Designing an Augmented Writing Surface

Elizabeth D. Mynatt
Georgia Institute of Technology

Takeo Igarashi
University of Tokyo

W. Keith Edwards
Xerox Palo Alto Research Center

Anthony LaMarca
Yahoo

We designed the Flatland augmented whiteboard interface for informal office work. Our research investigates three different building approaches based on input from user studies.

At Xerox Palo Alto Research Center (PARC), we're investigating strategies for augmenting the *individual* office environment. In making this distinction, we envision the principally private space of an individual employee. Despite experiments with "hoteling" and other depersonalized environments, the use of personal spaces remains the norm for many cultures. Our goal is to blur the boundaries between the physical and virtual realms by augmenting common office tools with computational capabilities. One design goal underlying this approach is to retain the natural affordances of the existing tool, even if this constraint requires limiting the features or complexity of the augmented tool.

We're currently focusing on extending the common office whiteboard—a ubiquitous writing surface typically hung on office walls. We envision replacing the whiteboard with a touch-sensitive surface, "colored" stylus makers, and video projection, but first we wanted to understand the typical uses and affordances of office whiteboards. Although whiteboard and wallboard use have been studied in other settings[1,2] such as meeting rooms, classrooms, and production environments, our initial observations indicated that whiteboard use in the office differed from its use in more public spaces:

- In contrast to meeting room or production environments, office whiteboards tend to be used for a heterogeneous set of tasks in parallel.
- Whiteboard content seems to group in natural clusters or segments. These segments may correspond to different tasks, different people, or writing at different points in time.
- Personal whiteboard content in particular seems heavily context-dependent. Whether it is a seven-digit number or a list of items, the user must remember why and when it was written. Likewise the content may be incomplete, with just enough information present for the user to remember the missing data.

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

Studying whiteboard use

Using observational data and interviews, we collected information about personal whiteboard use in the research lab environment at Xerox PARC. We had a diversity of participants, ranging from senior managers to visiting researchers and support personnel. We took daily photographs of office whiteboards for approximately two weeks. The snapshots provided useful clues in understanding how whiteboard content changes on a day-to-day basis. Although we often found it difficult to completely decipher the tasks behind the content, the snapshots proved incredibly useful for grounding subsequent discussions with study participants. We then interviewed half of the participants, discussing the whiteboard snapshots as well as their ideas for uses of an augmented whiteboard.

Our study confirmed some of our expectations and provided some surprises. A complete account of the study is available elsewhere.[3]

Segments

One of our expectations about whiteboard use was that people typically create and maintain multiple clusters of content on their whiteboard. We certainly saw clusters (or segments) of whiteboard content, typically an average of five segments. When queried, participants rarely identified nested segments.

We were surprised at the longevity of some segments with continued input, for example the common itemized to-do lists where items are checked off and new items added. But many segments stayed in flux as individuals and then multiple people took turns working with the material.

Offline computer recognition of segments would be difficult at best, especially as the board changed over time. In general, handwriting quality was poor, making domain-based recognition difficult. Nevertheless, our inspection of whiteboard snapshots identified segments reasonably easily, as they still tended to maintain an overall gestalt.

Getting white space

A common complaint from most whiteboard users was the continual challenge to find usable space amongst content they didn't want to erase. We identified two strategies in play. First, the clean-desk users often erased their entire whiteboard at the beginning or end of the day. Second, we recognized space scavengers.

Most users were space scavengers. These folks came in two flavors. Many had a known hot spot, where material changed frequently, bordered by longer-lived content. They typically used this hot spot when other people were in the office either for brainstorming with multiple writers or for illustrating points to an office visitor. In any case, content in this area was known to be short-lived. Other users had no obvious hot spot but migrated across the board, erasing where and when possible.

Color

One question we had was whether participants had strategies for using color. If so, knowing those strategies, could we design computerized tools that would more easily recognize segments and tasks? For the most part we found that color choice was random and uninformative. A few users did create on-the-fly color codes when working with more complex material. A larger number of users commented that they would automatically pick up a contrasting color when writing near unrelated material.

Frequency of use

Most whiteboard users described their use as "bursty" (short, intense periods of use with lulls in between). Depending on the circumstances, days to months passed between uses. An open question is, if basic support for whiteboard use improved—such as making it possible to retrieve whiteboard content and thus freeing visual space—how much would whiteboard use increase?

Tasks

We observed a variety of lightweight tasks carried out on whiteboards, including

- Reminders: Either a to-do list or a note on the board designed to prompt a future task. Visibility of reminders is key.
- Quick capture: Especially for users minimizing the amount of paper notes in their office, the whiteboard was a favored medium for capturing quick, specific data such as phone numbers. One surprise was the number of URLs, given how cumbersome they are to write.
- Thinking: All manner of incomplete and seemingly vague content was written as participants used their whiteboards as a scratch surface while pondering concepts much larger than their surface representations. Few illustrations resembled traditional outlines; at best they used a list.

Users asked for capabilities in an augmented whiteboard that would let them retrieve past whiteboard content without the naming and filing overhead associated with desktop computers. They wanted to manipulate a virtual space without losing visibility of existing content. The transformation of material from the whiteboard to desktop computer was often done "in the head," either translating or, less often, transcribing material. Nevertheless, users wanted the ability to view whiteboard content from remote computers.

Augmenting whiteboards

The whiteboard offers a flexible tool for quickly capturing input with an informal look-and-feel, and its large visible surface supports parallel tasks including awareness. However, 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 a to-do list, a series of calculations, or an illustration. Additionally, writing may be illegible and the visual quality of drawings poor.

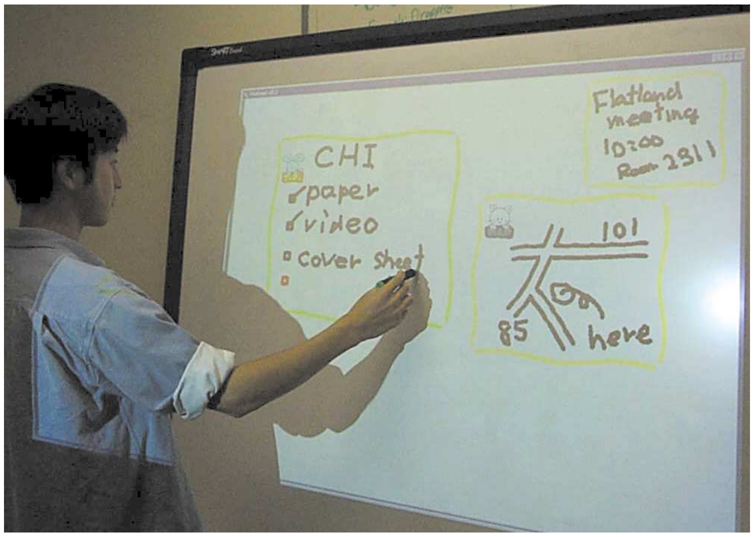
We intend to create an augmented whiteboard called "Flatland" to better support typical whiteboard use in an individual office. In our design, we attempt to extend the existing whiteboard look-and-feel with an interface whose feel and aesthetics match its role in informal office work. Our initial hardware configuration employs a SmartBoard 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 provide a look and feel appropriate for informal whiteboard tasks and distinct from production-oriented tools typically found on a desktop computer.
- To support informal whiteboard 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 informal and context-dependent information. For example, 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 white space for new input while maintaining the visibility of current content.

Flatland basics

Scenario: 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.)

Flatland supports two modes of stylus input. The primary mode accepts drawing strokes on the board. The secondary mode, activated by holding a button, creates meta-strokes. These meta-strokes form gestures used for managing the board's visual layout as well as for applying behaviors to segments. (Although we took great care to design a small gesture set, we won't discuss this process in detail due to space constraints.) The tap gesture causes a pie menu to display, and directional gestures are shortcuts for the pie menu (that is, a marking menu). (See Figure 2.)

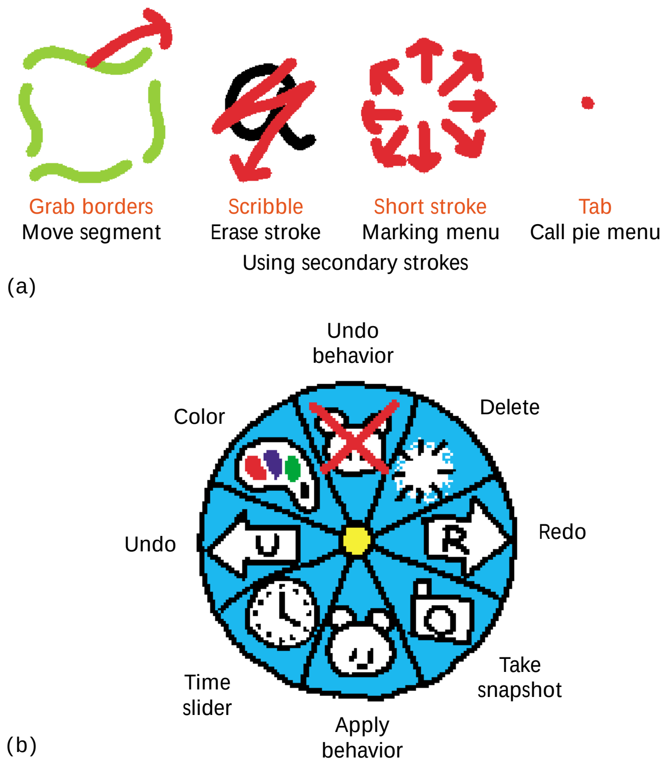


1 **Typical use of Flatland.**

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 let users acquire white space while still ensuring the visibility of existing content.

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



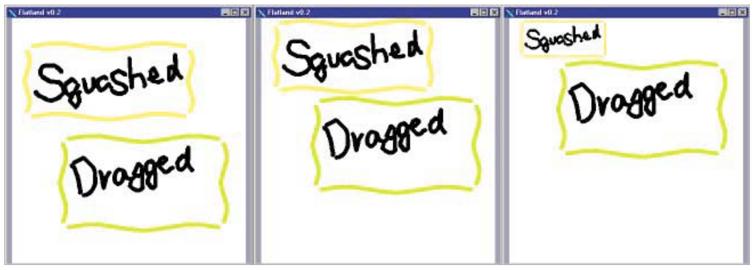
2 **Gestures and pie menus support simple actions.**

Auto-segmenting

Most whiteboard users manage multiple clusters of content on their whiteboards. They take advantage of the large visual surface to use different parts 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 don't 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 clean board, a border appears when the user begins writing, 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 interacted 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.



3 Inactive segments squash to reduce size when they bump into the board's border.

- *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. The bounding box for the active segment expands 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. (Although aesthetically marked as thick, wavy lines, the segments are rectangles, easing coding complexity and performance costs. The discrepancy hasn't been a problem.) 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 because 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 doesn't significantly raise the threshold for initial use, and it provides a base for supporting interaction tailored to natural clusters of content.

Our design differs from other whiteboard interfaces,[4] since users don't have to explicitly group material and clusters aren't recognized by their content (such as recognizing a list or a table). Although the system works in the background, Flatland users feel like they're driving the interaction—the auto-segmenting mechanism is simple, and users 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 looks much brighter than the lighter borders of the inactive segments. In Figure 3, for example, the dragged segments are active, while the squashed segments are inactive. We experimented with a number of approaches in delineating segments, including not showing borders at all and only showing the active segment's border. Our informal use favored showing borders for all segments because they convey a great deal of information about the board's state. 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 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—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 and squashing

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 aren't allowed to overlap. As other segments are bumped, they move out of the way. Many people use their whiteboard as a surrogate memory.[3] If segments were allowed to overlap, important reminders might be completely obscured.

Although we didn't want users to obscure content via overlapping segments, we still needed mechanisms for creating more white space. To meet this need, Flatland automatically squashes segments as they bump into the border of the board. With each bump the segment scales down until it reaches a minimum size (see Figure 3).

Flatland is biased to squashing segments that have been inactive the longest. We opted against using a scrolling or zooming space (a la Pad++[5]) to minimize the potential for losing track of whiteboard clusters. With further user testing, we will determine if users ever need to explicitly squash segments or if the automatic squashing will suffice. To gain more space, users can also explicitly remove segments from the board.

Applying behaviors

One of the primary design philosophies of Flatland is that the whiteboard should be usable like a normal, physical whiteboard and yet provide powerful assistance with everyday tasks as needed. To retain the simplicity of a whiteboard, in Flatland the user's input is always freehand strokes on the board with no pulldown menus, buttons, handles, or the like. At the simplest level these freehand strokes are inked as they are drawn onto 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 transformed into a double line to depict a road. Behaviors, however, don't 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.[6]

Because we implemented behaviors so that they only observe strokes, not lower level mouse events, they must wait until the user completes a stroke before interpreting the stroke. This design helps provide a unified interface similar to stroking a normal whiteboard, as all strokes look the same.

Users apply behaviors by selecting from a set of behavior icons. These animal figures indicate a working behavior at the top of the segment. To dismiss a behavior, the user taps on the animal figure and chooses "Good-bye" from the resulting menu. This design helps maintain an informal feel without menu 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.

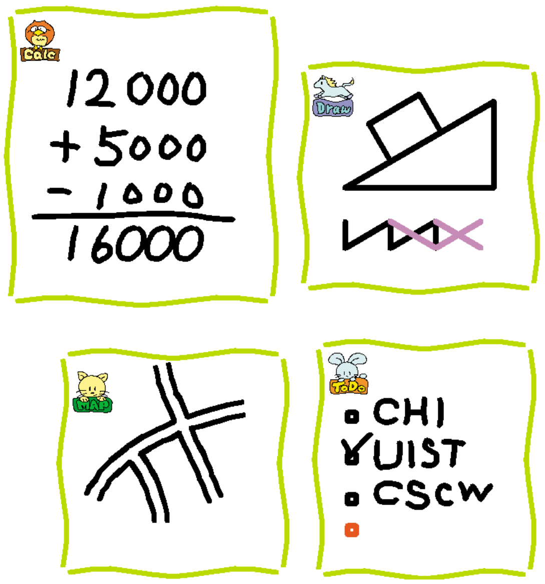
Sample behaviors

We 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, preproduction tasks, we strongly favored ease of use over providing features for producing a detailed artifact. Common themes in designing individual behaviors are

- Few explicit commands exist; strokes are interpreted on-the-fly.
- Generated output is rendered in a "hand-drawn" style.
- Minimal (if any) control widgets are added to the segment.
- "Infinite" undo-redo supports easy error recovery.
- Handwriting recognition is generally not used to limit the need for error correction and recovery. The one current exception is the calculator behavior, which favors handwriting in lieu of push buttons.

This final design choice limits some potential uses of the system, but significantly simplifies user interaction. We're experimenting with offline 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.

To-do lists. The to-do behavior manages a series of strokes as a single-column list. The items aren't recognized per se, but remain individual inked strokes. Flatland renders a hand-drawn checkbox to the left of each item. Subsequent strokes across the checkbox check off the item. Strokes across an item remove it from the list. A simple gesture lets users move an item to a new location in the list. The system reformats the list after any change to the list's contents (such as add, remove, reorder).



4 Flatland behaviors marked with iconic assistants.

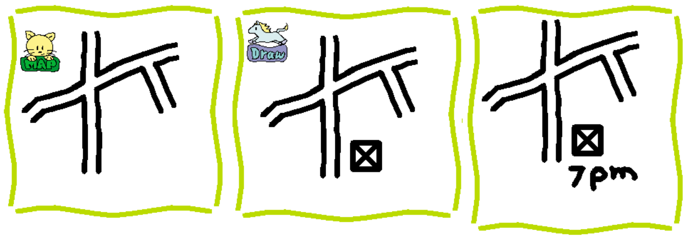
2D drawing. The 2D drawing behavior is a port of Pegasus,[7] an interactive beautifier, to the Flatland architecture. The typical frustration users feel when drawing illustrations on their whiteboards motivated the inclusion of this behavior. Strokes are neatened to create simple, formatted line drawings. To create an efficient and intuitive drawing process, Pegasus offers potential new strokes based on the drawing's current structure. Even 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, the system has no explicit controls for creating detailed, production-quality maps to get in the way of quickly sketching sufficient and powerful illustrations.

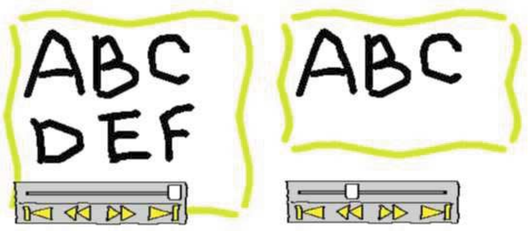
Calculator. In the calculator behavior, strokes are interpreted as columns of numbers to be added or subtracted. Output is rendered in a hand-drawn style. Successive calculations can be appended for further interpretation. Likewise, input can be modified at any point to trigger reinterpretation. Instead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

Combining behaviors

The difference between behaviors and traditional applications becomes more apparent when combining multiple behaviors over time. For example, starting first with the map behavior, a user can sketch out the relevant streets and intersections. After removing the map behavior and applying the 2D drawing behavior, the



5 **Combining behaviors by sketching a map (left), adding a building (middle), and annotating the map (right).**



6 **Snapping to an interesting point in time using the time slider.**

user can sketch relevant buildings and other landmarks. Now, with no behaviors present, the user can label the map (see Figure 5).

Retrieving segments

Current whiteboards have one obvious limitation: once you erase material on the whiteboard, you can no longer recover it. In our design, we wanted 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 loosely associated with any product or deliverable is difficult. In contrast to production artifacts, whiteboard content is heavily context-dependent (for example, "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[8] 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 we asked whiteboard users to describe past segments, as well as strategies for retrieving segments, they cited time and visual recognition as the two cues that would aid them most in retrieval.[3] We're currently experimenting with a number of context-based retrieval methods for Flatland segments, including semantic time snapping and context queries.

Semantic time snapping

Time provides a powerful cue in retrieving context-rich information. In a previous study,[3] most whiteboard users couldn't say exactly when they wrote something on their board, but they had a good idea for a general range in time (a few days ago, sometime last week, a couple of months ago).

To support time-based retrieval in Flatland, users can attach a time slider to any segment. The slider works 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, and explicit snapshots by the user. Presto automatically tags and stores these states.

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,[9] a system to support autonomous collaboration through the use of multiple, editable timelines.

Context queries

Visual recognition via thumbnails offers another powerful method for retrieving files.[3] The find behavior lets users 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 further modified (for example, negated). When the number of matching segments is small (20), thumbnails of the segments appear. To retrieve a segment, the user drags it out of the search segment and onto the root segment. This retrieval interface isn't fully implemented, but the underlying storage and retrieval mechanisms are in place.

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 article 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 and future efforts

One obvious area of future work involves the creation of additional behaviors for the Flatland system. Users have suggested a number of common tasks in office preproduction work that 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 lightweight, 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 goal of our work we haven't yet begun to address 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. ■

Acknowledgments

Numerous groups of people with Xerox PARC have contributed to this work. Thanks to the "Magic Office" team that was the home base for these efforts. Thanks to Mark Weiser and the Computer Systems Laboratory for funding for people and equipment. Also thanks to the Tivoli team for design discussions and an initial gesture code base to get us going. This article is largely derived from a presentation at the 1999 ACM Conference on Human Factors in Computing Systems and published in *Proc. CHI 99*, © 1999 Association for Computing Machinery.[10]

References

1. G. Abowd et al., "Teaching and Learning as Multimedia Authoring: The Classroom 200 Project," *Proc. ACM Multimedia 96,* ACM, New York, 1996, pp. 187-198.
2. T.P. Moran et al., "Evolutionary Engagement in an Ongoing Collaborative Work Process: A Case Study," *Proc. Computer-Supported Cooperative Work (CSCW) 96*, ACM, New York, 1996, pp. 150-159.
3. E.D. Mynatt, "The Writing on the Wall," *Proc. 7th IFIP Conf. on Human-Computer Interaction (Interact 99)*, M.A. Sasse and C. Johnson, eds., IOS Press, Edinburgh, UK, 1999, pp. 196-204.
4. T.P. Moran et al., "Implicit Structures For Pen-Based Systems Within A Freeform Interaction Paradigm," *Proc. CHI 95*, ACM, New York, 1995, pp. 487-494.
5. B.B. Bederson and J.D. Hollan, "Pad++: A Zooming Graphical Interface for Exploring Alternate Interface Physics," *Proc. 1994 ACM Symp. on User Interface Software and Technology (UIST 94)*, ACM Press, New York, 1994, pp. 17-26.
6. T. Igarashi et al., "An Architecture for Pen-based Interaction on Electronic Whiteboards," *Proc. Advanced Visual Interfaces (AVI) 2000*, ACM Press, New York, May 2000.
7. T. Igarashi et al., "Interactive Beautification: A Technique for Rapid Geometric Design," *Proc. UIST 97*, ACM, New York, 1997, pp. 105-114.
8. P. Dourish et al., "Presto: An Experimental Architecture for Fluid Interactive Document Spaces," *ACM Trans. Computer-Human Interaction*, Vol. 6, No. 2, June 1999, pp. 133-161.
9. W.K. Edwards and E.D. Mynatt, "Timewarp: Techniques for Autonomous Collaboration," *Proc. CHI 97*, ACM, New York, 1997, pp. 218-225.
10. E.D. Mynatt et al., "Flatland: New Dimensions in Office Whiteboards," *Proc. CHI 99 Conf. on Human Factors in Computing Systems*, ACM, New York, 1999, pp. 346-353.

Elizabeth D. Mynatt *is an assistant professor in the College of Computing at the Georgia Institute of Technology. There she directs the Everyday Computing Laboratory, examining the human-computer interaction (HCI) implications of the continuous presence of computation in everyday life. She received her PhD in computer science from Georgia Tech in 1995.*

Takeo Igarashi *is a postdoctoral research associate of the University of Tokyo. He earned MS and PhD degrees in information engineering from the University of Tokyo in 1997 and 2000, respectively. He is interested in the development of new interaction techniques for graphical applications, including pen-based 2D drawing, sketch-based 3D modeling, and information visualization.*

Keith Edwards *is a Senior Member of Research Staff at the Xerox PARC Computer Science Lab. His research interests include interactive systems, distributed software systems, and how the two interact. He did his PhD work at Georgia Institute of Technology and is the author of* Core Jini, *published by Prentice-Hall.*

Anthony LaMarca *is a member of the reseach group at Yahoo. He is particularly interested in improving human-computer interaction in everyday applications. He holds a BA in computer science from the University of California at Berkeley and a PhD in Computer Science from the University of Washington.*

Contact Mynatt at the College of Computing, Georgia Institute of Technology, Atlanta, GA 30332-0280, mynatt@cc.gatech.edu.