for joining and splitting segments that act as error-recovery mechanisms. In general, automatic segmenting does not significantly raise the threshold for initial use, and it does provide a base for supporting interaction tailored to natural clusters of content.

Our design differs from other whiteboard interfaces [14] as users do not have to explicitly group material and clusters are not recognized by their content (e.g. recognizing a list or a table). Although the system is working in the background, the feel in Flatland is that the user is driving the interaction since the auto-segmenting mechanism is simple and the user

1. Although aesthetically marked as thick, wavy lines, the segments are rectangles easing coding complexity and performance costs. The discrepancy has not been problematic.

can easily activate segments to add more content, or create a new segment by tapping on the root segment.

Active and Inactive Segments

Flatland supports active and inactive segments where there can be one or zero active segments at a time. Active and inactive segments differ in their behavior and appearance. First, the border of the active segment is much brighter and visibly noticeable than the lighter borders of the inactive segments (see Figure 1). We experimented with a number of approaches in delineating segments, including not showing borders at all, and only showing the border for the active segment. Our informal use favored showing borders for all segments since they convey a great deal of information about the state of the board. Since Flatland is biased to including new strokes in the active segment, visually marking the active segment informs the user where the bias resides.

Inactive segments are sized to take as little screen real estate as possible while still showing their content. In contrast, active segments expand to include white space. This expansion visually marks the bias for new input to fall into the active segment. When the segment becomes inactive, it shrinks to remove the surrounding whitespace.

One lesson from informal use pertains to deleting strokes in an active segment. In our first design, the active segment would shrink based on the deleted material. This behavior was disturbing when followed by more pen input since the input area was now smaller, and new strokes in the area of the deletion, which was the last location of the pen, might now fall outside of the active segment. Currently, the active segment can only become larger, taking the more compact presentation when made inactive

Moving, Squashing and Flipping

Users can move segments with a standard select and drag motion. To reduce the complexity of working on the whiteboard, and to ensure visibility of each segment, segments are not allowed to overlap. As other segments are *bumped*, they move out of the way. According to [17], many people use their whiteboard as a surrogate memory. If segments were allowed to overlap, important reminders might be completely obscured.

Although we did not want users to obscure content via overlapping segments, we still needed mechanisms for creating more whitespace. To meet this need, Flatland automatically squashes segments as they bump into the border of the board. With each bump the segment is scaled down until it reaches a minimum size. (see Figure 3) Flatland is biased to squashing segments that have been inactive the longest. With further user testing, we will determine if users need to ever explicitly squash segments or if the automatic squashing is sufficient. To gain more space, users can also explicitly remove segments from the board.

Users can also flip to a new board by dragging the right or left border of the board [18]. The metaphor is a long roll of paper partitioned into *flip charts*. This mechanism provides a fast way to get a clean board, and perhaps quickly hide content from an office visitor.

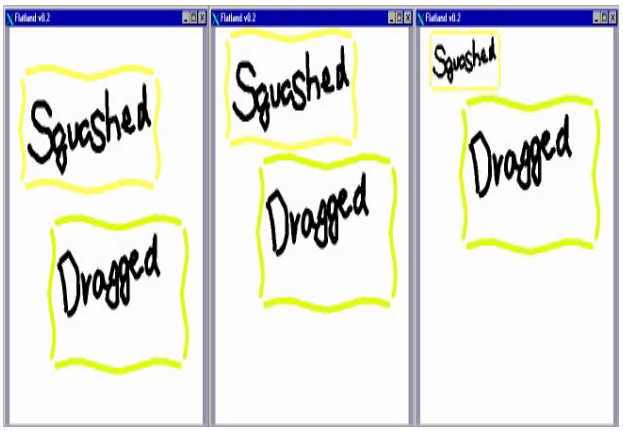

FIGURE 3. Segments squash to reduce size

In interviews regarding how users thought a virtually-extended whiteboard should be organized [17], they expressed two main desires. First they did not want to lose material given the whiteboard's role of surrogate memory and reminder. Some explicitly asked for the ability to squash existing content out of the way so they could gain more whitespace. Second, they wanted a constrained virtual space only four to eight times larger than their existing whiteboard. Flatland's design addresses these needs and concerns. In its principal use as a single board, the borders provide an intuitive affordance for scaling segments to gain space. The flip charts are a quick way to acquire a clean board for an office visitor or to create secondary boards for particular projects and activities. We opted against using a scrolling or zooming space (a la Pad++ [2]) to minimize the potential for losing track of whiteboard clusters.

APPLYING BEHAVIORS

One of the primary design philosophies of Flatland is that the whiteboard should be usable exactly like a normal, physical whiteboard, and yet should be able to provide powerful assistance with everyday tasks as needed.

The first goal is one of simplicity—physical whiteboards allow any type of stroke to be made anywhere, in a completely free-form manner, without interpretation or regard to semantics. Simplicity is one of the traits, along with size and the ability to serve as a focus of context in workgroups, that makes physical whiteboards so appealing. This freeform use of space and natural input is often lacking in existing "desktop" applications, even ones that are superficially similar, such as paint programs, that typically reserve parts of the screen for controls, and embed active elements such as move/grow handles in the content area of the display.

Conversely, the second goal is one of specificity—-to provide assistance for common tasks Further, different tasks on the whiteboard likely require different support—akin to running multiple applications, in parallel, on a desktop computer. However, pre-production tasks require informal, simple, and fluid interaction distinct from most production-oriented desktop applications.

The tradeoffs between providing specific support for certain tasks while creating an informal, simply interaction are at the core of Flatland's design. Ideally, one would like a

system in which the general freeform marking of the whiteboard is smoothly integrated into a set of tools that can understand the semantics of particular tasks.

To retain the simplicity of a whiteboard, in Flatland, the user's input is always freehand strokes on the board with no pull-down menus, buttons, handles and the like. At the simplest level these freehand strokes are inked as they are drawn on to the board. As previously discussed, these strokes are grouped into segments. Flatland supports specific tasks by allowing the user to apply *behaviors* to segments. Behaviors interpret input strokes potentially adding strokes and replacing existing strokes. For example with the map behavior, a single line is replaced by a double line to depict a road. Behaviors, however, do not render the strokes themselves, they just modify the strokes belonging to a segment. The segments then paint the strokes creating a unified appearance for the entire board.

Behaviors are implemented so that the behavior only observes strokes, not lower-level mouse events. Thus, behaviors must wait until a stroke is completed before it interprets the stroke. This design helps provide a unified interface similar to stroking a normal whiteboard as all strokes look the same.

A working behavior is indicated as an animal figure on top of the segment. This design helps maintain an informal feel without menus bars while providing a handle to behavior-specific functions. The metaphor is of an assistant or muse that interprets user input and personifies the behavior.

FIGURE 4. Flatland Behaviors

Sample Behaviors

We have designed and implemented a few behaviors to support typical office whiteboard tasks (see Figure 4). Flatland's design goals of simple, informal interaction extend past the general look-and-feel of the interface into the design of individual behaviors themselves. Since the purpose of the behaviors is to support informal, pre-production tasks, ease-of-use is strongly favored over providing features for producing a detailed artifact. Common themes in designing individual behaviors are:

- There are few explicit commands; but strokes are interpreted on-the-fly.
- Generated output is rendered in a "handdrawn" style.
- Minimal (in any) control widgets are added to the segment.
- Handwriting recognition is generally not used to limit the need for error correction and recovery user interfaces. This design choice limits some potential uses of the system but significantly simplifies user interaction.[1] The one current exception is the calculator behavior which requires recognition to be useful.
- "Infinite" undo-redo supports easy error recovery.

To-Do Lists The to-do behavior manages a series of strokes as a single-column list. The items are not recognized per se, but remain individual inked strokes. A handdrawn checkbox is rendered to the left of each item. Subsequent strokes across the checkbox checks off the item. Strokes across an item removes it from the list. A simple gesture allows users to move an item to a new location in the list. After any change to the list's contents (e.g. add, remove, reorder) the list is reformatted.

2D drawing The 2D drawing behavior is a port of Pegasus[9] to the Flatland architecture. The typical frustration users have drawing illustrations on their whiteboards motivates the inclusion of this behavior. Strokes are neatened to create simple formatted line drawings. To create an efficient and intuitive drawing process, Pegasus offers potential new strokes based on the current structure of the drawing. Without explicit commands, the user can quickly author compelling and useful line drawings.

Map Drawing Another common drawing task is sketching maps for other people. Like the 2D drawing behavior, the map behavior replaces input strokes with improved strokes. Single lines become roads with double lines and open intersections. Again, there are no explicit controls for creating detailed production quality maps to get in the way of quickly sketching sufficient and powerful illustrations.

Calculator In the calculator behavior, strokes are interpreted as columns of numbers to be added or subtracted. Output is rendered in a hand-drawn style. Successive calculations can be appended for further interpretation. Likewise input can be modified at any point to trigger re-interpretation. Instead of supplying a calculator widget with push buttons and a display, this behavior leaves a persistent, editable trail that is more easily shared with others and reused.

Combining Behaviors

The difference between behaviors and traditional applications is more apparent when one combines multiple behaviors over time. For example, starting first with the map

1. We are experimenting with off-line handwriting recognition that makes best guesses at recognizing the content of segments. Recognized keywords at a reasonable level of confidence can be used for later retrieval of the segment.

behavior, a user can sketch out the relevant streets and intersections. After removing the map behavior and applying the 2D drawing behavior, the user can now sketch relevant buildings and other landmarks. Now with no behaviors present, the user can label the map. In the future a writing behavior that cleans up letters might be used (see Figure 5).

FIGURE 5. Combining Behaviors

RETRIEVING SEGMENTS

One obvious limitation of current whiteboards is that once material on the whiteboard is erased it can no longer be recovered. In our design, we sought to enable users to retrieve past whiteboard content without adding to the complexity and overhead of using the whiteboard.

Naming a file and deciding on its location is a common, albeit heavyweight task, too heavyweight for informal interaction with a whiteboard. Simply determining a name for content that is loosely associated with any product or deliverable is difficult. In contrast to production artifacts, whiteboard content is heavily context-dependent (e.g. "the outline I was working from last month," "the diagram that Amy and I worked on a few days ago," "my latest to-do list").

To support lightweight, context-rich storage and retrieval of whiteboard content, Flatland uses the Presto [5] document repository. By default, each segment in Flatland is automatically saved as a Presto document without requiring an explicit action or input from the user. The document is identified by its surrounding context (date, time, color(s), spatial location, active and past behaviors). Other forms of context, such as people present in the office, are possible but not yet implemented.

With saving as an automatic process that doesn't require explicit attention from the user, we still must provide a means for retrieving saved segments. When whiteboard users were asked to describe past segments, as well as strategies for retrieving segments, time and visual recognition were the two most cited cues that would aid them in retrieval [17]. We are currently experimenting with a number of context-based retrieval methods for Flatland segments. Two are described below:

Semantic Time Snapping

Time is a powerful cue in retrieving context-rich information. In [17], most whiteboard users could not say exactly when they had written something on their board, but they had a good idea for a general range in time (e.g. few days ago, sometime last week, couple months ago). To support time-based retrieval in Flatland, users can attach a time slider to any segment. The slider can be used as expected, to change the display backward and forward in time for that segment (see Figure 6). Touching the endpoints of the slider makes the slider jump to the next *interesting* point in that timeline. Interesting points are states prior to long periods of no input, prior to input to another segment, prior to removing that segment, as well as explicit snapshots by the user. These states are automatically tagged and stored via Presto.

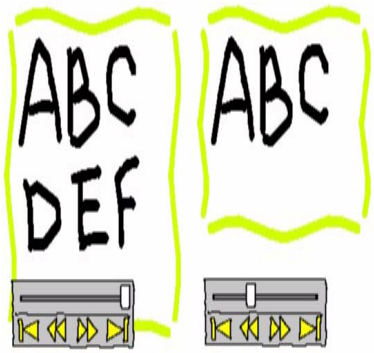

FIGURE 6. Snapping to an Interesting Point in Time

The history mechanism used to implement the time slider also provides infinite undo/redo capability. With a leftward gesture, users can undo strokes in a segment and quickly access a past version. Undo strokes on the root segment *play back* the whole board including the creation and deletion of segments. This history mechanism is based on Timewarp [6].

Context Queries

Visual recognition via thumbnails is another powerful method for retrieving files [17]. The Find behavior will allow users to scan and retrieve past segments. To constrain the search, users select context attributes for a desired segment such as "the map behavior was used," "about last week," and "Ian was in the room," "Icons corresponding to the choices (query terms) are visually depicted in the segment and can be furthered modified (e.g. negated). When the number of matching segments is small (20), thumbnails of the segments are displayed. To retrieve a segment, the user drags it out of the search segment and onto the root segment. This retrieval interface is not fully implemented, but the underlying storage and retrieval mechanisms are in place.

RELATED WORK

Tivoli

One of the major examples of previous work in this area is Tivoli [13][14][14][15], a pen-based interface designed to run on a LiveBoard. Tivoli is principally designed for a specific task—supporting focused meetings about a single issue. It has been most used and studied in the context of supporting PARC intellectual property management.

Although there are many surface similarities, the interaction styles in the two systems are significantly different. As a tool for meetings, Tivoli's interface is geared for sorting, categorizing and annotating whiteboard content. Different types of content such as tables and lists are implicitly and explicitly marked and supported. At any time, however, the strokes on the board are simple strokes that must be

reparsed to support subsequent operations. To support application-specific controls, Tivoli provides "domain objects"—packaged data and controls that can be freely interspersed with freeform ink strokes.

In contrast, Flatland is designed to support ongoing, continuous work across a host of domains, rather than a series of meetings. Content is clustered automatically as the user moves among different segments. In general, the basic interface is much simpler with fewer controls and a smaller gesture set. Further, Flatland engenders a kind of fine-grained persistence, in which the entire history of a user's experience with the whiteboard is captured and available for retrieval and use, as opposed to the potentially more coarse persistence of Tivoli, where, although meeting records may be kept indefinitely, history is largely "chunked" by particular meeting or topic. In contrast to the use of domain objects and continuously reparsed strokes, Flatland provides behaviors that can be associated with any content region building persistent application state for those strokes. This ability to dynamically associate an application's interpretation of content with the representation of that content is much closer to Kramer's notion of Translucent Patches[11].

Translucent Patches and Magic Lenses

Kramer's work [11] identified the importance of separating representation from perceived structure and the interpretation of that structure, especially in the creative design process. Kramer's work focused on the "window system" issues around this free-form association, in the form of "translucent patches." These patches are non-rectangular regions which allow users to associate interpretations of content area with regions on the screen.

Kramer's work applied translucency as a way to preserve context. Arbitrarily-shaped patches were a way to allow spatial multiplexing of interpretations—different nearby regions of the screen could have different interpretations applied to them.

Our work takes a different approach. Our dynamic behavior infrastructure does not require translucency to preserve context, because the user is *always* working in the context of the representation. That is, the user does not need to work "through" the translucent patch to acquire the new interpretation of the content. Second, instead of spatial multiplexing of interpretations, we use temporal multiplexing. In Flatland, a given region of content can have multiple interpretations over the course of its lifetime. These interpretations can be added and removed over time. The focus on temporal multiplexing relieves the user from the tasks of spatial management of interpretations (such as patches) in addition to the spatial management of content. To describe the Flatland model in terms of Translucent Patches, Flatland supports one interpretation layer at a given time, instead of multiple "onion skins" layers at the same time.

Magic Lenses[3], much like Translucent Patches, provide a way to view and manipulate data in different ways. Many of the distinctions between Behaviors and Patches are the same distinctions that can be drawn between Behaviors and Magic Lenses: spatial versus temporal multiplexing, no need for special "see through" presentations, etc.

Further, Lenses provide a way to temporarily modify the presentation or interaction with the data in a non-persistent way. That is, the transformation is in effect during the time when the lens is over the data area. While Flatland behaviors affect input and output only while they are applied, they "annotate" the back-end structure of the data they have been applied to with persistent information. That is, if a set of strokes has ever been interpreted as a map (by having a Map Behavior applied to it), that information is stored in with the strokes in a way specific to the semantics of the Map Behavior. Now, if the Map Behavior is ever removed and then reapplied, those strokes are able to regain their "mapness" as a result.

DynaWall

DynaWall is one or the *roomware* components developed within the i-LAND project [7]. It has an active area of 4.5 meters by 1.1 meters with a resolution of 3027x768. Using multiple computers and projectors, it is clearly designed for a setting quite different than Flatland's intended setting. Its gestures for "throwing, shuffling and taking" whiteboard content are designed for its large size and collaborative use.

Rekimoto

In [21], Reikimoto investigated how to combine multiple pen-based devices. With the Pick-and-Drop interaction, users could move content from one tablet to another. In [22], he demonstrated pick-and-drop and other techniques for using multiple personal tablets with shared whiteboard in a collaborative setting. In the future, we would like to investigate coupling tablets to Flatland as a means to connect Flatland with existing tablet and desktop interfaces as well as to support personal spaces in conjunction with the shared whiteboard space.

Sketching ++

Saund and Moran's perceptual sketch editor (Persketch)[19], Gross and Do's work on an "intelligent cocktail napkin" [8] and Landay and Myer's work on sketching user interfaces (SILK)[12] are good examples of interpreting sketched input while maintaining the look and inherent ambiguity of sketches. SILK recognizes GUI layouts while the Cocktail napkin recognized shapes from a number of domains (furniture layout, circuit diagrams). Persketch extracts perceptual structure out of a collection of strokes. We clearly share the same design ideas of using free strokes to support creative thinking, but our focus is on an environment to support those tasks. Additionally, both SILK and the Cocktail Napkin were tablet interfaces similar to using a untapped. Flatland is designed for long-term use of a whiteboard with heterogeneous content and context-based storage and retrieval of content.

STATUS

The Flatland system has been implemented in Java using JDK1.1.6 and the Swing UI toolkit. The current implementation is approximately 42,000 lines of code. The system uses the Presto document management system as the basis for saving and retrieving "documents" that represent the histories of segments. All of the behaviors described in this paper have been implemented; the Calculator behavior uses the Calligrapher online handwriting recognizer from

Paragraph Corporation; this is the only native code in the system.

CONTRIBUTIONS

This work has investigated approaches to building an augmented whiteboard in the context of continuous, long term office use. In particular, we have pursued three avenues of research based on input from user studies: techniques for the management of space on the board, the ability to flexibly apply behaviors to support varied application semantics, and mechanisms for managing history on the board.

Unlike some previously reported systems, our design choices have been influenced by a desire to support informal work in an office setting, rather than heavy-weight "production" tasks. In particular this focus manifests itself in the four characteristics of use mentioned in the introduction: whiteboards provide interfaces for thinking through pre-production problems; they are useful for organizing and managing "everyday content;" information is implicitly clustered on the board; and office whiteboard use spans the range from private to semi-public. Our designs have attempted to address these characteristics, to make the augmented whiteboard fit easily into existing office work practices.

FUTURE WORK

One obvious area of future work is in the creation of additional behaviors for the Flatland system. There are a number of common tasks in office preproduction work that have been suggested by users and could profitably be supported by behaviors: paper outlining, rough budget analysis, communications, and so on. One of our goals is to evolve the system into a development environment for the creation of light-weight pen-based tools for whiteboard settings.

Although our work has been informed by usage studies of whiteboards in actual offices, we plan to validate our designs via several additional studies: first, an evaluation of the specific UI techniques presented here, and second, an in situ evaluation of the board in its intended setting.

Finally, one of the goals of our work which we have not begun to address yet is the blurring of the physical and the virtual in the office setting. We plan on extending the notions of ubiquitous computing throughout the office, with a particular focus on integrating physical artifacts.

REFERENCES

[1] Abowd, G., Atkeson, C., Feinstein, A., Hmelo, C., Kooper, R., Long, S., Sawhney, N. and Tani, M. Teaching and learning as multimedia authoring: the classroom 200 project. Proceedings of ACM Multimedia '96. New York: ACM.

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

[3] Bier, E.A., Stone, M.C., Pier, K., Buxton, W., and DeRose, T. Toolglass and magic lenses: The see-through interface. In James T. Kajiya, editor, Computer Graphics (SIGGRAPH '93 Proceedings), volume 27.

[4] Communications of the ACM, Special Issue on Augmented Environments, 36 (7), 1993.

[5] Dourish, P. Edwards,W.K., LaMarca, A., Salisbury, M. Uniform document interaction using document properties. Submitted to CHI'99.

[6] Edwards, W.K, Mynatt, E.D. Timewarp: techniques for autonomous collaboration. Proceedings of CHI'97. New York: ACM.

[7] Geissler, J., Shuffle, throw or take it! Working efficiently with an interactive wall. Proceedings of CHI'98. New York: ACM.

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

[9] Igarashi, T., Matsuoka, S., Kawachiya, S., & Tanaka, Hidehiko. Interactive beautification: A technique for rapid geometric design. Proceedings of UIST'97. New York: ACM.

[10] Ishii, Hiroshi & Ulmer, B. Tangible Bits: Towards seamless interfaces between people, bits and atoms. Proceedings of CHI'96. New York: ACM.

[11] Kramer, A. Translucent patches - dissolving windows. Proceedings of UIST '94. New York: ACM.

[12] Landay, J.A. and Myers, B.A. Interactive sketching for the early stages of interface design. Proceedings of CHI'95. New York: ACM.

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

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

[15] Moran, T.P., van Melle, W., & Chiu, P. Tailorable domain objects as meeting tools for an electronic whiteboard. To be published in the Proceedings of CSCW'98. New York: ACM.

[16] Moran, T.P., Chiu, P., & van Melle, W. Spatial interpretation of domain objects integrated into a freeform electronic whiteboard. To be published in the Proceedings of UIST '98. New York: ACM.

[17] Mynatt, E. D. The writing on the wall. Submitted to CHI '99.

[18] Nakagawa, M., Oguni, T., Yoshino, T. Human interface and application on IdeaBoard. Published in Interact '97.

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

[20] Weiser, M. The Computer of the 21st Century. Scientific American 265(3):94-104, 1991.

[21] Rekimoto, J. Pick-and-drop: a direct manipulation technique for multiple computer environments. Proceedings of UIST'97. New York: ACM.

[22] Rekimoto, J. A multiple device approach for supporting whiteboard-based interactions. Proceedings of CHI'98. New York: ACM.