Support For Multitasking and Background Awareness Using Interactive Peripheral Displays

Blair MacIntyre[1], Elizabeth D. Mynatt[1], Stephen Voida[1], Klaus M. Hansen[2], Joe Tullio[1], Gregory M. Corso[3]

[1]*College of Computing*
GVU Center, Georgia Tech,
Atlanta, GA, 30332-0280
{blair,mynatt,svoida,jtullio}@
cc.gatech.edu

[2]*University of Aarhus*
Aabogade 34A,
8200 Aarhus N
Denmark
marius@daimi.au.dk

[3]*School of Psychology*
GVU Center, Georgia Tech,
Atlanta, GA, 30332-0170
gregory.corso@psych.gatech.edu

Abstract

In this paper, we describe Kimura, an augmented office environment to support common multitasking practices. Previous systems, such as Rooms, limit users by constraining the interaction to the desktop monitor. In Kimura, we leverage interactive projected peripheral displays to support the perusal, manipulation and awareness of background activities. Furthermore, each activity is represented by a montage comprised of images from current and past interaction on the desktop. These montages help remind the user of past actions, and serve as a springboard for ambient context-aware reminders and notifications.

Keywords: Context-aware computing, ubiquitous computing, ambient displays, office computing, Rooms.

1 Introduction

Advances in technological capabilities enable new forms of interaction and often suggest the re-examination of previous interface concepts that could not be fully realized by the technology of the day.

In this research, we take as a starting point the use of interactive, projected displays in individual offices. Often discussed in the context of ubiquitous computing and augmented environments, these displays are envisioned as a natural extension to traditional computing in a work setting. In particular, we are interested in leveraging projected displays as peripheral interfaces that compliment existing focal work areas, and supporting the natural flow of work across these two setting. We are not alone in believing that the effective design of peripheral displays can revolutionize human-computer interfaces; the intuitive appeal of such displays has given rise to an assortment of exciting research that is exploring possible ways to take advantage of people's uncanny ability to utilize peripheral information with comparatively little effort [2][12].

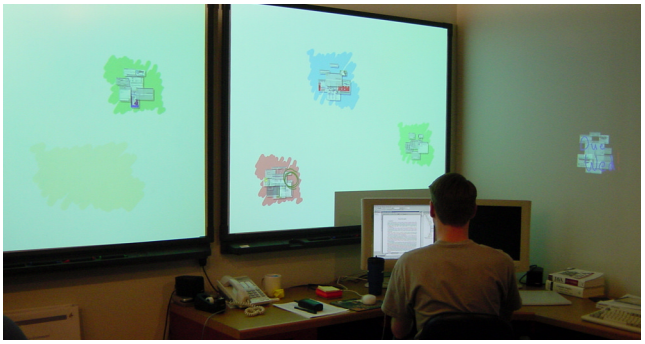

Figure 1: The augmented office environment including the focal and peripheral interactive displays.

We use these peripheral displays to assist users in managing multiple "working contexts"—coherent sets of tasks typically involving the use of multiple documents, tools, and communications with others. This goal of supporting multi-tasking is not new and has received considerable attention in a variety of research communities (e.g., [17]). Awareness of the need to support multiple simultaneous activities drove the development of the multi-windowed graphical user interface [17], and the subsequent addition of multiple "virtual desktops" to these interfaces [7].

Unfortunately, these graphical user-interfaces do not provide effective support for managing multiple working contexts. Limited screen real estate makes it impossible to maintain an awareness of background activities. Moreover, constraining the interaction to the desktop is a poor match for common human behaviors such as using large amounts of physical space to simultaneously organize, monitor, and manage multiple activities [20]. The goal of our research is to leverage large projected interactive surfaces to support innate human abilities such as peripheral awareness, and human cognitive practices such as multi-tasking and off-loading information into the physical environment [10].

Our system, Kimura, separates the user's "desktop" into two parts, the focal display on the desktop monitor, and the peripheral displays projected on the office walls, as shown

© ACM, 2001. This is the author's version of the work. It is posted here by permission of ACM for your personal use. Not for redistribution.

The definitive version is published in *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST '01)*, November 11-14, 2001, Orlando, Florida, USA, pp. 41-50.

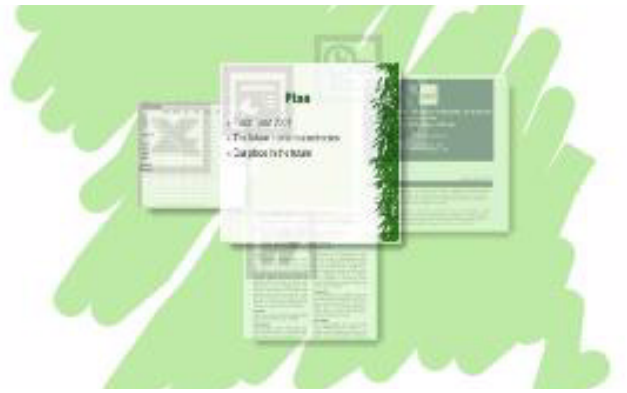

Figure 2: One montage design. Items spiral out from the center based on relative importance.

in Figure 1. As the user shifts between working contexts, background activities are illustrated as visual *montages* on the peripheral display.

From Kimura's point of view, a working context is the cluster of documents related to a general activity, such as managing a project, participating in a conference, or teaching a class, as well as the collection of on-going interactions with people and objects related to that activity. Any cluster can have numerous documents, including text files, web pages, and other application files, that have been used in the course of the activity, plus indications of on-going activity such as email messages without replies and outstanding print jobs. Kimura automatically *tracks the contents of a working context*, tagging documents based on their relative importance. As in previous systems, such as Rooms [7], users demarcate the boundaries of working contexts manually, as this operation is light-weight from the user's perspective and error-prone if left to the system. One contribution of this work is creating and using logs of activity to support both awareness, and resumption, of background tasks.

Background activities (working contexts) are visualized as a *montage* of images garnered from the activity logs. These montages are analogous to the "room overviews" provided by other multi-context window managers, but where these systems show the exact layout of the current windows in each room, our goal is to show visualizations of the past activity in the context. These visualizations help *remind the user of past actions* (see Figure 2); the arrangement and transparency of the component images automatically creates an icon for the working context. Another contribution of this work is the design of these visualizations of past activity.

The montages are displayed on an interactive projected surface, and thus help *support common whiteboard practices* [20]. Users can reposition montages, for example, to indicate the respective priority of background activities, as well as annotate them with informal reminders. Additionally, montages serve as anchors for *background awareness* information that can be gleaned from a context-aware infrastructure. Supporting interaction with the montages, and their integration with background contextual cues, represents another key contribution of this research.

1.1 Paper Overview

In this paper, we first present a scenario that highlights several key interactions with our system. Following this illustration, we discuss the substantial related work in this area, and describe our contributions with respect to these previous efforts. We then describe in more detail the novel aspects of our interface, namely the design of, and interaction with, the montages. Next, we discuss our system architecture, and our design decisions with respect to creating a scalable and flexible context-aware architecture. We close by describing our plans for future research.

1.2 Scenario

As Charlie walks into his office Monday morning, his whiteboard displays multiple montages consisting of documents and other computer images. Glancing at the board, Charlie decides that working on the new budgets can wait until Wednesday and jots a quick reminder on the montage. Next, he decides to start his day by working on his advisory board briefing for next week. As he selects the montage, his desktop reconfigures to contain the applications he left running when he worked on the briefing last Friday, and the montage appears on the wall near his monitors. The Netscape browser still contains the agenda for the meeting, and his initial set of slides are loaded into PowerPoint. He notices that a different Netscape window is prominently displayed in the montage, showing part of a review of last year's briefing that he studied carefully on Friday afternoon. As he works on the slides, he decides to email the laboratory director to ask if he should include some preliminary data in the presentation to answer some of the criticisms in that review. As he sends the message, Charlie wonders when, if ever, he'll get a reply, as the busy director is not known for his timely responses. Charlie works on the slides for another hour and then sends a copy to the printer. Checking the printer queue, he finds that he is behind three large print jobs. Mentally reminding himself to get the printout later in the morning, he decides to shift gears and review his notes before a lunchtime meeting.

As he selects the project montage from his board, the briefing materials disappear from his desktop and the updated montage is now visible on the wall. His recent efforts at writing a project paper are now on his desktop, as well as his notes from the design session last month. As he contemplates his notes, he notices that the face of the laboratory director is visible on the whiteboard, layered on top of the briefing notes. Ah, the director is likely in the coffee area. Charlie intercepts the director and gets the quick answer he needed. As he finishes reviewing the design notes, Charlie realizes that his lunchtime meeting will convene shortly.

Charlie quickly saves his notes and grabs his lunch. Out of the corner of his eye, he notices that the briefing montage has a printer icon overlaid on top of it. The printout! Charlie heads off to retrieve his printout before the meeting.

2 Related Work

This research leverages and extends efforts in many areas of HCI, especially the extensive past work on multiple-workspace window managers (especially Rooms [7]) and the use of projected displays in office settings (especially Flatland [19]). We are also influenced by, and build on, research in context-aware and ubiquitous computing, ambient displays, and activity monitoring.

2.1 Multiple-Workspace Window Managers

It has long been recognized that a fundamental problem of desktop computer systems is the small amount of display real estate available to users. Starting with Rooms [7], and continuing though recent 3D systems, such as the Task Gallery for Windows2000 [25], virtually every window-based desktop computer system has had one or more "virtual desktop managers" to help users manage large numbers of application and document windows in the small space of the desktop monitor. The mismatch between the small amount of screen space and the common "messy desk" work practices people engage in when working with paper is argued eloquently in [7], and their arguments and observations have formed the basis for most of the subsequent virtual desktop managers. Except where other systems differ, we will refer to Rooms in the discussion that follows.

Rooms is based on the observation that, when working on a specific task, users typically interact with a small "working set" of documents and tools. The difficulties of working on multiple tasks cannot be overcome by simply giving the user more desk (screen) space, since some windows are shared between tasks, making it impossible to arrange the windows so that all windows for all tasks will be near each other. Furthermore, it is difficult to navigate efficiently between window groupings in a large flat virtual space without additional navigation metaphors or constraints.

The "rooms" metaphor allows users to collect the windows representing these documents and tools into screen-sized rooms, one for each task, and navigate between the rooms to switch their working set of windows. Rooms, and all subsequent systems, provide a variety of tools for navigating between rooms, obtaining an overview of the space, and sharing windows between one or more rooms (e.g., clocks, system monitors, control panels, etc.). Rooms also allows a shared window to have a different size and configuration in each room, a feature not found in most subsequent systems.

As discussed in Section 1, in our work we extend the notion of "task," as defined in Rooms and subsequent systems, to "activities" that include more than just the documents and application windows currently being used. One implication of this distinction is that we portray past actions, including completed tasks (e.g. working with a now closed application), as part of an activity. Additionally, we move the iconic representation of the activity off the desktop into the physical office (onto the augmented whiteboard). The montages we use as the iconic representation of the activities are designed to convey what was actually being done in the task, not just what windows are currently open. The montages are constructed from images of the most "important" windows, with different measures of importance being possible. Furthermore, we collect additional information about the activities, such as the status of print jobs, email and collaborators, and use this information when generating the montages to support peripheral awareness of the state of the activities as a whole.

We place the activity icons (montages) onto the augmented whiteboard to support awareness of background tasks (see Section 2.2 for a more detailed discussion of our use of the augmented whiteboard). Many of the navigation and interface design issues in Rooms, and subsequent systems, were designed to overcome the fact that only the focal desktop is typically visible. By having representations of all activities continuously visible on a large, interactive surface, we can take advantage of users spatial abilities to organize, monitor and navigate directly to a desired activity.

There have also been attempts at leveraging our 3D abilities within a standard display by replacing the 2D desktop with a 3D world containing 2D documents (e.g., [1][25]). Of these, the Task Gallery [25] has gone the furthest in bringing live 2D applications into a true 3D world. It takes advantage of the input and rendering redirection features of a custom version of Windows2000 to present existing 2D applications in a 3D space. The Task Gallery is effectively a 3D version of Rooms, where the rooms are laid out along a 3D hallway, with the current room on the wall at the end of the hall. While proposing a number of interaction metaphors to support interacting with 2D documents in a 3D environment on a 2D display, the Task Gallery still suffers from many of the same limitations of Rooms, stemming from the lack of screen real estate.

Manufaktur [1] is a 3D collaborative workspace supplement to the traditional 2D desktop, that uses OLE/ActiveX containers to capture images of live applications on the Windows desktop. It focuses on supporting the organization of documents in a shared 3D world, analogous to how designers and architects organize physical artifacts in the real world. Users can select documents and models they are working on for inclusion in the 3D workspace, arrange them according to their working contexts, and reopen them at a later time. However, it is not designed to support multi-tasking activities, being analogous more to a file manager than a task manager.

A number of systems have proposed moving 2D documents off the desktop and into the real 3D physical world via head-mounted displays [5][6] or projectors [21][23]. These systems aim to increase the apparently available screen space, capitalize on people's spatial abilities, and leverage the association of virtual information to physical objects. One limitation of many of these systems is that they do not support people orchestrating their work between areas of focused activity that require high resolution displays, and peripheral areas of information that require minimal attention and interaction.

2.2 Interactive Wall-Sized Displays

This work is also influenced by research in augmented whiteboard interfaces, in particular Flatland [19], as it strove to support the informal work practices for individual offices. Our system is designed to compliment Flatland's interface. Each of our montages is a segment that responds to gestures for moving and annotating the segment. More generally, the whiteboard interface is designed to support casual inspection and organizational activities.

Our work extends previous efforts in whiteboard interfaces by directly connecting interaction on the whiteboard with interaction on the desktop. As an extension of traditional desktop computing, the whiteboard hosts montages that act as links back to previous activities. Additionally the whiteboard serves as the display medium for background awareness cues.

There has been substantial research in augmented whiteboards for conference rooms, including Tivoli [18] and iLand [27]. Some of the interaction techniques for large display surfaces, such as "throwing" in iLand, would be useful in our environment. Likewise the advanced projector display techniques of [21] could enable users to paint their interactive whiteboard on any surface in their office.

2.3 Other Related Work

There has been a large number of systems that attempt to capture information from live desktop systems for a variety of purposes, and while we do not share the same goals as many of these system, we share the engineering concerns. Manufaktur and the Task Gallery, mentioned above, are the closest to our goal of capturing as much information about running applications as possible. Lumiere [9] is closest to our current implementation, which uses Windows system-level hooks to get access to all applications and user activity. Like Lumiere, we use the information to build a model of user activity, although the end applications are far different.

As mentioned in Section 1, we rely on the same human perceptual abilities that motivated much work in ambient and peripheral displays (e.g., the AmbientRoom [12]). Our montages act as peripheral displays that present information about background tasks in a non-obtrusive manner. One novel aspect of our work is the construction of ambient displays from actual images of the user's work, in contrast to using only abstract or iconic imagery. Our montage styles are reminiscent of the "piles" [16] that conveyed the age and type of items in desktop folders.

Our system can also be viewed as a context-aware application; to function, we rely on the continued deployment of a context sensing and aggregation infrastructure such as the Context Toolkit [26]. We do not know of any context-aware applications that combine a detailed model of the user's multi-tasking activity with external context in the way we do here.

3 Interaction Design

Multitasking is a complex, albeit common, human activity. Piles of paper around the periphery of a desk are a physical manifestation of multitasking, indicating a repeated practice of pulling materials into the center for focused work and then collapsing them back into a pile on the periphery when attention is turned elsewhere. Phrases such as "keeping tabs on things" and "juggling as fast as I can" harken to the need to constantly monitor multiple activities.

It is the intent of our design to support these common multitasking practices. Constantly available visual representations of background tasks afford many interactions that support multitasking. The representations are available for perusal, reminders and large-scale organization and prioritization. Moreover the content of the representations serves to remind users of past actions. Finally, new information about a background activity can be attached to these representations, leveraging peripheral awareness capabilities.

In the following sections, we describe our interface design in detail. Although we attempt to be consistent, we use a few terms interchangeably. Notably, in our design, we envision an interactive *wall display.* This large display is created by three projectors that project behind, and to the left and right of, the desktop monitors. Currently the display is made interactive by projecting on SMART Boards™, a commercially available interactive whiteboard. Additionally, our research is influenced by our work on the interactive whiteboard, Flatland. Hence we also refer to our wall displays as whiteboards. The final version of our system will include other whiteboard functionality as found in Flatand and similar systems.

3.1 Basic interaction with the wall display

We envision two types of interaction with the wall display. First, and most importantly, users will treat the wall display as a peripheral interface for keeping track of the existence of, and changes in, background activities. Second, users will directly manipulate the montage images, in conjunction with other whiteboard tasks, while standing at the wall display.

Selecting a montage triggers a task switch. This operation can be performed from the desktop or from the wall display. The contents of the past activity disappear from the desktop and reappear as a montage on the wall display. Simultaneously, the contents of the new task appear on the desktop. The montage for the current task is also displayed near the desktop monitors. This near-periphery display allows the user to remain aware of contextual cues, such as a past browsing activity, that are no longer part of the active desktop. Moreover any additions to the montage, such as annotations (described below), are also available for perusal. Montages retain their position on the wall display so that a background task will return to its prior location unless the user explicitly rearranges the montages.

Montages can be manipulated in the obvious ways on the wall display: moved, deleted and so on. Simple gestures are associated with these common behaviors; montages are segments as in Flatland [19], and therefore react according to a specified behavior when gesturing on them and adjust their size to fit their contents. Currently, the behaviors

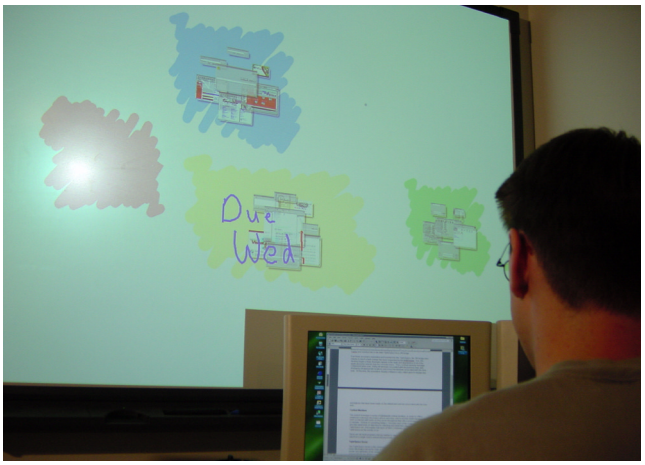

Figure 3: Overview shot of the peripheral wall display and the desktop monitor. Two of the montages include annotations (a red scribble and the blue text "Due Wed").

connected to montages are *moving* when selected, and *annotating* when de-selected.

Annotating montages is an example of an interaction that is well-suited for the wall display: using the dry pens of various colors provided with the SMART Boards, the user may annotate montages with simple ink.

3.2 Visualizing tasks with montages

Montages are peripheral representations of a user's working contexts. As such, they should express the semantics of the working contexts and their relationships in a non-intrusive, albeit suggestive, way. We have explored various visualizations of the information conveyed by montages (see Figures 4-6). In all of the montage prototypes, images from the user's actions in the working context are manipulated to provide a quasi-summary of past activity.

At this point, our designs are based on our own informal understanding of the key characteristics of a working context's history; namely characteristics such as primacy (what consumed most of the user's time), recency (what were the last actions of the user) and inter-relationships (what parts of the tasks are performed in close concert with each other) are highlighted. We combine literal representations of the working context (application snapshots) with various visualization techniques to convey its history at a glance.

For the montages, we have tried to obtain a sketchy look in order to suggest that the information on the wall displays is peripheral to the active working context of the user: montages are shown with sketchy backgrounds in soft colors using a separate color for each montage.

Some visualization techniques are common to all three of our prototype designs. For example, recency is encoded as transparency so that the most recently used documents are the most opaque. We are using five levels of transparency. Another example is our use of watermarks (highly translucent images). In many cases the low-res images of documents are not entirely readable; their visual utility is

Figure 4: Two montages arranged in a spiral based on the decreasing significance of their contents.

Figure 5: Visualization of two montages retaining original spatial layout of documents.

Figure 6: Two montage visualizations based on relative interdependence of documents.

similar to a thumbnail image. Therefore, to enhance the recognizability of the images, we incorporate watermarks of the application icon for major applications.

In Figures 4-6, we demonstrate three major organization schemes for montages.

Spirals of Significance. In the first design, documents are organized according to their overall significance in the task, as the most significant documents should be more easily recognized. As shown in Figure 4, document images are organized in a spiral with the most significant document placed in front and the less significant documents spiraling out in order of decreasing significance. The sizes of the documents also decrease according to their significance. The current significance rating is a measure of how much time was spent on a particular item, weighted by how recently it was used.

Preserving Spatial Relationships. Since the spatial organization of documents on the desktop is often visually salient for recall [10], an iconic rendering of this relationship may be easily recognizable by the user. As shown in Figure 5, document images in the montage are placed akin to where they were on the desktop display, and their sizes are also relatively the same. Additionally, the stacking order of the documents is preserved so that the most recently used document is represented at the front. Montages retain the same aspect ratio as the desktop display (0.75 in this case).

Relative Interdependence Mapping. Complex activities likely include a number of inter-related tasks; as different information sources are used for different purposes, sub-groups emerge in the working context. Likewise documents may have strong similarity ratings due to common content. Exposing these relationships may help characterize the activity, especially activities that are left untouched for long periods of time.

The visualization in Figure 7 tries to take advantage of these relationships by using a modified version of the automatic layout algorithm presented in [14]. The measure of relative interdependence between two documents is currently based on the number of times the user has switched between documents.

The algorithm creates a graph of nodes and edges using a mechanical system simulation: nodes with edges tend to stay together and nodes without edges get repelled. Also, edges may have an intensity, a higher intensity of an edge meaning that nodes connected by the edge will be more attracted.

In our case, nodes are documents and there is an edge between two documents if the user has switched between the documents. The "connectedness" of two documents (the intensity of their edge) is calculated from the probability that the user will switch between the two documents. This measure is calculated using the actual switches a user has made between documents.

In Figure 6, the top left document in the left montage has not been used a lot in connection with the other documents. In the right montage, the two leftmost documents have been used together.

3.3 Background awareness cues

As stated previously, montages serve as anchors for background awareness cues related to a particular working context. Two examples are shown in Figure 7 based on the earlier scenario. When a person who is deemed critical to a background activity becomes available, their face is shown on the montage. In the current system, we notice emails sent to individuals where there has not been a reply. When one of these individuals is available in a public place, such as the coffee room, the montage is adjusted to note their availability. As faces are extremely salient visual cues, our intention is to use them sparingly.

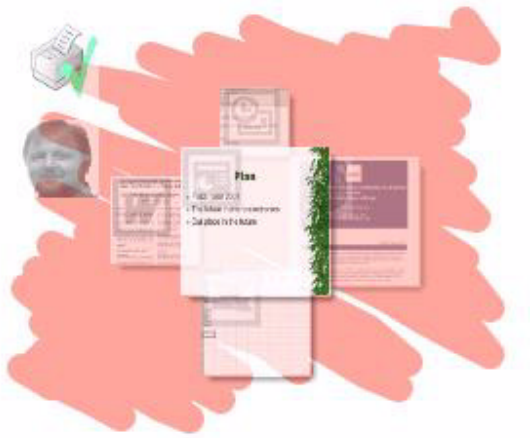

Figure 7: Awareness cues associate with a montage

Another example is the use of tools that are left operating in the background. The status of these jobs, such as a print request, is reflected in the montage. Figure 7 also illustrates a completed print job for a particular activity.

We are currently conducting experiments to determine the relative noticeability of different forms of background cues.

3.4 Working contexts and the desktop

Other multi-desktop systems, such as Rooms, provide a variety of facilities for controlling which applications appear in the different desktops. The architecture of Windows 2000, however, has minimized the need for these facilities in Kimura. First of all, many of the small utility applications that were commonly shared across desktops are integrated into the taskbar, which is automatically shared across all desktops. Perhaps more importantly, if we ignore programs that use the old "multiple document interface" (where the application opens one large window and creates subwindows for each document), the applications themselves generally "do the right thing" when they (or their documents) are opened in the context of a multi-desktop manager.

When an application is running and the user tries to open it, by clicking on its icon or one of its documents, applications that should only have one shared instance, such as messaging applications (e.g., Instant Messenger) or mail readers (e.g., Outlook Express), attempt to activate their window. Applications that should have one window per document (e.g., Word), activate the window for the already opened documents and create new windows for new documents. Some programs, such as web browsers (e.g., Internet Explorer), always create a new window when the user tries to open the application.

In Windows, multi-desktop managers function by using Win32 facilities to hide windows that are not on the current desktop. Since our desktop manager keeps track of the windows that are opened in each desktop, when a hidden window (i.e., one that was created on a different desktop) is activated, our desktop manager reveals it and adds it to the current working context. Therefore, it becomes part of the

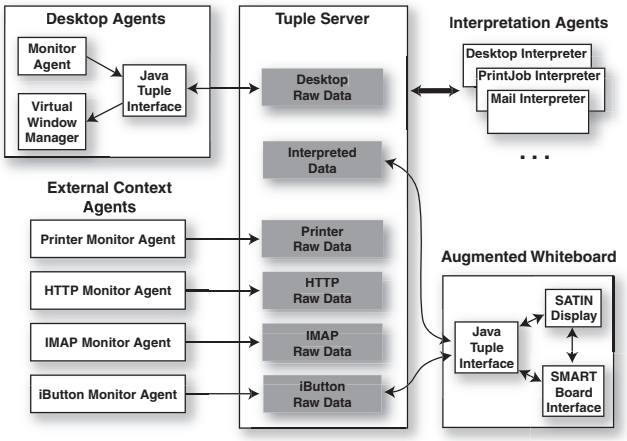

Figure 8: Architecture of Kimura. Arrows indicate primary data flow. The system is designed as a collection of agents communicating via shared tuple spaces.

current desktop, and continues to exist in both desktops. We expect to discover applications that do not behave "correctly," and will introduce additional window management controls as they become necessary to deal with these special cases.

3.5 Inferring working contexts

The problem of inferring a person's working contexts is non-trivial. As a person goes about their daily activities, they interact with a multitude of documents (files, email messages, and web pages) to accomplish each task. We view a working context as a cluster of documents that relate to a particular task or activity. A basic problem that we must address, then, is how to tell which documents are associated with each working context. For example, when a new web page is accessed or document opened, is it part of the current working context, the start of a new working context, or a signal to shift to some other existing working context?

For this stage of our research, we will not attempt to solve this problem. We have chosen to avoid automatic techniques because it is unclear how well they will work, and we do not want the success or failure of these automatic techniques to confound our study of the utility of peripheral information displays. Instead, we will enlist the help of the user by having them identify and explicitly switch between working contexts, using a set of lightweight tools to create, destroy, and manipulate working contexts over time.

4 System Architecture

Kimura's architecture can be broken down into five main components, as shown in Figure 8: desktop monitoring and management agents (for the Windows2000-based focal display), external context monitoring agents, tuplespace-based communication, activity interpretation agents, and the augmented whiteboard.

In general terms, the desktop and external context monitoring agents continuously collect information about the user's activities and store it in the "raw" tuple spaces. The desktop agent also keeps track of which windows belong with which activities, and switches which windows are visible when the user requests a different working context. The activity interpretation agents collect this information and use it to create a representation of the user's activities in the "interpreted" tuple space. The whiteboard process uses this representation to create the montages on the augmented whiteboard display, and supports the interactions with the whiteboard.

4.1 Design Considerations

The architecture is designed to be

- flexible enough for research exploration, and
- practical for real use.

To satisfy the first goal, the majority of the system (aside from some low-level Windows routines) is implemented in Java, and a blackboard architecture is used for communication. The system is designed as a collection of distributed agents communicating via centralized tuple spaces (implemented using Java TSpaces [28]). This approach is well understood, and is also used in systems such as the Open Agent Architecture [3] and Interactive Mural [13]. Tuple spaces are robust in the face of process failure, and allow each agent to be implemented independently.

To ensure the system is practical for real use, we made three design decisions:

- We use the low-level Windows hooks API to monitor and control applications. These hooks work with all applications, although they do not provide information in exactly the form we desire (e.g., it is hard to robustly acquire window images). Over time, we expect to add special handling for some applications (see Section 5).
- We do not change the behavior of the Windows desktop in any substantial way (aside from controlling which windows are visible). This approach contrasts sharply with the Task Gallery, for example, which replaced the Windows desktop with an entirely different metaphor.
- The desktop is controlled asynchronously to the rest of the system. The cost of activity monitoring, data interpretation and display on the augmented whiteboard does not impact the performance of the desktop. Similarly, when the user switches activities, the desktop reacts immediately, regardless of the speed of change on the whiteboard.

In the remainder of this section we will describe the major components of the system, and close by discussing the engineering challenges of creating a system of this sort.

4.2 Desktop Monitoring and Management Agents

Our focal display is a Windows2000-based computer. Win32 "hooks" lets us intercept events, ranging from low-level input to window manager events, on all windows system-wide. A component running on our desktop system uses a DLL that provides callbacks to hooked events

detected by the operating system. The callback information for each hooked event is packaged and sent as a Windows message to the *desktop monitoring and management agent* (written in Java), which stores the information in the *desktop raw data* tuple space.

This agent captures the entire history of the layout of windows on the desktop, and maintains a list of currently open windows for each activity. Each time a window is moved or resized, or the focus changes, the window layout is recorded. Each time a window acquires the focus, a snapshot of the window is taken and stored in a networked filesystem. This strategy ensures we have a relatively up-to-date image of each window, without overloading the system by capturing each window update. Since Windows only lets us capture the visible part of a window, capturing when the window has focus ensures that the window is not obscured.

The desktop agent also watches the tuple space for *SwitchMontage* tuples (see Section 4.4), which signal that the user has switched to a different activity. When this tuple is seen, the windows for the current activity are hidden and those for the requested activity are exposed. The desktop agent also handles exposure of hidden windows (from another activity) when they are activated, as discussed in Section 3.4.

4.3 External Context Monitoring Agents

In addition to monitoring a user's interaction with application windows, we also want to acquire any relevant information to provide a clearer picture of each activity. To illustrate the use of external context, we are monitoring email and web accesses, as well as the status of print jobs. All of the monitoring is currently done without instrumenting specific applications, although this strategy may change over time.

Web access is monitored by an HTTP proxy server (the *web monitoring agent)*. The *printer* and *email monitor agents* run on our Unix mail and print servers. The email monitor agent periodically scan a user's inbox and "sent mail" folders for new messages, correlates them based on message ids, and writes a trail of mail activity into the *IMAP raw data* tuple space. The printer monitor agent watches the print queues for jobs created by the user, and writes status information to the *printer raw data* tuple space.

Kimura assumes it will operate within a more general context system, such as the Context Toolkit[1] [26] or CoolTown [4]. Currently, we use Java i-Buttons [11] to trigger the sorts of external context events that such a system would generate (such as the arrival of a colleague). The iButton events are written into the *iButton raw data* tuple space by the *iButton monitor agent*, and used by various agents for testing and demonstration purposes.

1. We have not yet hooked into the Context Toolkit infrastructure at Georgia Tech, but plan to do so soon.

4.4 Tuplespace-based Communication

As mentioned above, the use of tuple spaces (and other blackboard systems) is common in current distributed interactive systems (e.g., [28]), and offers a number of advantages over connection-oriented event-distribution schemes. These advantages include resistance to isolated process failure, global shared state, and the simplicity of using an unstructured data store. TSpaces also provides persistent tuple spaces, greatly simplifying the debugging of individual agents.

There are two situations that typically cause problems for tuple spaces. First, they have trouble dealing with high-volume updates that need to be distributed with low latency, making them inappropriate for distributing data such as mouse motion events. Second, the performance of the event matching algorithms suffers if a tuple space becomes large. We address the first concern by never sending high frequency data (i.e., we do not capture mouse motion, only actions like button or keyboard presses). We address the second concern by using multiple tuple spaces, as shown in Figure 8. The *raw data* tuple spaces (there are currently five) are used to store the transient data collected by the various monitors. The *interpreted data* tuple space contains the processed data that is used to create the montages.

Data Flow. The data flow is shown by the arrows in Figure 8. Most data flows from the monitor agents, through the interpreter agents, into the interpreted tuple space, and finally into the augmented whiteboard process. The whiteboard process also monitors the iButton raw data space for simulated context tuples, which it uses when generating the montages.

Control data flows in the other direction, from the whiteboard process into the interpreted data space. The whiteboard stores both the montage annotations and *SwitchMontage* tuples (created when the user selects a montage to switch to) in the interpreted space. Any monitor or interpreter agent that cares about activity changes can subscribe to receive *SwitchMontage* tuples. For example, the desktop monitor agent switches the contents of the desktop to the windows for the specified activity when it receives a *SwitchMontage* tuple.

4.5 Context Interpretation

The context interpretation is done by Java agents that collect data from the raw tuple spaces, merge the data into internal activity timelines, and store the information needed by the augmented whiteboard in the interpreted data space.

The principle agent is the *desktop agent*, which extracts a representation of the current document activity from the desktop raw data space. We have implemented two other agents as examples of the potentially useful activities. The *printer agent* extract the status of print jobs in the current activity from the printer raw data space, and creates *printJob* tuples associated with the current montage. The *email agent* extracts from the IMAP raw data space a list of email messages that have been sent during the current activity, for

which replies have not been received, and creates *unrepliedEmail* tuples associated with the current montage.

Even though the current collection of montages only uses a fraction of the data we collect (e.g., we currently use only the last position, size and image of each window, and are ignoring the web access log), the architecture makes it simple for us to experiment with alternative montage styles and content. The interpreters maintain complete internal representations of the merged data, and can access any of the tuple spaces, including the interpreted data space, as desired. Therefore, they can be modified relatively easily to extract the alternate collections of activity information and add it to the interpreted data store.

4.6 Augmented Whiteboard

The augmented whiteboard is implemented as a single process with three main components, all implemented in separate threads: graphical input/output based on SATIN [8], communication with the interpreted and iButton raw data spaces (described in Section 4.4), and communication with multiple SMART Boards.

The whiteboard display class is an instance of a SATIN Sheet: montages are implemented as segments on top of SATIN Patches, annotations are basic SATIN Strokes, and montage image elements are implemented using the SATIN image class. We use standard and custom SATIN interpreters and recognizers to control the montages.

On start up, the whiteboard reads tuples for existing montages from the interpreted data space and creates the initial display. The whiteboard process then subscribes to the interpreted data space for similar tuples, and reflects any tuple space updates on the display. If any *unrepliedEmail* tuples exist for a montage, the process monitors the iButton space for the appearance and disappearance of the recipient of the email, and uses this information as discussed in Section 3.3.

The whiteboard process talks directly to the two SMART Boards on the office wall. It translates tool position messages to the coordinate system of the SATIN window, based on the ID of the SMART Board (provided by the SMART Board API), and sends synthetic mouse events to the SATIN Sheet. Tool change messages (e.g., blue pen picked up) are also sent to the SATIN Sheet and used for actions such as coloring montage annotations.

4.7 Engineering Challenges and Obstacles

Aside from the usual challenges associated with building any complex distributed application (such as communicating huge amounts of image data between components and dealing with replicated distributed state [15]), the most significant engineering challenges are related to monitoring and controlling the activity on the Windows desktop. While the Hooks API allows us to monitor window, mouse and keyboard activity, it does not allow us to see what is going on inside the applications, such as what files are open and what windows are associated with an application. As pointed out in [25], without a standard application interface to inspect the applications, we must resort to dealing with applications on a case-by-base basis. It is even difficult, for example, to sort out splash screens and dialog boxes from content windows.

Another feature we foresee needing in the future is to reopen documents. Assuming we can discover more than just the window images for a selected set of applications (such as the URLs of web page accesses, the folders and message numbers of email messages, and the document details for a few other key applications), there are still a large set of problems that must be dealt with to properly reopen a document. Ensuring that the window is in the correct place, not to mention scrolling the document to the correct location, is not possible in the general case.

5 Conclusions and Future Work

We believe that Kimura is an important step toward unifying focal and peripheral office displays in a way that supports existing work practices. By adding activity montages to an augmented whiteboard, we integrate peripheral awareness of background activities into the existing work area, and allow users to organize and annotate these montages as they would other content on their whiteboard.

While this paper illustrates the key ideas of this project, and Kiruma is currently useful for managing multiple working contexts, there are many interesting research questions left to explore. For example, we would like to integrate some of the ideas in Rekimoto's Time Machine Computing system in the system [22], to allow users to scroll back through time, seeing older montages for one or more activities.

Ideally, if we allow users to scroll through time, we also need to allow users to reopen old documents and applications. This requirement means we must discover more information about each open window, such as the application and document it represents, as discussed in Section 4.7. Our goal is to first deal with common applications (e.g., Microsoft Office, Netscape Navigator, etc.) and add additional application support over time. One key application that we need a richer undertanding of is the email reader. While we currently monitor mail that has been sent and received, we would like to know more, such as which message and folders are commonly accessed in an activity, if there are partially written messages open, and to whom are they addressed.

Another rich area of future work (both engineering and research) lies on the augmented whiteboard. We intend to integrate many of the features of Flatland [19] into our whiteboard, and integrate these features with desktop applications. We would also like to feed information about the user's activity, such as what they are typing, to a recommender system [24], and use some of the space on the whiteboard to display documents relevant to the current activity. In general, we would like to make better use of our actitity logs in support of the current activity, such as displaying a collection of images of unopened documents near the focal display to provide greater context to the user (and eventually give them access to these documents).

Finally, a major focus of our future work will be the design of, and interaction with, the montages. As we gain a better understanding of the working contexts, we will continue to refine what elements are shown and how they are arranged. For example, we can adjust the montage contents based on the length of time since the activity was last conducted to help users reacquire their mental context as their memory of the events fade. We intend to provide much better support for interacting with the annotations, and develop methods of rearranging the annotations as the montages change over time.

Acknowledgments

This work has been supported by the NSF under grant 9988712, as well as an Academic Equipment Grant from Sun Microsystems, and a software donation from Microsoft. We would like to thank all our colleagues and students for their feedback on the ideas in this paper.

6 References

[1] Büscher, M., Mogensen, P., Shapiro, D., and Wagner, I. (1999) "The Manufaktur: Supporting Work Practice in (Landscape) Architecture." In *Proceedings of the The Sixth European Conference on Computer Supported Cooperative Work (ECSCW 99)*, Copenhagen, Denmark, pp 21–40.

[2] Buxton, W. (1995) "Integrating the Periphery and Context: A New Model of Telematics" In *Proceedings of Graphics Interface '95*, pp. 239–245.

[3] Cohen, P. R., Cheyer, A., Wang, M., and Baeg, S. C. (1994) "An open agent architecture", in *AAAI Spring Symposium*, pp. 1–8, Mar. 1994.

[4] CoolTown home page, http://www.cooltown.hp.com/

[5] Feiner, S., MacIntyre, B., Haupt, M., and Solomon, E. (1993) "Windows on the world: 2D windows for 3D augmented reality" In *Proceedings of the ACM UIST '93 Symposium on User Interface Software and Technology*, pages 145–155.

[6] Feiner, S. and Shamash, A. (1991) "Hybrid user interfaces: Breeding virtually bigger interfaces for physically smaller computers" In *Proceedings of the ACM UIST '91 Symposium on User Interface Software and Technology*, pages 9–17, Hilton Head, SC.

[7] Henderson, J.D.A., and Card, S.K. (1986), "Rooms: The Use of Multiple Virtual Workspaces to Reduce Space Contention in Window-based Graphical User Interfaces," *ACM Transactions on Graphics*, Vol. 5, No. 3, July 1986, pp. 211–241.

[8] Hong, J. I. and Landay, J. A. (2000) "SATIN: A Toolkit for Informal Ink-based Applications." In *Proceedings of the ACM UIST 2000 User Interfaces and Software Technology,* San Diego, CA., pp. 63–72.

[9] Horvitz, E., Breese, J., Heckerman, D., Hovel, D., and Rommelse, K. (1998). "The Lumiere Project: Bayesian User Modeling for Inferring the Goals and Needs of Software Users," In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, July 1998.

[10] Hutchins, E. (1995) *Cognition in the Wild*, Cambridge, MA, MIT Press.

[11] iButton home page. http://www.ibutton.com/.

[12] Ishii, H. and B. Ullmer (1997) "Tangible Bits: Towards Seamless Interfaces Between People, Bits and Atoms," In *Proceedings of ACM CHI '97 Conference on Human Factors in Computing Systems*, pp. 234–241, 1997.

[13] Johanson, B., Fox, A., Hanrahan, P., and Winograd, T. (2000) "The Event Heap: An Enabling Infrastructure for Interactive Workspaces", available at http://graphics.stanford.edu/papers/eheap/

[14] László, S.-K. (1994) "Dynamic layout algorithm to display general graphs." In *Heckbert, P.S. (Ed.) Graphics Gems IV*, Academic Press, pp 505–517.

[15] MacIntyre, B. and Feiner, S. (1996) "Language-level support for exploratory programming of distributed virtual environments," In *Proceedings of the ACM UIST '96 Symposium on User Interface Software and Technology*, pages 83–94, Seattle, WA.

[16] Mander, R., Salomon, G., and Wong, Y.Y. (1992) "A 'Pile' Metaphor for Supporting Casual Organization of Information," In *Proceedings of ACM CHI '92 Conference on Human Factors in Computing Systems*, pp. 627–634, 1992.

[17] Miyata, Y., and Norman, D. A. (1986) "Psychological Issues in Support of Multiple Activites," In *User Centered Design*, D. A. Norman and S. W. Draper, eds., Lawrence Erlbaum, NJ, pp. 265–284.

[18] Moran, T., Chiu, P., Harrison, S., Kurtenbach, G., Minneman, S., and van Melle, W. (1996) Evolutionary engagement in an ongoing collaborative work process: A case study. In *Proceedings of CSCW'96*.

[19] Mynatt, E.D., Igarashi, T., Edwards, W.K., and LaMarca, A. (1999) "Flatland: New Dimensions in Office Whiteboards." In *Proceedings of CHI'99*. pp 346–353.

[20] Mynatt, E. D. (1999) "Writing on the Wall," *Proceedings of INTERACT '99*. pp. 196-204.

[21] Raskar, R., Welch, G., Cutts, M., Lake, A., Stesin, L., and Fuchs, H. (1998) "The Office of the Future: A Unified Approach to Image-Based Modeling and Spatially Immersive Displays," In *Computer Graphics (Proc. ACM SIGGRAPH '98)*, Annual Conference Series, pp. 179–188.

[22] Rekimoto, "Time-Machine Computing: A Time-centric Approach for the Information Environment," In *Proceedings of the ACM UIST '99 Symposium on User Interface Software and Technology*, pages .

[23] Rekimoto, J. and Masanori Saitoh, "Augmented Surfaces: A Spatially Continuous Workspace for Hybrid Computing Environments", *Proceedings of CHI'99*, 1999

[24] Rhodes, B.J. (1997) "The Wearable Remembrance Agent: A system for augmented memory," in *Personal Technologies Journal Special Issue on Wearable Computing*, Personal Technologies 1(4), pp. 218-224.

[25] Robertson, George, van Dantzich, Maarten, Robbins, Daniel, Czerwinski, Mary, Hinckley, Ken, Risden, Kirsten, Thiel, David and Gorokhovsky, Vadim. (2000) "The Task Gallery: A 3D Window Manager." In *Proceedings of CHI 2000*, pp 494–501.

[26] Salber, D., Dey, A.K. and Abowd, G.D. (1999) "The Context Toolkit: Aiding the Development of Context-Enabled Applications," To appear in *Proceedings of ACM CHI '99 Conference on Human Factors in Computing Systems*.

[27] Streitz, N.A., Geißler, J., Holmer, T., Konomi, S., Müller-Tomfelde, C., Reischl, W., Rexroth, P., Seitz, P., and Steinmetz, R. (1999) "i-LAND: an interactive landscape for creativity and innovation." In *Proceedings of the CHI '99*. pp 120–127

[28] Wyckoff, P., McLaughry, S. W., Lehman, T. J. and Ford, D. A. (1998) "T Spaces", *IBM Systems Journal,* Vol. 37, No. 3, p. 454