FLUID INTERACTION
FOR HIGH RESOLUTION WALL-SIZE DISPLAYS

A DISSERTATION

SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE

AND THE COMMITTEE ON GRADUATE STUDIES

OF STANFORD UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

François Guimbretière

January 2002

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Terry Winograd (Principal Advisor)

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Pat Hanrahan

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

David Kelley

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Maureen Stone (StoneSoup Consulting)

Approved for the University Committee on Graduate Studies:

Abstract

As computers become more ubiquitous, direct interaction with wall-size, high-resolution displays will become commonplace. The familiar desktop computer interface is ill-suited to the affordances of these screens, such as size, and capacity for using pen or finger as primary input device. Current Graphical User Interfaces (GUIs) do not take into account the cost of reaching for a faraway menu bar, for example, and they rely heavily on the keyboard for rapid interactions. GUIs are extremely powerful, but their interaction style contrasts sharply with the casual interaction style available with traditional wall-size displays such as whiteboards and bulletin boards.

This thesis explores how to bridge the gap between the power provided by current desktop computer interfaces and the fluid use of whiteboards and pin-boards. Based on our observations of fluid expert interactions from everyday life, such as driving a car or playing a violin, we have designed and built a fluid interaction framework which encourages gesture memory, reduces the need for dialog with the user, and provides a scoping mechanism for modes. Together, these features progressively make the cognitive load of using the interface disappear. The user becomes free to focus on other tasks, the same way one can drive a car while conversing with a passenger.

To validate our design, we built the Stanford Interactive Mural, a 9 Mpixel whiteboard-size screen, evaluated the performance of our proposed menu system FlowMenu, and implemented two applications using our framework. The Geometer's Workbench allows one to explore differential geometry; PostBrainstorm is a brainstorm tool that lets users gather and organize sketches, snapshots of physical documents, and a variety of digital documents on the Interactive Mural. PostBrainstorm was tested in brainstorming sessions by professional designers. It demonstrates the feasibility of fluid, transparent interactions for complex, real life applications.

Acknowledgments

I would like to thank the many people who helped me as I did the work described in this dissertation. This work would not have been possible without their support.

It was a real privilege to work with my advisor, Terry Winograd. We had many long and insightful discussions exploring how human computer interactions can be improved. Even though we sometimes disagreed on the specifics, I believe we share the same vision in which humans and computers will be engaged in a synergetic relationship, each leveraging the strength of the other while minimizing the other's weaknesses. Terry was always ready to discuss new ideas and was always very supportive during the four or so years of our collaboration.

I next thank the other members of my committee for their support. Pat Hanrahan brought the vision of ubiquitous, large, high-resolution displays to my attention after his visit to AT&T Research Labs. All through this project, he was an invaluable support for bringing this vision closer to reality. Maureen Stone and I spent endless hours together—first designing, then assembling, then calibrating the version of the Stanford Interactive Mural presented in Chapter 4. Her insight and support during the interactions design phase of what was to become PostBrainstorm were invaluable. David Kelley introduced me to the world of product design, IDEO style. This work would not have been possible without his support. With his help, I was able to observe many brainstorming sessions inside IDEO, and I was able to draw from a population of skilled brainstormers to run more realistic user studies.

Many designers contributed to this project and I would like to thank them here. I would like to thank the numerous designers from IDEO who helped me. It was a great pleasure to interact with them and gather their feedback at different stages of the project. Among them, Chris Kurjan was

always very supportive and went out of her way to help me find my way inside IDEO. David Law, principal at Speck Product Design, broadened my perspective on how brainstorming could be conducted and was adventurous enough to trust one of his design meetings to our system. Phil Weaver and his team from Electronics for Imaging provided us with an early version of the Ebeam system and modified it to better fit our needs. Thanks to Greg Wolff from Ricoh Innovations for giving us an early version of Ricoh's RDC-7 and its control software. Finally the Mural may never have been built without the skills of David Mallard and Adrian James, who built many custom parts for us.

Many fellow graduate students made working in the Graphics Lab a great experience. Among them Tamara Munzner was probably the most influential. Tamara introduced me to the field of information visualization, and our endless nights discussing what a good visualization is all about were key to establishing my research agenda. Her comments on many drafts of my papers and this dissertation were always insightful, helped me improve greatly the quality of my work, and encouraged me to go further. I owe her most of what I know about making an interesting video. Ian Buck, Matthew Eldridge, and Greg Humphreys, all members of the Distributed Rendering Group at Stanford, provided invaluable support and were very patient with my bug reports. I hope that Brad Johanson (and the iRoom team) will forgive me for coming into his office almost every day to try to convince him to implement a feature I needed. James Davis was always there in time of need, and I believe he is the only one to truly understand the video rack. I thank my officemates for accommodating my very special desk management strategy, and members of offices 360 and 376 in the Gates building for letting me invade their offices and babble away about the latest idea I had at the time.

I shall not forget John Gerth, whose technical expertise saved my day many times, or the computer science administrative staff, who made it a snap to order equipment, get reimbursed, and fulfill administrative requirements.

Finally, I owe a lot to my family and my friends, including Corinna Löckenhoff, Elena Krasnoperova, Tamara Munzner, Maureen Stone, James Vera and his wife Jennifer Ochs, and Walt Mann and his wife Becky Jennings who edited this document. Their unconditional support was needed all along the way. Special thanks to Nicolas Scapel who drew the sketches reproduced in Figures 1 and 2.

A mon père

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

Most people, when asked to draw a picture of a computer, will sketch out a display screen and a keyboard/mouse combination for controlling computations (Figure 1). These prevalent user interfaces, based on a windowing system and mouse-driven interaction, have been highly refined to deliver the best out of this configuration. Using a desktop computer, users can perform a huge variety of tasks including editing text, balancing checkbooks, surfing the web, and creating a piece of art.

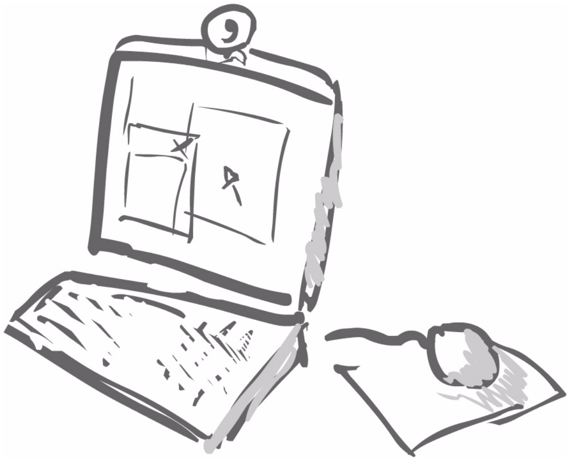


Figure 1: A sketch of a "computer" today. Connected to the screen are a keyboard and mouse used for interacting with the screen. This desktop computer interface has been optimized to exploit this configuration over the past 20 years or so. It is a powerful tool but often demands too much of the user's attention.

As we enter the era of ubiquitous computing, one can expect the question "What does a computer look like?" to elicit a far greater variety of sketches, depending on the context and intended usage. We believe that in the context of office spaces and meeting rooms, the sketch will look more or less like the drawing shown in Figure 2: a wall-size display that one interacts with using an electronic pen. Gone are the keyboard and mouse, yet the user can still perform a wide variety of tasks by interacting directly on the display surface with the pen.

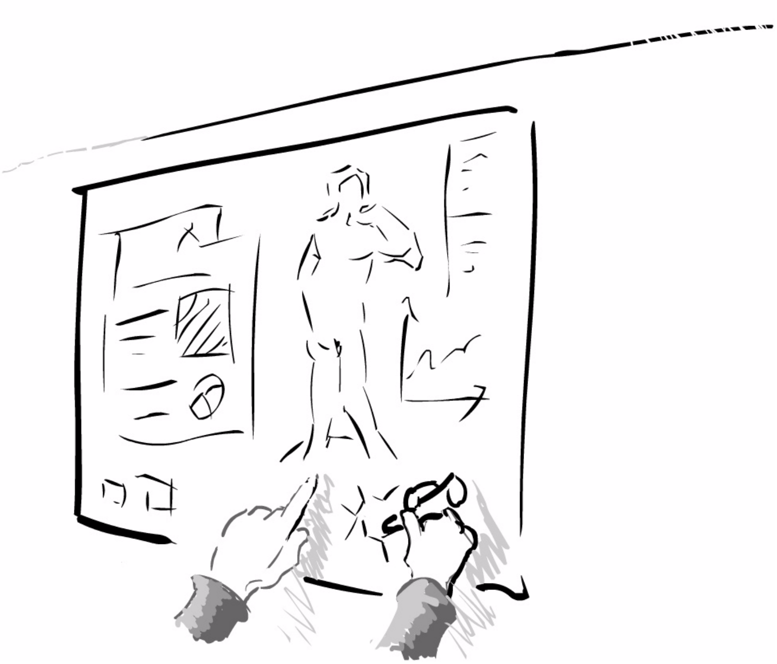


Figure 2: A sketch of a "computer" in the near future. This wall-size display is used without a keyboard or a mouse. Instead, the user interacts with the display via an electronic pen and can perform most of the tasks currently performed on a desktop computer but with the ease of working at a whiteboard.

The visual contrast between the configurations shown in Figures 1 and 2 reflects a contrast of affordance between the two configurations. On one hand, desktop computers are very powerful but require sustained attention to be operated. On the other hand, large display surfaces such as bulletin boards, whiteboards, or simply walls used as a support to create a storyboard, require very little conscious attention to be operated, yet prove to be a powerful tool to structure, display, and manipulate a large quantity of information.

The main motivation behind the work presented in this dissertation is to answer the following question: Is it possible to provide a user interface which combines the ease of use of a board with the computational power provided by a desktop computer? Answering this question has implications beyond the creation of a new digital whiteboard system. Activities such as brainstorming sessions and design meetings that are performed with large boards reflect a pattern of interaction often seen during the design process. Early in the design process, designers often deal with ill-defined, poorly understood problems. During this phase, they will use tools that emphasize flexibility and fluidity (such as sketching) so that they can rapidly explore the solution space and gain a better understanding of the problem at hand and the difficulties ahead [Sch83]. Only when the design is well underway do the designers introduce digital representation. Yet gaining access to computational resources early in the design cycle helps designers explore a larger set of options,

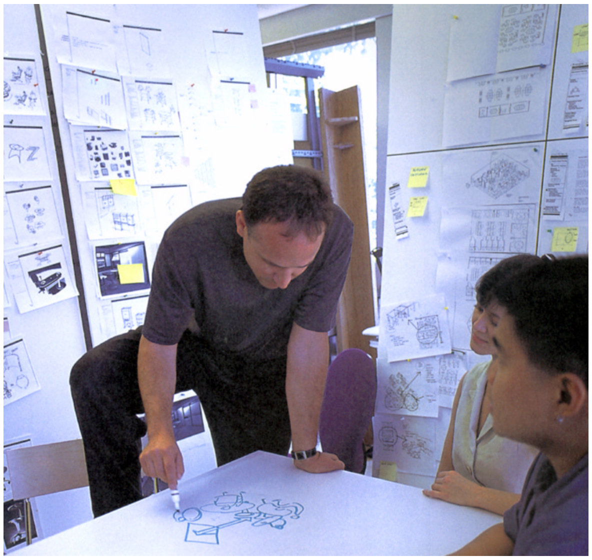


Figure 3: A design studio at IDEO. The picture illustrates the use of physical walls for displaying and working with large quantities of information. Note the diversity of information posted on the wall, from sketches to photographs to Post-It™ notes, and the abundance of printouts of digital media. (Photo courtesy of IDEO)

reuse their work further along in the design process, and document their process more effectively. In other words, mixing the flexibility of early design technique with the computational resources of a desktop computer provides a new tool for *interactive cognition* [Ged98].

Contrasting the patterns of use for current state-of-the-art Graphical User Interfaces (GUI) for desktop computers with the way people use walls to organize and structure information during brainstorming sessions (Figure 3), we concluded that the main limitation of GUIs in the brainstorming environment is that they can severely disrupt users' workflow. This disruption takes place during command selection using a menu bar, during parameter entry (which often requires the user to switch modalities), and because GUIs often rely on temporal modes, which are error prone.

In this dissertation we propose a set of mechanisms that address these disruptions. *FlowMenu* is a new type of marking menu mixing command and direct manipulation to limit the use of temporal modes. Used in conjunction with *Typed Drag and Drop,* an extension of the common drag and drop interaction, FlowMenu lets users build complex, multi-parameter commands on the fly with minimal disruption of their workflow. We will also propose a set of design principles to create *fluid interfaces*, which limit workflow disruptions.

We put these principles into practice by designing two applications for our wall-size interactive surface. The *Geometer's Workbench* users study differential geometry, providing a new graphical

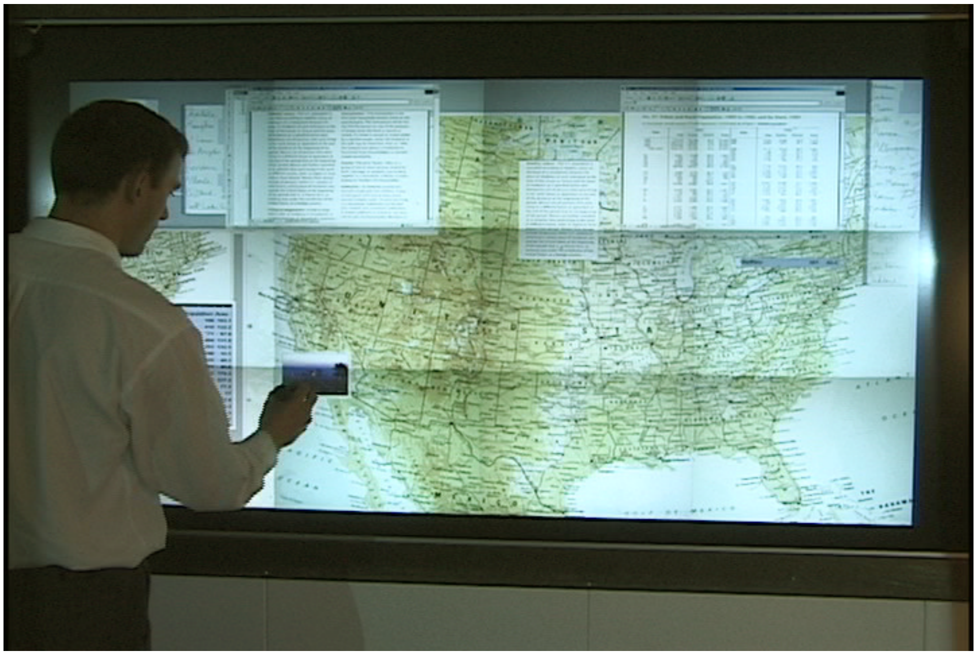


Figure 4: The Stanford Interactive Mural running PostBrainstorm. PostBrainstorm offers tools for gathering, displaying, and managing a wide variety of materials, much as people use walls in project rooms today.

interface to Mathematica [Wol96] a symbolic computation engine. *PostBrainstorm* (Figure 4), a brainstorming tool, lets users gather and manage a wide variety of material the same way people use walls in project rooms today. It also lets people manipulate 3D models, access information on the internet, and archive or exchange the content of a meeting. PostBrainstorm was the main test bed for testing the validity of our design in a real-world application. Using PostBrainstorm we conducted user studies on both low-level interactions such as FlowMenu command selection times, and high-level qualitative assessment of the tool as a substitute for current brainstorming tools.

1.1 Contributions

This dissertation will present the system we built as a demonstration of the feasibility of delivering a fluid user interface for wall-size displays. We needed to address many different design problems while building this system. Some had been previously solved and were just adapted for our purpose; others represent contributions to the field. Our original contributions are:

1. **Interaction techniques for large interactive surfaces.** The main contribution of our work was to develop a set of interaction mechanisms that improve the fluidity of

interactions with wall-size interactive display surfaces. These mechanisms included FlowMenu, a new kind of marking menu; Typed Drag and Drop, an extension of drag and drop that lets users use FlowMenu to specify a semantic type as part of a drag and drop interaction; ZoomScape, a Focus+Context mechanism that lets the user organize a large amount of data on the screen; and MultiPoint, a tool to explore a complex two-dimensional parameter space in the presence of latency.

2. **Stanford Interactive Mural**. To carry out our research we needed a wall-size, high-resolution, seamless, interactive surface. Such surfaces are forecast to become commercially available within ten years or so, and the only substitutes available today (such as [Bar]) were too expensive for us to consider for our project. The display we designed and built measures 6 x 3.5 feet and provides a 64-dpi image, a resolution that we determined sufficient for our research and real-world user testing. The Mural is well adapted for studying close-range interactions and proved to be an order of magnitude less expensive than currently available systems. This work was done in collaboration with Maureen Stone.
3. **MilleFeuille.** MilleFeuille is a user interface toolkit providing the necessary support to implement a fluid interface and take advantage of the high-performance rendering engine driving the Stanford Interactive Mural.
4. **Real-world validation.** We first studied how trained practitioners at IDEO, a Palo Alto-based product design firm, performed brainstorming. We then designed and built PostBrainstorm, our brainstorming application, using MilleFeuille. Finally we ran a user study to validate our design. Furthermore, we conducted controlled experiments to provide a better characterization of the performance of FlowMenu when it is used on a large interactive surface such as the Stanford Interactive Mural.

1.2 Dissertation organization

After this introduction, Chapter 2 presents a description of work previously done in our domain.

Chapter 3 presents the interaction techniques we developed to be used with large interactive surfaces including FlowMenu, Typed Drag and Drop, ZoomScape, and MultiPoint.

Chapter 4 presents our design for a wall-size high-resolution display. Since alignment is paramount for close-up interaction, our solution is based on an accurate mechanism that provides a simple control for each degree of freedom. We also present an alignment protocol that simplifies the alignment process.

Chapter 5 presents MilleFeuille, our toolkit. MilleFeuille was designed to take advantage of the high-powered rendering engine that is connected to the screen. It also provides the support needed to implement the interaction techniques presented in Chapter 3.

Chapter 6 presents the applications we built using our system. The Geometer's Workbench is a new front-end to Mathematica that helps explore differential geometry. PostBrainstorm is a brainstorming tool for use with our display. It allows users to gather a large quantity of information from many sources, organize them on the screen, and generate a post-brainstorm report (hence the name).

Chapter 7 presents the user studies we carried out to understand how people use wall surfaces during brainstorming as well as two quantitative studies. The first quantitative study was designed to measure the command selection times for FlowMenu. The second was a user evaluation of the PostBrainstorm tool.

We finish with our conclusion and directions for future work in Chapter 8.

Chapter 2
Previous Work

Our research explored how to design a user interface that would combine the computational power of today’s desktop computers with the ease of use of a whiteboard. It encompassed many different areas such as building an affordable high-resolution screen, designing a new command mechanism for a pen based system, and understanding user requirements for our target applications. In this chapter, we will present existing works in each of these areas.

2.1 Large wall displays

There have been two main motivations to build large wall displays: visualization of large, complex data sets and as supports for digital whiteboard systems. These two applications have very different requirements. The former focuses on creating high-pixel-count displays designed to be seen from a distance. The latter focuses on direct manipulation on the screen surface but often uses low-pixel-count screens.

2.1.1 Visualization-oriented project

The challenge of building a wall-size display out of commodity parts has attracted the attention of several research groups [FL00]. Their projects have focused primarily on the problem of scalable rendering for large displays [HH99, HBEH00, LCC+00, HEB+01]. In a typical setting, the user drives the display while seated at a workstation some distance from the wall. Because viewers are several feet away from the screen, typical resolutions are 35 dpi or less. In this context, several groups have proposed alignment mechanisms [FL00, HJPS00] and methods to provide automatic geometric and radiometric calibration [RBY+99, Sur99, LCC+00].

High-end commercial systems such as [Bar] have also been available for some time but in limited configurations. They provide very high quality images but are often an order of magnitude more expensive than tiled displays made out of commodity parts. Finally, for a long time, large-wall displays have been assembled using video cubes. They are often used in control centers so that operators can gather contextual information from their workstations.

2.1.2 Digital whiteboard

Since the introduction of the LiveBoard, the first digital whiteboard system [EBG+92], many such systems, including the SMART Board™ [Sma], have become commercially available. A typical system provides a large touch-sensitive screen on which a computer image is projected. Their functionality is identical to that of the desktop machine, using the user's finger as a mouse. Most of these systems provide a tool set that includes different colored pens and a digital eraser in an effort to reproduce the tools and work style of a traditional whiteboard and to extend the basic desktop model to support annotation and a software keyboard.

2.2 Beyond mouse and keyboard

Since its introduction 20 years ago, the GUI, pioneered by the Xerox Star system [JRV+89], has been extremely successful. Using a mouse/keyboard combination as the input device and a bitmap screen for its display, it has proved powerful and versatile. As its underlying mechanisms and limitations have become better understood and new capacities such as two-handed input, the graphic tablet, and faster rendering hardware have been developed, several authors have presented alternatives to address these limitations. With the Cat system [Ras00], Raskin explored how to limit the use of modes in an interface. In an ongoing effort, the “Tangible Bits” project at MIT explores how the interface can be moved from the computer screen to physical objects that people can grasp and manipulate [IU97]. With the See-Through Tool [BSP+93] Bier, et al, explored a new two-handed paradigm for which the non-dominant hand controls a translucent tool palette while the dominant hand performs interactions through the tool palette. Since the advent of notepad-size computers such as the Momenta [Mom91], the Go system [CS91], and the Newton system [New], one of the most active areas of research has been the quest for a new interaction paradigm for direct pen interaction on a screen. Two problems needed to be addressed: how to issue a command and how to enter parameters without a keyboard.

2.2.1 Pen-based command mechanism

For computers using a graphics tablet, menu systems offering some form of context (or pop-up [Tes81]) menu have proved very effective. They enable users to make a menu selection at the point of focus rather than in a distant menu bar, which would require excessive hand movement on the tablet. Pie menus [Hop91] use radial rather than linear selection, which enables the user to learn directional gestures so selection can be made without looking at the menu items. For more fluid interaction, the menu does not even display if a gesture is made immediately upon menu activation. Pie menus can be hierarchical, with the end of one stroke becoming the beginning point for a secondary selection. Users can learn to execute zigzag menu patterns as single gestures that correspond to multi-level selections. Marking menus [Kur93] are an extension of pie menus in which the shape of the path (which is a visible trail on the screen) is interpreted as a selection, if appropriate, or as another gesture. Marking menus can be more flexible since recognition is done based on shape rather than on the precise positioning of the angles in the path [CHWS88]. Marking menus facilitate learning by displaying the underlying menu hierarchy if the user pauses while marking, and they provide a way to merge object selection, tool selection, and direct manipulation [Shn83]. Because marks are normally terminated by a pen-up transition, the discrimination between mark and direct manipulation is sometimes difficult to identify, requiring a time-out [KB91]. The earlier Momenta [Mom91] system provided a similar capability with more limited scope, called the “command compass.” Pook [PLVB00] introduced control menus, an extension of marking menus, to allow the entry of continuous parameters (such as distances and zoom control) associated with a command selection. The system relies on an arbitrary threshold distance to distinguish between marking and direct manipulation.

2.2.2 Parameter manipulation

Besides the obvious software keyboard [SRS+93] and gestures based systems like Unistroke [GR93] or Graffiti [Pal], many designs have been proposed for entering text with a pen device using a circular layout: T-cube [VN94], Quikwriting [Per98], and Cirrin [MA98]. These last two are of particular importance for us since they focus on uninterrupted input of a sequence of characters. Quikwriting was developed to improve text entry on small hand-held devices such as the Palm Pilot [Pal]. It can be seen as a specialized form of marking menu in which the end of a selection gesture is not indicated by lifting the button or pen (as it is for marking menus), but by returning to a home position. The user never has to remove the pen from the surface in entering a series of characters, so common sequences become learned as single complex gestures. Cirrin is a soft keyboard in which letters are arranged at the circumference of a circle. Like Quikwriting, it provides a way to enter successive letters of a word in a continuous stroke without having to lift the pen. After an initial training period, words can be remembered as a kind of shorthand.

2.3 Applications

Whiteboards serve many purposes in our everyday lives—from a note-taking device in an office to a collaboration medium used during a meeting to a simple menu board on a cafeteria wall. For our research, we focused on two particular settings. In the first test application, one can study differential geometry on the Stanford Interactive Mural to understand how a curve is mapped onto a parametric surface. In the second setting, a small group of people carry out a brainstorming session using the Mural to collect and organize ideas as the session progresses.

2.3.1 Differential geometry

For a long time computers have been used as a symbolic algebra tool. Examples include MACSYMA [Bog86], ScratchPad/AXIOM [Jen84], Maple [MGH+97] and Mathematica [Wol96]. These were designed principally for the manipulation of algebraic structures presented in linear textual form. Later systems such MathCAD [Mat], Theorist [The], and GraphingCalculator [Pac] allow the user to drag expressions around to trigger computation and have allowed the interactive manipulation of blocks of text. One of the earliest applications for graphical interaction with geometrical content was SKETCHPAD [Sut63], in which a user manipulated a light-pen to generate and move line geometry with constraints. More recent examples include VPS [Hof96], Geomview [PLM93], and Oorange [GOP+97]. All of these latter systems represent geometry as 3D graphical structures rather than as formal structures such as those that can be defined in a symbolic algebra system. The Geometer's Sketchpad [JF93] supports the graphical manipulation of geometrical, as opposed to graphical, structures but it is limited to classical Euclidean geometry. A closely related project is the *SmartBoard* project for direct mathematical manipulation [Arv] (note that this is different from the SMART Board™ commercial electronic whiteboard product). The Geometer's Workbench is distinguished by its use of a powerful, existing symbolic engine (Mathematica) and its support for informal sketching and casual board work rather than for the generation or verification of proofs.

2.3.2 Brainstorming

Brainstorming has been the subject of many research efforts [SBF+86, SFB+87, NDV+91, PLD01], but for the purposes of our research we focus here on work exploring the use of digital whiteboards in informal settings and small meetings, such as Colab [SBF+86].

2.3.2.1 Strokes-only systems

Strokes-only systems are modeled closely on analog whiteboards' passive capture of strokes. They typically are characterized by a low-resolution display and limited graphic performance. Tivoli running on a LiveBoard [EBG+92] and its successors [PMMH93, MCM97] focused on the automatic recovery of the information present with its layout intact. FlatLand [MIEL99, MIEL00] is a more recent digital whiteboard application that supports small everyday tasks in a casual setting. To help in everyday tasks, FlatLand provides behaviors that can be automatically applied to a set of strokes. Behaviors include a map beautifier, a list manager, and a calculator. FlatLand also provides a powerful transaction-based undo system.

2.3.2.2 Mixing digital and non-digital content

While Tivoli and FlatLand provide very good support for stroke-based annotations, they often lack the flexibility offered by whiteboards to gather information from a wide variety of sources. Early work in that direction includes the DigitalDesk [Wel93] at EuroParc. DigitalDesk was the first attempt to seamlessly integrate digital and analog information. The system let the user scan documents or augment documents with digital content. Related to this approach is the work of Rekimoto [RS99], who uses a combination of 2D tags and pan-tilt camera to provide an integration of the physical and digital worlds. More recently, some projects have approached this problem by providing easy ways to catalog and organize information that is in physical form. For example, the Zombie board [Sau99] and Collaborage [MSM+99] use cameras to scan marks written on a whiteboard and can respond to commands that users give by drawing recognized symbols. The Insight Lab [LJM98] has combined large displays with the capability to identify paper materials by their bar codes and use the codes to organize them. Unfortunately In both cases the display remains passive. Others have focused on capturing the structure of information presented on the board. Using a camera the Designers' Outpost [KNF+01] identified Post-It™ notes on the display surface and recorded arcs drawn between them. Information drawn on each Post-It™ is scanned by a front camera.

2.3.2.3 Space management

Given the size of current digital whiteboard displays, meeting-support software must address space management on the screen. Tivoli [MCM97] provided the user with a tool to zoom in and out on content. FlatLand [MIEL99] automatically shrinks content when it is squeezed against a border of the screen.

Of course, presentation of large data sets on small screens has been explored in depth for information visualization purposes. Pad++ [BH94] provides global zooming with semantic zoom so that the user can view the data set at increasing levels of detail. Although well suited for small displays, the global zoom metaphor can be very distracting for large displays, creating artifacts and preventing users from managing regions of the screen independently. Another solution is the common Focus+Context metaphor, exemplified by the perspective wall [MRC91]. On one hand, Focus+Context techniques appealed to us since they closely relate to behaviors people use with traditional whiteboards. On the other hand they amplify distortion, which can be problematic during meetings.

Chapter 3
Fluid interaction on large displays

As we move into the era of ubiquitous computing, large display surfaces will become more and more common: wall-size digital displays will progressively replace the traditional whiteboards and bulletin boards now found in every conference room and provide most of the functionality of today's desktop computers. In designing a user interface for this large interactive surface one faces the following challenge: how one can bring together the scale and ease of a whiteboard with the powerful organizational and computational environment provided by today's desktop computers?

There are obviously many ways to tackle this problem. Our approach was inspired by our observations of designers interacting with large display surfaces in a project room (Figure 3, page 3). We focused on developing an interaction style that would be as fluid as interacting with a traditional whiteboard: using a pen tool and manipulating virtual sheets of paper, the user can place a wide variety of documents and drawings on the display, similar to what designers do when they pin sketches and lists onto large pieces of foamcore. But unlike foamcore, digital displays let users undo their work, send it to a friend, manipulate 3D models, and access computational resources.

To make this vision come true, three major problems needed to be solved: First we needed a command mechanism that would be well adapted to the characteristics and uses of such a display. In particular, since such displays are often used in meetings where users' attention is in high demand, this command mechanism should require as little attention as possible from its users. Second, to avoid the space crunch that often limits the usefulness of current digital whiteboards, we needed to provide a space-management mechanism so that the amount of information gathered during a typical meeting would not overflow the board. Finally, as the tasks performed by the user became increasingly complex, we had to find solutions that minimized latency while providing a smooth interaction experience.

In addressing these three problems, our goals (somewhat similar to the T3 system [KFBB97]) were to minimize workflow disruption introduced by the interface and maximize the amount of screen used to display information managed by the user—to recreate the look and feel of a non-digital wall-size board. In this chapter we will identify the major causes of workflow disruption and visual clutter in today's GUI and propose a set of design principles to build *fluid interfaces* that improve workflow and require a very small visual footprint. We will then present a solution that follows these design principles for each of the three problems presented above

We will introduce our fluid command mechanism *FlowMenu* [GT00], which when extended to *Typed Drag and Drop* provides a versatile and powerful command mechanism for our interactive surface. Then we will show how *ZoomScape,* a location-dependent zooming interface, can address the real estate problem with very little workflow disruption. All these techniques will be presented in the context of PostBrainstorm, a brainstorming tool running on the Stanford Interactive Mural. Using PostBrainstorm, users can create and manipulate small text fragments written on digital writing paper. Each fragment is created by simply writing on the screen. As strokes are added to each fragment, the system automatically performs an opportunistic handwriting recognition so that fragments on the board may be searched or used as command parameters. Once created, they can be moved, erased, scaled and selected using a lasso. They can also be assigned attributes. A full description of PostBrainstorm can be found in Section 6.2, page 77.

Finally we will illustrate how latency can be dealt with using the advantages of a large, high-resolution display surface using MultiPoint. MultiPoint illustrates how one can explore a simulation parameter space in the presence of latency. It was developed as part of the Geometer's Workbench, a new front end for Mathematica running on the Stanford Interactive Mural. A full description of the Geometer's Workbench can be found in Section 6.1, page 73.

3.1 Fluid Interface

Webster's dictionary defines fluid as "characterized by or employing a smooth easy style." This is a good characterization of the way people use bulletin boards in everyday life. Using a bulletin board requires very limited disruption of their workflow, so the user can focus on the task at hand.

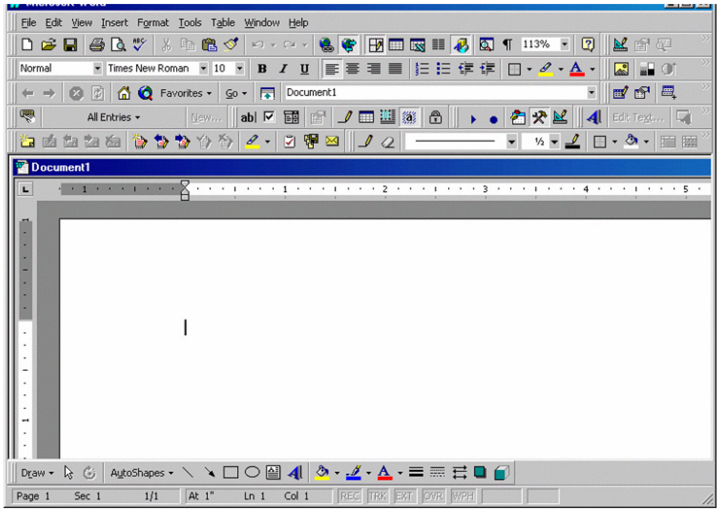


Figure 5: Overuse of decoration by GUIs. The overuse of the tool bar can leave the user with very little space to manipulate her own data. In this screenshot only one third of the screen surface is available for user data.

The board is passive and lets users manipulate data on it as they please, adding little clutter (only small pins or tacks) to the pieces of information posted on the board.

Looking at patterns of use for a typical GUI interface, one can observe four major causes for workflow disruption:

1. While selecting a command using a menu bar, the user must switch her locus of attention to the menu bar to perform the selection,
2. While providing command parameters she must switch her attention to the structure of the dialog box,
3. While starting a new task, she must make sure that the application is in the correct mode (the correct tool is being used),
4. While starting a new task, she must be sure that the system is done with the previous task and is ready to accept her new inputs.

A related problem of the GUI is that it relies heavily on visual artifacts in an effort to make its rich set of commands directly accessible (Figure 5). These artifacts take up a sizable portion of the screen, leaving little room for the data manipulated by users. In this section we will study each of these problems in turn, identifying key design principles for a fluid interface.

3.1.1 Command selection

For devices without keyboards, such as handheld computers and digital whiteboards, the primary command mechanisms of GUI interfaces are the menus. Depending of the context of use, menus can take several forms such as pull-down menu, pop-up menu, and tearable tool palette. But in all cases, using a menu requires a cognitive involvement that cannot be reduced by training. For example in the case of a pop-up menu, the user must first shift her attention to the specific submenu to be selected. During this interaction constant visual feedback is required.

This contrasts sharply with our everyday experience with tools such as cars or musical instruments, which feel as if they become extensions of our bodies, helping us perform tasks that our limbs cannot. Even though our first experiences with many of these tools were difficult and attention-consuming, after sufficient practice their use requires less and less attention to the point where one can often drive while maintaining a conversation with a passenger, much as one would walk and talk on the seashore with a friend.

Psychologists have studied this phenomenon and more generally how we acquire a given skill (see for example [And94]). During the first stage of skill acquisition, the *cognitive stage*, we use instructions provided to us by an instructor and translate these instructions into the correct behavior. During this stage, performing the task requires a lot of attention. As we practice, we move to the *associative stage,* during which we shift from a declarative to a procedural description of the task. Performing the task becomes increasingly fluid and error free. Finally as we practice even more, we reach the *autonomous stage,* during which we rapidly become faster and faster at performing the skill and our cognitive involvement gradually disappears, letting us focus on other topics.

It is important that mechanisms used for fluid interfaces take into account the different stages of skill acquisition and help users reach an autonomous stage of skill where command selection requires a **reduced cognitive load.** In the case of pen interaction, the Marking menu technique [Kur93] provides a transition from the cognitive stage of discovering the menu to the autonomous stage of making a mark on the screen. FlowMenu [GT00], the main command mechanism of our system, provides a similar function.

3.1.2 Parameter entry

Many commands require the user to enter specified parameters. For example, while enlarging an object to a specified magnification in a drawing program, the user must provide the zoom value to be applied. To gather command parameters, many programs use a modal dialog as a way to structure user interaction. Modal dialogs [App87] were first introduced so that applications could force the user to take notice of an abnormal condition by preventing the user from interacting with the application until she dismissed the dialog. Unfortunately their use for parameter entry can severely disrupt the interaction flow: the user has to switch her focus of attention to the dialog, parse it visually, and provide the required value using the keyboard. Often the information present on the screen cannot be accessed directly because the dialog is modal.

Although they are convenient for the programmer, modal dialogs are not the only way to enter parameters. In the paper on T3 [KFBB97], Kurthenbach points out that StudioPaint (a high-end paint application), was designed without any modal dialogs. The Cat system [Ras00] is an early example of a system that avoids dialog boxes for entering parameters. In the Cat system, the user can compose the parameters needed for a command using the material present on the screen. To limit workflow disruption, a fluid interface must **avoid dialogs** as a parameter-entry mechanism.

3.1.3 Modes

As applications become more and more complicated, direct manipulation tools have been overloaded to provide the needed control. For example, using a tool palette, the same mouse is used to draw a line, move an object, and erase. These overloadings are often referred to as *modes.* There are three types of modes:

- **Spatial mode** describes an overloading that is dependent on the location of the tool. We are very familiar with this kind of mode: a click of the mouse has very different effects depending on the area over which it is performed. On a drawing canvas, the click will create a drop of ink. On the menu bar it will open a menu. On the border of a window it will help us resize the window. Because the locus of attention is often at the location of the mouse, these modes are not prone to error.
- **Temporal mode** describes an overloading that depends of the state of the system. For example using the CapsLock key one can change the keyboard to an all-capital keyboard. Once the key is pressed the system stays in that state until the key is pressed again. Tool palettes are another very common example. After picking a tool in the palette, the mapping between mouse interactions and action performed on the screen stay fixed until another tool is selected. This mode proves error prone since users are required to keep track of the system status. Even if the interface provides the correct feedback about the status such as changing the cursor shape, this feedback often goes undetected because the user’s task, not the feedback mechanism, is the user locus of attention [Ras00, page 40]. In fact, Sellen [SKB92] showed that expert users working on a mode-based system often learn to automatically return the system to a known mode to limit their errors.
- **Quasi mode**, like temporal mode, depends on the state of the system. But unlike a

temporal mode, it relies on kinesthetic feedback to keep the user aware of the current overloading. For example, the Shift key operates in quasi mode: the mode lasts only as long as user presses the key. CapsLock, on the other hand, stays on once the key is pressed (temporal mode). Sellen [SKB92] showed that quasi mode is far less prone to error than temporal mode.

Because large displays are often used in group situations, temporal modes are even more disruptive. In a meeting, interruptions are the rule and it is difficult for the user to remember the state of the system after being interrupted to answer a question or check out a reference. To attain the command complexity required with a limited set of tools and to limit-workflow disruption, a fluid interface should **avoid temporal modes** and rely on spatial and/or quasi modes.

3.1.4 Latency

Latency is probably the most pervasive form of workflow disruption. When a command takes more than a couple of tenths of a second [RCM89, RCM93] the user changes her behavior by slowing down her interaction and sometimes taking time to check that the commands she is issuing have been received correctly. This is especially true for expert users. A simple solution to this problem may be to improve the performance of the underlying hardware, but our everyday experience shows that users push their systems to the limit of their capabilities until the latency between commands become unbearable. These two observations seem to lead to a dead end, but fortunately users are also very good at switching between tasks [CMN83] and will accept latency if the system provides low-fidelity feedback right away and they can switch to another task during the wait. Simple design guidelines such as the use of separate threads for lengthy tasks and disallowing blocking operations inside the interaction loop are key to **developing graceful latency management**. More examples can be found in [Joh00].

3.1.5 Visual impact

Current GUI interfaces use many visual artifacts (such as menu bar, tool palette, scrollbar) to provide functionality that is both readily available and self-disclosing. Unfortunately this approach causes the interface to take up a large portion of the screen and has a distracting visual impact on the user's data presented on the screen. We believe that this is too high a price for the user to pay in our application domain. Instead, like Kurtenbach in T3 [KFBB97], we tried to **avoid visual artifacts** while designing our interactions to provide a look and feel similar to a non-digital board. As in T3, we designed our interface so that, given an introduction of ten minutes or so on the basic capabilities of the system, the user could use the system and start discovering more functions by herself.

3.1.6 A Fluid interface for interactive surfaces

The five principles highlighted in the analysis above:

1. **Reducing cognitive load**,
2. **Avoiding temporal modes**,
3. **Avoiding dialogs**,
4. **Developing graceful latency management**, and
5. **Avoiding visual artifacts**.

were key principles we used in our design of a fluid interface for our interactive surfaces. In our design, the surface only uses spatial modes to separate different application areas. It does not use any temporal modes but relies only on quasi modes using the contact of the pen on the screen as kinesthetic feedback. The surface uses a command mechanism called *FlowMenu* [GT00]*,* which combines command, parameter entry, and direct manipulation at the locus of attention. The user invokes the menu by pressing the command button on the pen, and the scope of its operation is always limited to the length of the stroke drawn by the user after issuing a command.

Like many other pop-up menus, FlowMenu command construction follows the *noun-verb* format [Ras00], but practice showed us that this model proved limited for more complex commands, such as assigning an attribute to an object. Because FlowMenu can mix command selection and direct manipulation, we extended the noun-verb format to *Typed Drag and Drop* (TDD). Using TDD the user can use handwritten material present on the surface, assign it a meaning with the FlowMenu, and in the same interaction drop it on the target material at another location on the screen. TDD proved very effective at eliminating the need for dialog boxes in our interface.

Using FlowMenu, the user can easily combine command selection and direct manipulation in one interaction. This requires graceful latency management in order to be carried out. In applications using FlowMenu, it is critical that lengthy or blocking commands are performed outside the main feedback loop.

Like other pop-up menus, FlowMenu is invisible until invoked, letting the application designer limit the clutter on the screen. This comes at the price of reduced disclosure of the available features, but lets expert users focus on the data they are manipulating.

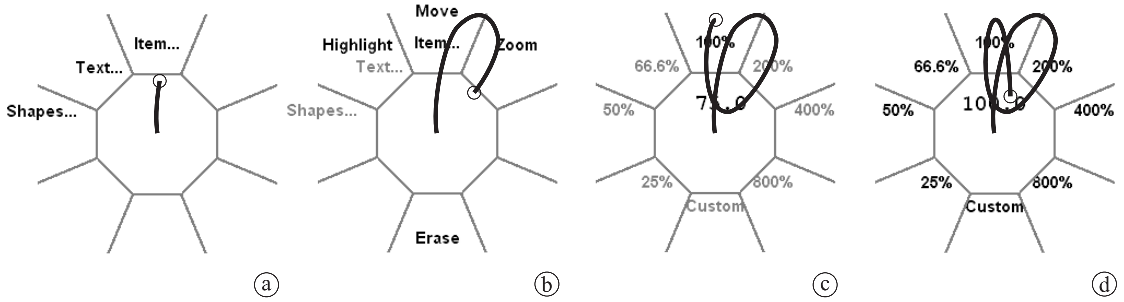


Figure 6: Setting a predefined zoom level with FlowMenu. To zoom, the user moves the pen from the rest area into the *Item...* octant (**a**). Submenus (Highlight, Move, Zoom) appear and the first-level menu items not selected are grayed out (**b**). Entering the Zoom octant submenu, then moving back to the rest area dismisses the root level menu and brings up the zoom menu with the current zoom value (75%) displayed in the center (**c**). A new zoom value of 100% is selected by moving into the octant for the desired value and back to the center at which point the zoom is applied (**d**). Several zoom values can be tried out during the same interaction since the zoom menu stays in place until the pen is lifted. For explanatory purposes, the figures in this chapter explicitly show only the pen track (the underlying selected object is shown only in Figure 8). In normal use, the pen track is not displayed and the selected object is visible behind the transparent menu.

Overall these design principles proved very well suited to the design of a fluid interface for large surfaces used in casual settings such as a meeting, a brainstorming session, or exploratory mathematics on a digital blackboard. We believe that they will be easily adopted for early-phase design activities that rely on rapid, free-form exploration of broadly specified problems. During this phase, rapid exploration of the problem to assess possible solutions and refine the problem statement [Ged98], requires a fluid mode of interaction where the flow of ideas is more important than a precise outcome. It is yet to be seen if the same principles can be applied to simulations, word processing, or e-mail management.

3.2 FlowMenu

FlowMenu is a command-entry system well suited for interactive display surfaces with pen-based input. It provides menu selection, text entry, and parameter adjustment in an integrated mechanism, delivering a smooth and efficient interaction for experienced users while providing a learning path for novice users.

3.2.1 Basic principles

The FlowMenu is presented as a radial menu with eight octants and a central rest area (Figure 6). Starting from the rest area, the user selects a top-level menu item by entering the corresponding octant. As she does, sub-menus for this menu appear laid out further away from the

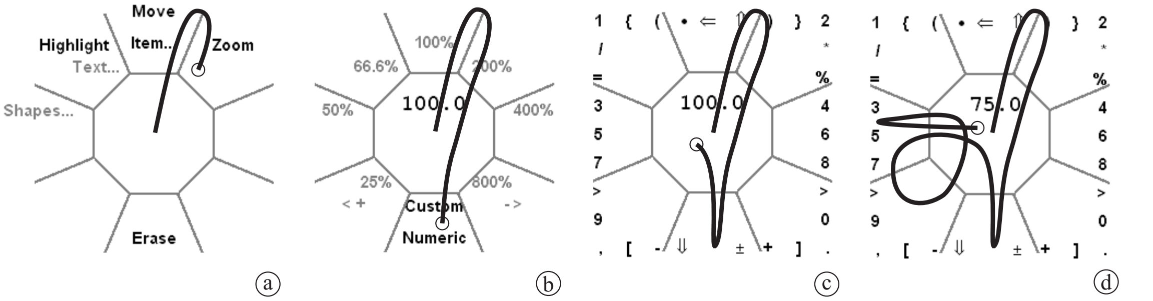


Figure 7: Setting an arbitrary zoom level with FlowMenu. After selecting *Item...* → *Zoom* from the root menu (**a**), the user selects *Numeric...* to enter the new zoom value as a sequence of digits (**b**). The zoom menu is dismissed and the Quikwriting system comes up (**c**) so that she can enter the zoom value (**d**).

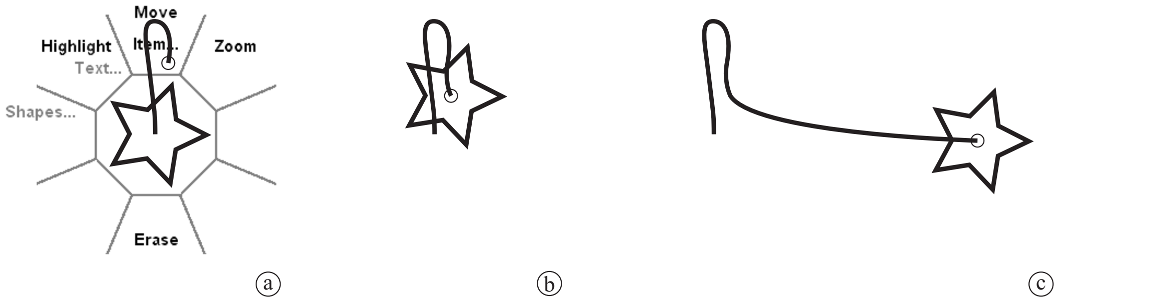


Figure 8: Smoothly integrating FlowMenu interaction and direct manipulation. After selecting the move action from the root menu (**a**), the user continues directly with the drag interaction (**b**,**c**). In contrast to marking menus, the selected object follows the cursor immediately. The initial jump of the object from the center of the menu to the beginning of the drag interaction (**b**) has not been a problem in practice since during a drag, users focus their attention on the target location [Car81].

center while non-selected top-level items are grayed out. Moving the pen to the sub-menu octant and reentering the rest area from this octant triggers menu selection. With a simple FlowMenu, the user can access eight top-level menu items, each with eight submenu items. However, since each selection of a menu item ends with the cursor at the center of the menu, successive menu interactions can be merged together to build deeper hierarchies and arbitrarily long sequences of interactions. Figure 6 shows an example where, after selecting the zoom submenu from the system menu, the system menu disappears and the zoom menu comes up to let the user adjust the zoom.

Merging menu selection and parameter entry is easy because commands are segmented by the return of the cursor to the rest area. To let the user enter an alphanumerical value after a menu selection, we remove the menu from the screen and present in its place a Quikwriting pad [Per98]. Figure 7 shows such an interaction. The selection *Item...* → *Zoom* → *Numeric* brings up the Quikwriting system where the user can enter a numeric zoom value. The user can learn a composite sequence of commands and text as the superposition of simple loop gestures such as that shown in Figure 7d.

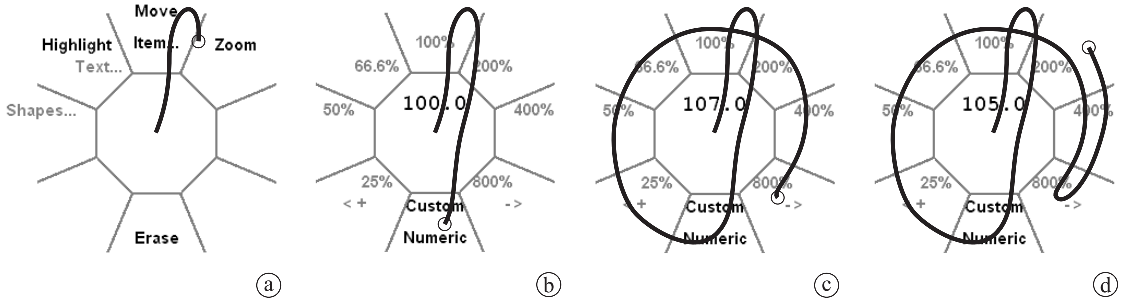


Figure 9: Using the knob interaction to adjust the zoom level. After selecting *Item... → Zoom → Numeric* (**a**,**b**), the user circles the pen around the center area, using the menu as a knob for fine adjustment. Each time an octant line is crossed, the value is incremented by a small amount (**c**) (decremented if counterclockwise (**d**)). The zooming is done in real time, with the object visible (omitted in this figure for clarity) so that visual feedback is provided at all times.

FlowMenu can also be used like Control Menus [PLVB00] by letting the user perform a drag after the action selection, as shown in Figure 8. In that case the return to the rest area obviates the need for an arbitrary threshold distance or time-out to distinguish between command issue and interaction. The menu also provides a “knob” mode in which the user moves the pen around a circle to adjust a setting. As shown in Figure 9, crossing an octant line clockwise increases the value by a small amount. Moving the pen counterclockwise across an octant decreases the value. This kind of interaction is very useful for dynamically fine-tuning parameter values such as zoom level or traveling though a linear history like in an undo log.

Note that unlike marking menus [Kur93], FlowMenu does not delay the appearance of the menu. Given the characteristics of our toolkit (use of transparency, high-speed rendering, and decoupled rendering and interaction loops) we could find no disadvantage to displaying them immediately even when the user is making a coordinated combination gesture. While immediate menu appearance has the potential for visual distraction, our data showed that expert users were not distracted, and that novice users benefited from the absence of a time-out pause.

3.2.2 Managing complex parameters: Typed Drag and Drop

As we developed more and more complex applications it became clear that the simple parameter entry provided by FlowMenu was insufficient: first because the menu required the user to learn the Quikwriting system, secondly because it was poorly suited to multi-parameter entry. One obvious way to tackle this problem was to use a dialog box to let the user enter the needed parameters. But as discussed earlier in this chapter, the use of the dialog box is often very disruptive for the user.

To address this problem we extended the *noun-verb* command construct to the Typed Drag and Drop construct. TDD combines the noun-verb command construct [Ras00] with a drag and drop

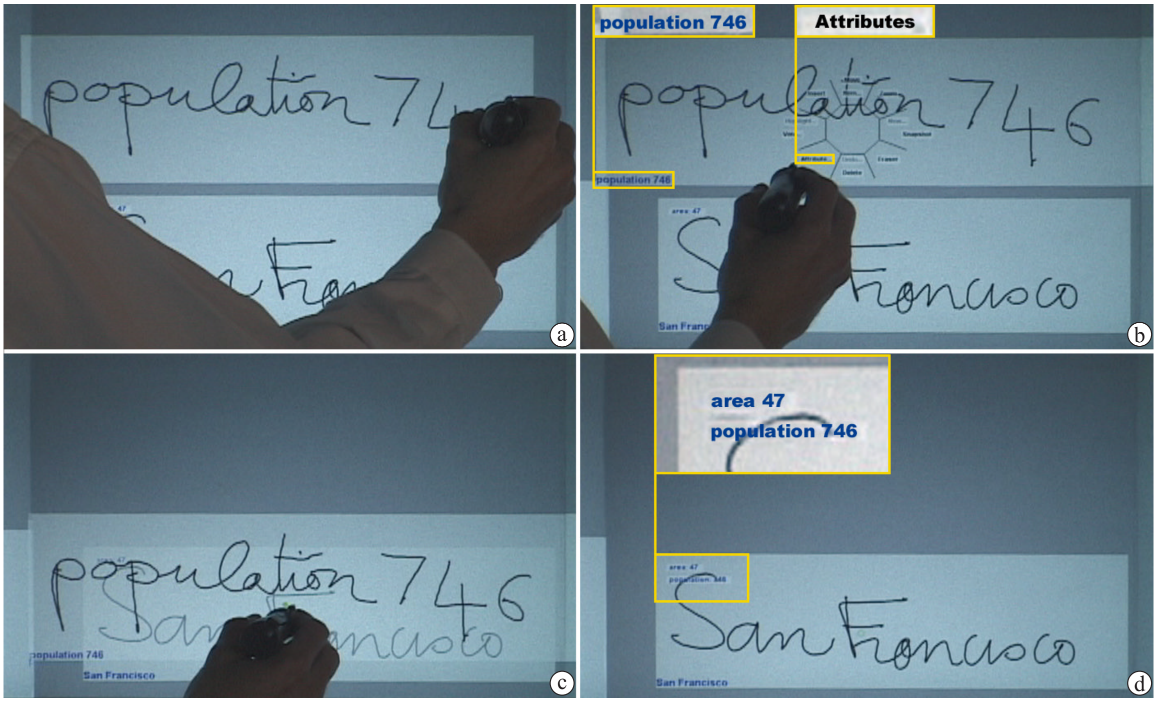


Figure 10: The Typed Drag and Drop interaction. Typed Drag and Drop is used here to assign the Population attribute of San Francisco to 746. After writing on the board "population 746" (**a**), the user uses the FlowMenu to interpret it as an attribute (name, value) pair (**b**), and in the *same* interaction drops the object on top of the San Francisco object (**c**) to update the attribute (**d**).

interaction [App87]. During the first phase of the interaction, a noun-verb construct allows the user to assign a semantic meaning to content on the board using the FlowMenu. Figure 10 shows that, after writing the phrase "population 746" on the board, the user uses FlowMenu to assign to this newly created object the type "attribute." Then, in the same interaction the user drops the object, which is now an attribute, onto the target "San Francisco," which updates its attribute as a result.

TDD lets the user construct complex commands on the fly using the material present on the Mural. Not only can the user assign an attribute to an object, as in the previous example, but she can also load a picture, from a database, or execute a command on a remote computer.

3.2.3 Discussion

FlowMenu shares the advantages of both marking menus and Quikwriting. Noun and verb selection are combined into a single operation, and sequences of verb selections can be combined as a flowing ideogram-like mark. FlowMenu offers a means of learning strokes by providing an underlying self-revealing menu hierarchy to help the user in the transition from recognition to recall. Quikwriting's "return to the central rest area" style of command segmentation provides a smooth way to distinguish between menu selection and direct manipulation interaction. This segmentation makes it possible to integrate alphanumerical text entry and direct manipulation as part of a menu interaction. This segmentation also provides a convenient abort mechanism for the user: at any time during the menu selection the user can abort the interaction by removing the pen from the surface before reentering the rest area.

Visual obstruction of some menu choices by the hand is an issue shared by all radial menus, including FlowMenu. It is a pressing problem for beginners, who sometimes need to move their hand to see menu items, which can be awkward to accomplish while keeping the pen tip on the Mural. Our solution was to avoid using the southeast octant (or southwest octant for left- handed users) or to use it as the opposite octant for complementary items [Hop91]. During user testing we found it helpful to remind the users that since they can abort a selection at any time by removing the pen from the screen, they should feel safe to explore the menu hierarchy. One successful strategy was to move one's hand away from the center so that all first-level menu items became visible and remove the pen from the screen when the exploration is completed. Note that this problem only occurs in direct interaction systems, not with indirect interaction systems (e.g., tablets separated from the display).

Activation of the menu near the borders of the display surface is also a problem for all kinds of pop-up menus. Since our system uses direct interaction on the display surface, it is impossible to use any kind of cursor warping [Hop91]. Given the size of our screen and our particular tracking technology, we could provide a tracking area larger than the display area, allowing the user to complete a gesture even when only part of the menu was visible. This solution does not work for small devices, in which case Kurtenbach's “pull out mark” [Kur93, section 6.2.3] can be used. Our radial menu had fewer problems with expanding near an edge since successive levels of the hierarchy appear at the same place on the screen.

Currently, our FlowMenu implementation uses a simple distance/angle transition detection mechanism. It requires a well-calibrated input stream because the user has to exit and enter the rest area using somewhat narrow corridors. “Eyes free” interactions are limited to simple sequences such as *move* (Figure 8) or *zoom 100%* (Figure 6). A stroke-feature based implementation similar to one described in [Kur93] will provide a more robust recognition, supporting “eyes free” operation for more complex interactions. Even with this limitation, our study results (reported in Chapter 7) show that FlowMenu's performance is similar to that of marking menus of similar complexity.

TDD is a flexible mechanism to construct commands to define and apply attributes without relying on a dialog box. We used it successfully in PostBrainstrom a brainstorming tool described in Section 6.2, page 77, to assign attributes to an object. Toward the end of a brainstorming session,

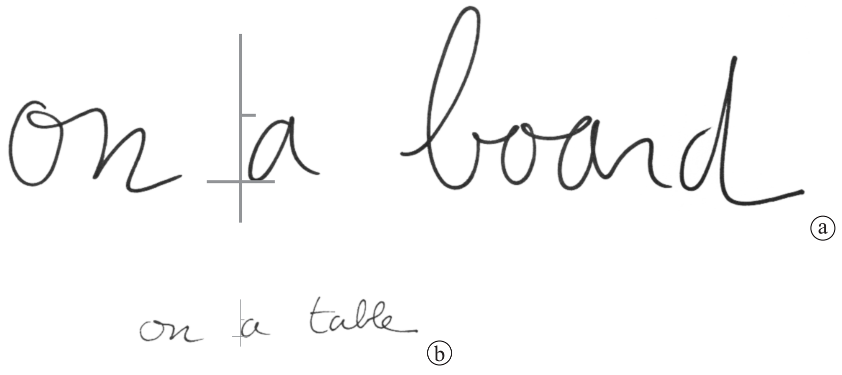


Figure 11: Handwriting on a board. Because users must use their arm to form the letter shapes while writing on a vertical board, they naturally write bigger (around 96 points) (**a**) than when they write on a table (around 24 points) (**b**). The rules indicate height for a lower-case letter, depth for a descender (such as y), and leading, based on 96-point and 24-point Times Roman, respectively.

users used attributes to characterize each idea. Usually the set of attributes and their values is unknown at the beginning of the session so it was impossible to use a menu hierarchy as a way to select an attribute and its value. PostBrainstorm also uses noun-verb and drag and drop constructs extensively in its interface. For example, to reset the bounds of a graph axis one just needs to drop the new value on the current value. Similarly one can enter text into a remote display system window simply by dropping the text onto it. These interactions appear to the user as a simplified version of TDD: in noun-verb construct the user does not need to specify an object, whereas in drag and drop there is no specific meaning assigned.

3.3 Managing information on a wall-size screen

One of the major user complaints about digital whiteboards is the limited amount of information that these can handle compared to a normal wall. To better understand how we could overcome this limitation using our whiteboard-size, high-resolution Mural, we observed users during brainstorming sessions. At the root of this limitation is the discrepancy between the size at which users write characters on the screen and the size at which users are comfortable reading the text they wrote. On a whiteboard, users tend to write at the same size as a typical 96-point font. At the same time, because of the high resolution of the Stanford Interactive Mural, they can easily read the same handwriting scaled down 4 times (to 24 points), a typical size for writing on a table (Figure 11). Using this 25% scaling, it becomes possible to manage more than a hundred short phrases on the screen—an important threshold in practice.

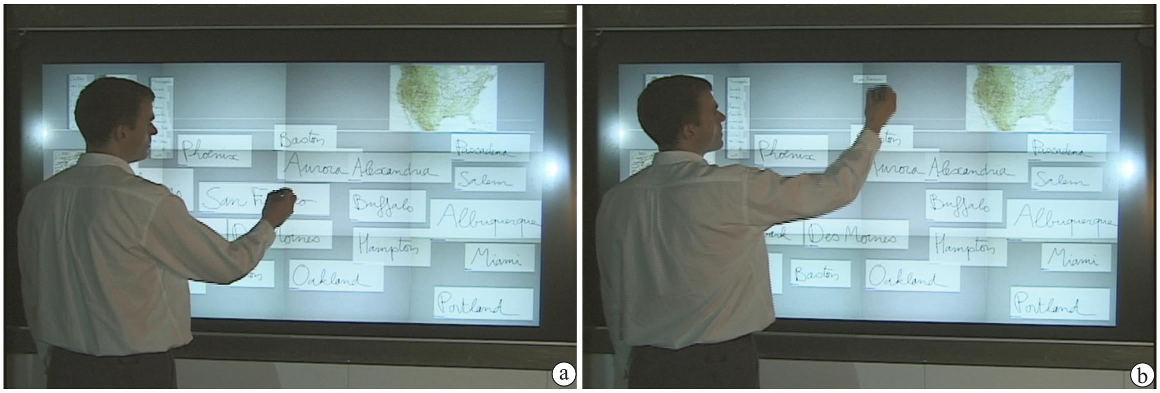


Figure 12: A typical ZoomScape configuration: the top fifth of the screen (context area) scales objects at 25%, whereas the rest of the screen (focus area) scales objects 100%. For smoothness a small transition ramp is provided between the two areas. Moving objects on the surface scales them automatically around the cursor according to the current cursor location. Here the object "San Francisco" is moved from the focus area (**a**) to the context area (**b**)

It is also the case that users enter text and manage content at different locations on the board: writing takes place at the center of the board, the natural focus of attention, whereas information management takes place at the periphery, where contextual information is kept.

To provide a fluid transition between the writing size typically used on a wall-size display and the information management capability of our high-resolution screen, we introduced *ZoomScape,* a position-dependent zooming surface. ZoomScape is similar to zooming techniques such as Pad++ [BH94] and Focus+Context techniques [MRC91] but was designed for manipulation on a large digital display: groups of objects can be manipulated safely and objects are active everywhere on the screen.

3.3.1 ZoomScape

ZoomScape is a location-dependent zooming surface where the scale of an object depends on its location on the screen. It is completely defined by a function $shrink(x, y)$, which returns the scale at which objects should be rendered for each location (x, y) on the screen. A typical ZoomScape configuration for our Mural is shown in Figure 12: on the bottom part of the Mural, objects are displayed at the same size they were entered (100%); on the top of the board they are scaled to 25% of their size. To assure smooth transitions, a small transition area is provided between the two areas. As an object is moved on the ZoomScape surface, its scale is updated dynamically to reflect the current value of the $shrink(x, y)$ function under the cursor.

It is often the case that users need to move a group of objects together. For example to clear the focus area, users will select a group of objects in that area using a lasso and then will move the group to the top of the screen for later reference. In that case, one has to consider not only how to

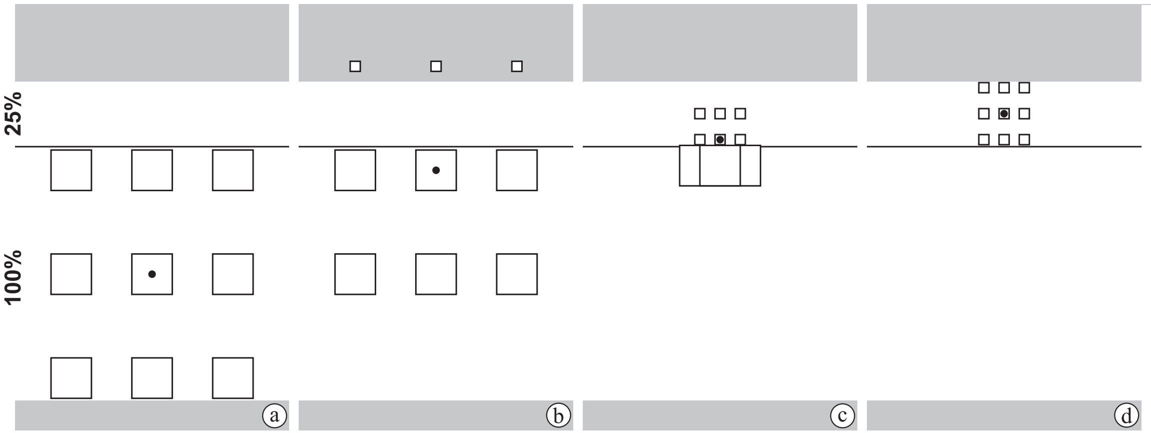


Figure 13: Manipulating a group as a rigid body on a ZoomScape. In this sequence a large group is moved over a ZoomScape that scales objects 100% on the lower part of the screen and 25% on the upper part. In this example, the group geometry is considered as a rigid body, the scale of which is adjusted according to the scale at the cursor location, represented here by a black dot. As the user moves the group upward (**a**), some objects may disappear off the top of the screen (**b**) before reappearing when the cursor reaches the 25% area (**c**, **d**). This solution not only causes distracting visual artifacts, but it can also cause objects to be lost if the interaction is interrupted while some objects are offscreen.

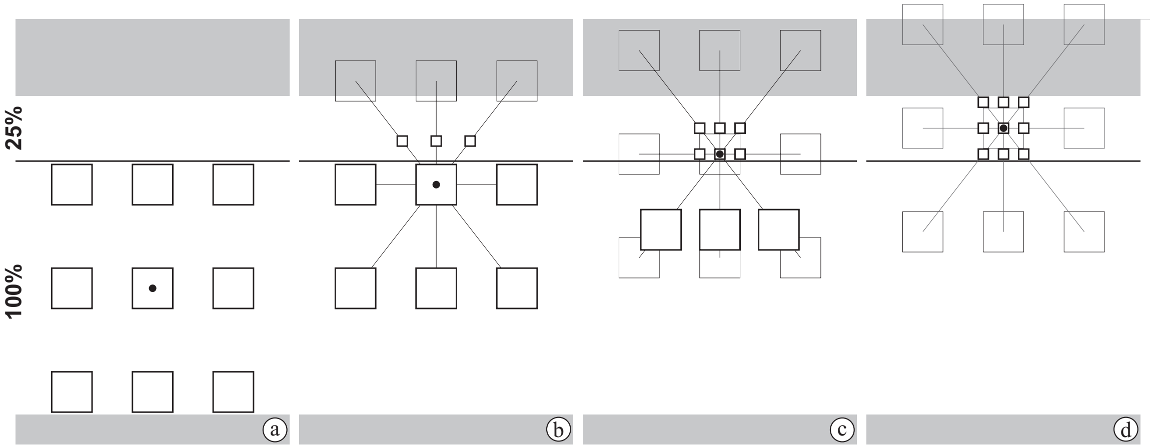


Figure 14: Safely moving a group on a ZoomScape. In our final implementation, each object is scaled independently so the group is smoothly deformed as it crosses into a differently scaled area on a ZoomScape. This deformation prevents objects from disappearing at the top of the screen during this simple interaction (**b**). For an easier comparison, the ZoomScape setting shown here is the same as in Figure 13: objects are scaled 100% at the bottom of the screen and 25% at the top of the screen; the cursor is represented by a black dot. So that the deformation is easier to see, in (**b**), (**c**) and (**d**) we show the reference geometry in the background. The reference geometry of a group is the geometry of the group when all its members lay on a 100% area (as in **a**). At each step of the interaction, it is used to compute the current distance between the cursor and each member of the group (see text). Notice in (**c**) how the objects below the cursor are pulled toward the cursor as soon as the cursor enters the 25% so that they can “catch up”.

scale each object in the group, but also how to change the geometry of the group. In our first attempt, we scaled each object according to its position and we considered the group geometry as a rigid body, the scale of which was adjusted according to the current value of the $shrink(x, y)$ function under the cursor. This simple solution caused some objects to temporarily disappear off the edge of the screen during the interaction (Figure 13). Not only does this solution create distracting

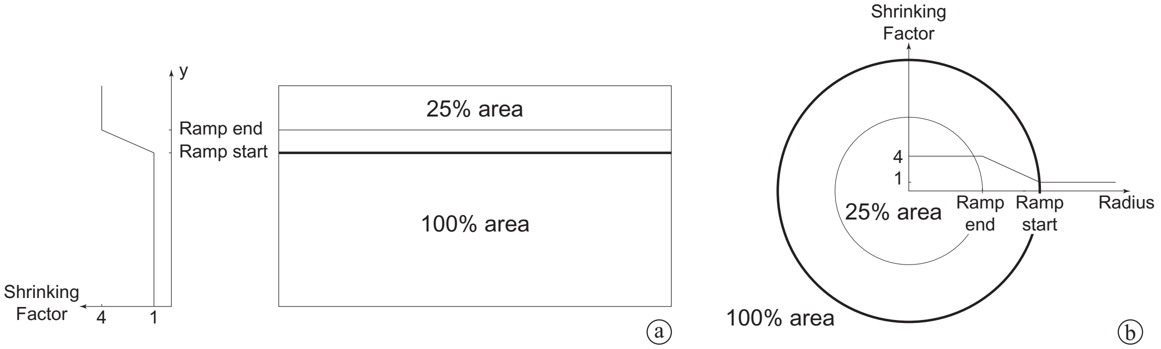


Figure 15: Two different ZoomScape settings. A diptych with a simple transition area between the two scale areas is used for PostBrainstorm (**a**).We can also divide a screen into rings, with a different scale for each ring. This could be used for a circular tabletop screen (**b**).

visual artifacts on the screen, but it can also cause some objects to be permanently lost if an interaction is interrupted while objects are offscreen.

To avoid this problem, ZoomScape scales each object in a group depending on its actual position and smoothly distorts groups as they are moved around. The resulting interaction sequence is shown in Figure 14. Notice that at any given time in the manipulation, all objects are visible. If the move were interrupted no objects would be lost off the screen and the user could just reselect them and proceed.

To achieve this effect, given two points A and O we define $\vec{u}$ as the direction between A and O and D_{ref} as the distance between A and O computed by taking into account the local shrinking factor along the path $\overrightarrow{AO}$. In effect, D_{ref} is the norm of $\overrightarrow{AO}$ if it were located completely in an area of uniform 100% scale. Given the function $shrink(x, y)$, D_{ref} is defined as:

$$D_{ref} = \int_{\overrightarrow{AO}} shrink(x, y) \tag{1}$$

While moving a group $\{O_1, \ldots, O_n\}$ using A as an anchor point (see Figure 14), our implementation keeps $\vec{u_i}$ and D^i_{ref} constant for each objectO_i (Figure 14). At the beginning of the move for each object O_i in the group, we computed the directions $\vec{u_i}$ and D^i_{ref}. Then during the rest of the move, we solved the following equation in O_i with the constraint that $\overrightarrow{AO_i}$ be colinear with $\vec{u_i}$:

$$\int_{\overrightarrow{AO_i}} shrink(x, y) = D^i_{ref} \tag{2}$$

Our system uses a parametric representation of each vector $\overrightarrow{AO_i}$ to compute (1) and (2) and Newton's approximation method [Atk88] to solve (2). More detail can be found in Section 5.2.3, page 60.

This approach is very general and can accommodate various ZoomScapes. Our PostBrainstorm tool uses a simple diptych configuration with a transition zone between the two areas (Figure 15a), but the system can also accommodate other geometries such as the one presented in Figure 15b, which is suitable for a round table. This system proved to be very easy to understand by our users. After seeing its use demonstrated once, most of them were able to take advantage of it to manage the information they were gathering on the Mural.

3.4 Managing latency

Users always push their tools to the limit. Accordingly latency should be dealt with while designing the interface, not dismissed as someone else's problem. Even though users can understand well that some tasks take longer than others, they are rapidly confused or annoyed if the delay is part of the user feedback loop. For example, a couple years ago printing was often a foreground task preventing the use of the computer for any other purpose. It is now handled as a background task, and even though printing times have improved little, decoupling the printing process from the use of the interface leaves the user with more flexibility in organizing her tasks and makes the printing process seem less disruptive. It is important to take this into account when designing an interface and provide a decoupling between lengthy tasks and interaction so that the user can carry out further interactions while the task is being completed in the background.

3.4.1 MultiPoint

One instance of how we addressed latency came about during our design of the Geometer's Workbench, a new front end for the Mathematica engine [Wol96], further described in Section 6.1, page 73. While using this front end, the user needed to pick one member from a family of parametric surfaces. Each parametric curve was defined by two parameters(u, v), and it took a couple of seconds for the system to compute the new curve given(u, v).

The standard answer to this problem would be to provide the user with two sliders (one to adjust each parameter) and a feedback window to display the curve generated by the current parameter input. Unfortunately that approach does not work well in presence of latency. Because a design relying on sliders uses time multiplexing to let the user explore the parameter space, latency is part of the interaction loop. As the user moves parameter sliders around, the feedback window is not immediately updated, causing confusion about the action commanded by moving the two sliders.

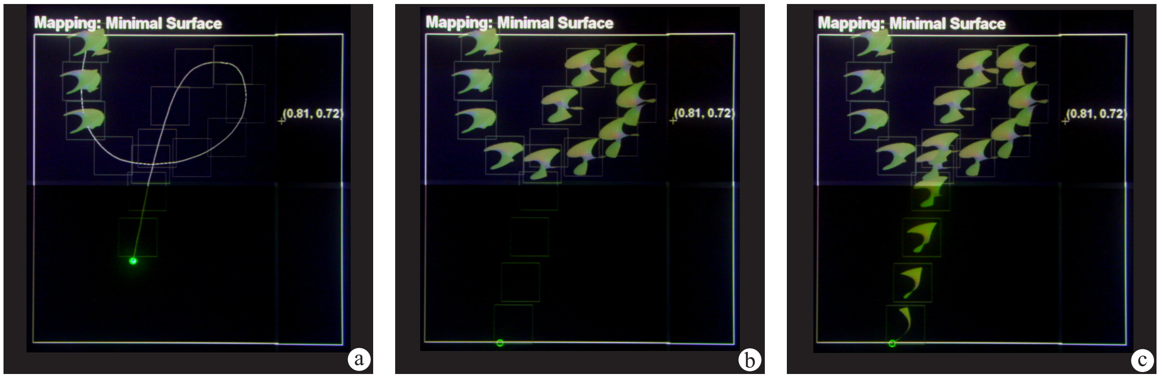


Figure 16: Using MultiPoint to explore a 2D parameter space. The Geometer's Workbench interface is designed to compensate for the latency of mathematical computations, by queuing calls to Mathematica and displaying results as they become available rather than blocking the user from further interaction. Here we show the evolution of the display as the user draws on the MultiPoint window. Notice that the pen trail is resampled by the system and that the trail of samples is updated progressively as the results of mathematical computations become available. Image (**b**) was taken 2.9 seconds after (**a**). Image (**c**) was taken 1.8 seconds after (**b**)

To use this design, users had to artificially slow down their interactions to accommodate the latency and stay in lockstep with the output.

Our solution, named *MultiPoint,* decoupled the interaction and the latency. Instead of using temporal multiplexing we provided the user with spatial multiplexing: the MultiPoint window represents the parameter space with u on the bottom axis and v on the left axis. To observe the shape of a specific (u, v) curve, the user points to the intersection of the desired coordinate pair in the MultiPoint window. A stand-in proxy square provides immediate visual feedback confirming the selected point. While the system processes the request, the user can continue picking points without waiting for the rendering. As requests are completed, the dummy squares are filled in with the relevant images. For each request the latency occurs outside the interaction loop. When the user is satisfied with the investigation she can use FlowMenu to designate one of the samples as the current setting for the system. This mechanism is not limited to individual samples: to observe continuous evolution of the parameter space, the user can draw a curve on the MultiPoint window as shown in Figure 16. MultiPoint encourages the user to explore the (u, v) parameter space as a two-dimensional space instead of forcing her to choose two decoupled one-dimensional choices. Another advantage of using space multiplexing is that the user can keep a record of past exploration, allowing her to compare her different options.

MultiPoint demonstrates how the special characteristics of a large, high-resolution display can be used to solve interaction problems such as latency. The high pixel count of these displays opens the door to techniques well known in printed matter, such as the small multiples design [Tuf91] where a range of alternatives are shown side by side to let the reader compare them at a glance.

Chapter 4
Display Design

For all direct manipulation user interfaces, the display is the interface between the user and the computer. The success of users' experiences with such an interface will depend on the visual quality of the display and the convenience of the tools we develop for interacting with it. For this project, our goal was to deliver a seamless, wall-size display with enough resolution so that people interacting with the screen at arm's length could not perceive its underlying pixel structure. We wanted the system to encourage users to write and interact directly on the screen.

When our project started, such a surface was unavailable, but emerging technologies such as tilable LCD display (flat panel) [Rai] and organic LED [Cam] made us believe that it would become available as a commodity in the near future. In order to carry out our investigations, we decided to simulate such a surface using today's technology. We focused our design on relatively inexpensive solutions tuned to letting us study direct interaction and focused our efforts on delivering resolution two to three times higher than in existing systems [PMMH93, LCC+00, Sma]. We rejected tiling discrete elements such as LCD panels or video boxes because the borders between elements disrupt the illusion of a unified surface and disrupt fluid pen motion. Since no single projector can provide a whiteboard-size display at our target resolution (64 dpi), we tiled several projectors together to build the overall image on a seamless screen. To avoid self-shadowing that would result from interacting with a front-projection system, we used rear projection.

Tiling offers many advantages for creating a wall-size image. Using tiling, one can use inexpensive off-the-shelf projectors to create a large image surface instead of relying on one large expensive imaging element. Tiling is also well suited to creating a scalable system. In our system, each projector is connected to a computer equipped with high-end rendering hardware. As more and more projectors are added to the system to increase its maximum imaging size, more and more rendering power is also added so that the system can handle the increased rendering load. Finally, tiling also makes it possible to create a more compact setting, which is needed to limit the effect of optical aberrations. As the size of the projected image increases, the throw distance (distance between the projector and the screen) has to increase if the focal distance (and the distortion level) stays fixed. Using a collection of small tiled imaging elements it is possible to increase the size of the screen by adding projectors without modifying the required throw distance to the screen.

There are many ways to design a tiled display [Bar, FL00], and the following criteria must be taken into account for any particular design to be successful:

- **Cost**. How much money should be spent per tile or per pixel? This parameter can vary by several orders of magnitude depending on the resolution and the degree of control available.
- **Flexibility**. Will the display be installed once and seldom adjusted, or can the design make it easy to change the display configuration?
- **Design complexity**. Shall the design use custom-made parts or mostly off-the-shelf components?
- **Target usage**. Will the display be used at arm's length, in which case precise alignment, a unified screen surface, and high resolution are primary design considerations, or viewed from a distance, in which case color and brightness corrections are more important?

Our choice for each of these criterion reflects our focus on building a tool to explore interaction on a large high-resolution surface. Indeed, high quality displays with performance similar to our goals are available commercially [Bar] but were too expensive to consider. Our goal was to deliver a more affordable system that would be easy to build and adjust. We used a 4 x 3 array of light DLP [TI] projectors so that we could build an inexpensive yet accurate alignment mechanism. We expected each display configuration to be stable in time so we relied on a manual alignment technique instead of a more expensive and complicated motorized system. Finally we relied on a simple solution (adjusting projector controls) to improve brightness and color calibration to keep our design simpler and less expensive.

This chapter will present our screen design in more detail. First, we will explain how the respective limitations of projector, screen, and the human eye guided our choices for the target value of the screen resolution (64 dpi). Then we will describe our alignment mechanism. One of the

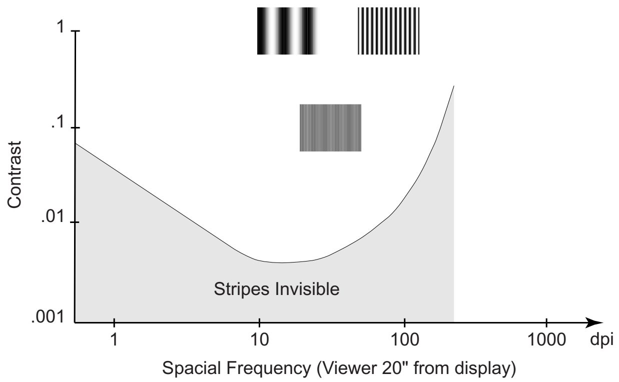


Figure 17: Contrast requirements for the human eye. Curve shows the minimum contrast required for a human eye to detect a sinusoidal signal displayed at a distance of 20 inches. Three typical signals are shown as an example (Eye data from [Rog83] for a luminance of 3cd/m^2).

shortcomings of tiled projection displays is that they require precise alignment to deliver the illusion of a seamless image. Our approach included the design of mechanisms to adjust the position of the image on the screen as well as a means to measure and correct any misalignments. Using our alignment protocol, described in Section 4.2.3, one can align our 12 projector screens in less than 12 hours. Next we will describe the observed limitations of our system with regard to brightness and color differences and how we think they should be addressed in the future. In the last section we present our goals for the input system and how it was implemented. The work described in this chapter was a joint project with Maureen Stone.

4.1 Resolution and contrast

The final resolution of our prototype (64 dpi) was dictated by many constraints. The first limiting factor was the maximum resolution that can be projected on the screen by the projector. At the time we designed this prototype, the readily available imaging elements could create an image of 1024x768 pixels using DLP or LCD or 1280x1024 pixels using the new D-ILA technology [JVC]. Projectors on the market were able to project an image measuring as small as 20-inch diagonally on the screen, delivering 64 dpi for a DLP/LCD system and 80 dpi for a D-ILA system. The next constraint was how close the projectors could be packed together. With each projector creating a rectangular image measuring 20 inches diagonally, the projectors had to be arranged so that the center lines of these rectangles were slightly less than 16 inches apart horizontally and 12 inches apart vertically.

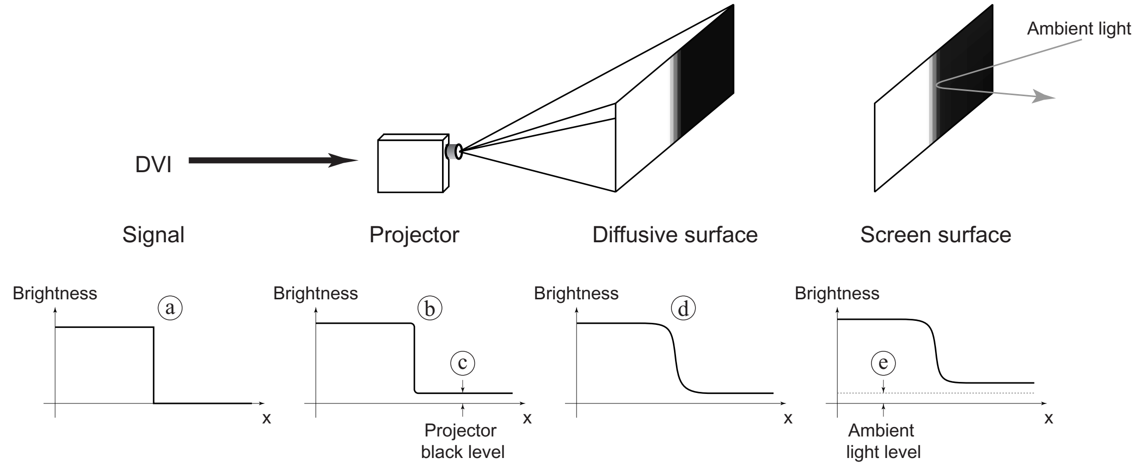


Figure 18: Factors influencing the contrast and sharpness of a picture on the screen. The final contrast and sharpness of the picture on the screen is the result of the transfer functions of the projector, the screen and the viewing conditions. This example shows the progressive distortion of perfect step function (provided as a DVI signal) (**a**). Imperfection of the optical system limits the step sharpness (**b**). Because the projector always outputs some light, even when a black signal is provided (black level), the contrast of the picture is degraded (**c**). As the image forms on the screen, the diffusive material the screen is composed of limits the step sharpness further (**d**). Finally ambient light interacts with the screen surface, adding an ambient light level to the picture and reducing contrast further (**e**).

Of course another constraint comes from the performance of the human visual system. Among other factors, human visual acuity is dependent on the contrast level of the image being viewed on a display [Rog83, Pal99]. A typical response curve is shown in Figure 17. To build this curve, subjects were presented with a sinusoidal grating, the contrast ratio of which can be modified. For each spatial frequency, the curve shows the minimum contrast ratio at which the subject can identify stripes. Figure 17 has been adapted from [Rog83] to show the response (to the fundamental frequency) expressed in dots per inch for an observer standing 20 inches away from the display. As the resolution requirement increases, one must use images with a higher contrast ratio to ensure good perception.

The contrast of the image as it appears on the screen is the result of the composition of transfer functions from all elements involved in creating that image. Figure 18 shows an example of how the different aspects of a rear-projection system combine to affect contrast. In the top row, the system is provided with a DVI signal representing a perfect step shown in Figure 18a: the left side the picture is showing full white and the right side of the picture is showing full black. Imperfections of the imaging element and in the projector's optical system limit the sharpness of the white-to-black transition in the projected image (Figure 18b). Because the projector outputs some light even when a black signal is provided *(black level)*, the overall contrast of the picture is diminished again (Figure 18c). When the projected image reaches the screen, the step is deformed even further

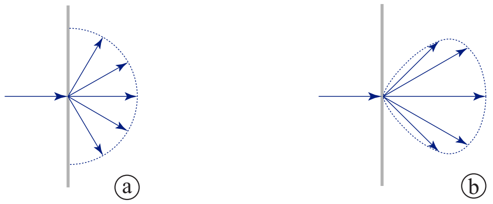


Figure 19: Screen gain. A screen with a gain of 1.0 is perfectly diffusive and outputs light evenly in all directions (**a**). As the gain increases, more and more light is output in a forward direction. (**b**) shows a typical output profile for a screen of gain 1.5, for which the intensity of the light emitted perpendicular to the screen is 1.5 times the intensity emitted by a perfectly diffusive screen.

because the screen's active surface is diffusive (Figure 18d). Finally ambient light in the room interacts with the screen's active surface adding a constant light level to the picture seen on the screen, further reducing the overall contrast (Figure 18e). Let us study each element in turn.

4.1.1 Imaging

We looked at three imaging technologies readily available at the time of the project: Liquid Crystal Display (LCD), Digital Light Processing (DLP), and D-ILA, a reflective LCD-based technology. Each technology provides pluses and minuses and our decision was influenced by several factors including black level, absolute contrast of the imaging element, weight and characteristic dimensions of the available units. It was also important for our needs that the projector used would accept DVI. At the time we were ready to order our hardware, the Compaq MP1800 DLP projector [Com] proved to be a good choice for our needs.

4.1.2 Screen

As shown in Figure 18d, the screen is one of the most limiting factors in the overall quality of the picture. Screens are characterized by their contrast and gain. Gain, for a rear-projection screen, measures the spatial distribution of the light output by the screen. Screens of gain 1.0 are perfectly diffusive and output an equal amount of light in all forward directions (Figure 19a). Screens with higher gain create a more and more directional distribution of light, outputting most of the incoming light perpendicular to the screen (Figure 19b). As the gain increases, it becomes more and more difficult to see the picture on the screen from a side angle. But even as a diffusive (low-gain) screen helps increase the range of viewing angles, it also limits the contrast because it creates cross-talk between adjacent pixels.

While designing a display, contrast and gain must be kept in balance. On one hand, the more diffusive a projection screen is the more efficient it is in creating an image visible from side angles. On the other hand, the more diffusive a screen is the less contrast it can provide.

Looking at the human eye response curve (Figure 17) one would think that picking the highest-contrast screen would provide the best solution. In fact high-contrast, high-gain screens are often used in auditoriums and other situations in which the audience's position is known in advance and is relatively far from the screen.

Our goal of arm's-length interactions meant not only that the user's position would not be well known (as they could be standing anywhere along its six-foot length), but also that side-angle viewing would be very common. Using a high-contrast screen would result in severe brightness variations caused by a high gain.

Our first prototype used a low-gain screen to provide a uniformly bright picture. But as we used this first-generation display more and more, it became clear that the screen diffuseness was limiting the sharpness of the picture that could be shown on it. While we were designing the second-generation system, a new type of refractive screen (BlackScreen™) developed by Jenmar [Jen] became available. This screen provides very high contrast while keeping gain relatively low. The brightness variation created by the gain was judged less important than the potential gain in sharpness and in the final prototype of the system we used the BlackScreen™ SG 100 on acrylic.

4.1.3 Rejection of ambient light

The light that is reflected off the screen and mixed with the projected image is also a source of diminished contrast. Screens using diffusive material are particularly sensitive to such light since they diffuse the incoming light back toward the viewer, making dark parts of the picture look lit. Jenmar's screens use beads embedded in an opaque substrate, which prevents the diffusion of ambient light.

4.2 Tiling

Tiling presents a specific set of design challenges. Because people are interacting close to the display, it is important that the tiles are aligned accurately so that content such as text, sitting on a tile boundary, can still be read. Because we are using several projectors side by side, slight differences in color and brightness characteristics will show and disrupt the illusion of a unified display surface. Finally, because each tile we used is relatively small, variations in the tile brightness or color will be picked up more easily, disrupting even further the illusion of a unified display. We will look at these problems in turn.

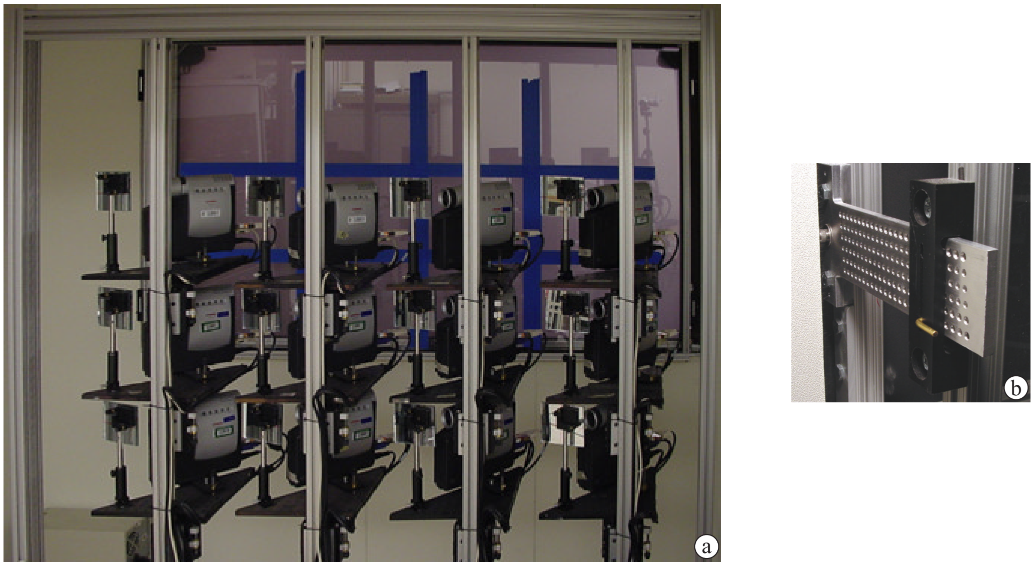
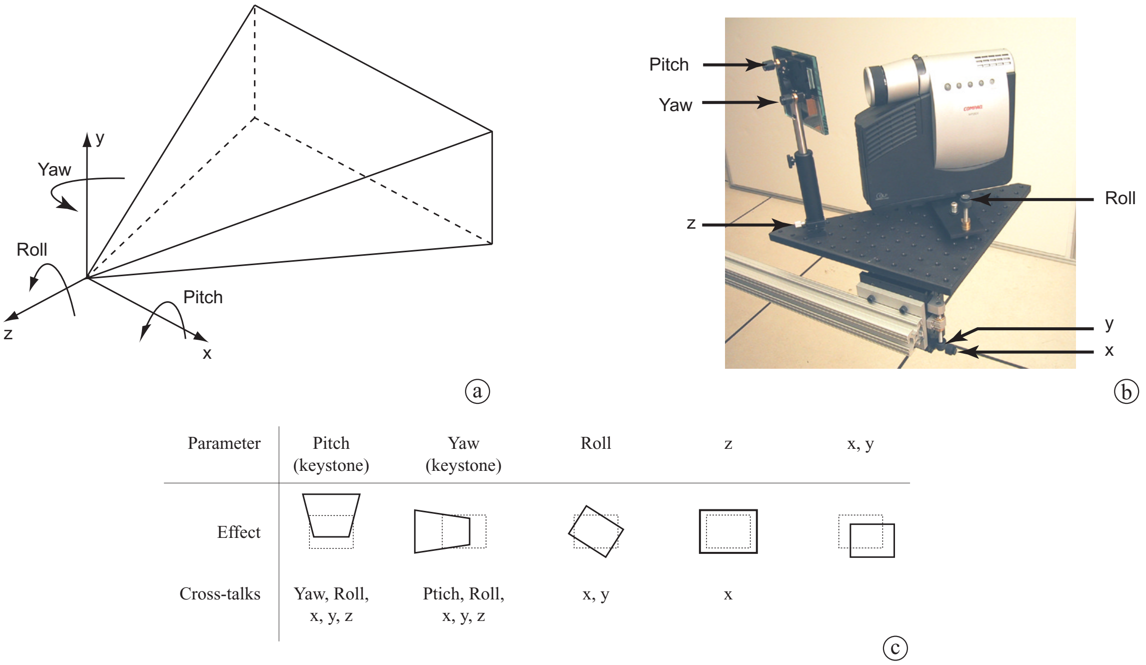


Figure 20: Overview of the final projector array setup. The system skeleton shown in (**a**) is built out of extruded aluminum, and each projector is mounted on an alignment platform (shown in more detail in Figure 21). In the background, one can see the masking tape (blue bands) used to mask the outermost 16 pixels of each tile (see Section 4.2.2). The pieces of tape adhere to a large sheet of acrylic held in position by four mounting points such as the one shown in (**b**). The pattern of holes in a mounting point is designed so that the distance between the masks and the screen can be adjusted by 1-mm increments.

4.2.1 Alignment

After building several prototypes out of wood and using an off-the-shelf projector alignment mechanism, it became clear that this approach was not accurate enough for close-up interactions. Standing close to the Mural, it is easy to detect misalignments less than a pixel wide, so we needed a rigid and finely adjustable framework. For our final prototype, we used a commodity extruded aluminum structure [80/20] and designed an alignment platform made out of standard optical bench components (Figure 20). Achieving a good alignment between adjacent tiles then became our main priority.

To create a good tiling, one has first to align the image plane of each projector with the surface of the screen to avoid *keystoning* aberrations. Then the size of each tile needs to be made uniform and its main axes aligned with the screen's main axes (tilt corrections). Finally tiles must be aligned vertically and horizontally to create the final image (on-screen translation). During this process one needs three degrees of freedom for adjusting rotation and three degrees of freedom for translation (Figure 21): keystoning aberrations are corrected by modifying the *yaw* and *pitch* of the frustum; size is adjusted by modifying the length of the optical path (z); tilt is corrected by modifying the *roll*; and alignment of tiles is performed by adjusting *x* and *y.* Ideally we wanted each adjustment parameter to be independent and without cross-talk. This can be done using a gimbal,



Parameter	Pitch (keystone)	Yaw (keystone)	Roll	z	x, y
Effect					
Cross-talks	Yaw, Roll, x, y, z	Ptich, Roll, x, y, z	x, y	x	

Figure 21: Aligning each tile on the screen. One needs to adjust six degrees of freedom to correctly position and align the projected tile on the screen (**a**). Our design (**b**) provides one knob for each parameter, but cost and complexity constraints led to a design that could not eliminate cross-talk between the different adjustment controls (**c**). By adjusting parameters from left to right as they are presented in (**c**) one can reliably and accurately place each tile at the correct location and alignment.

whose rotation center coincides with the optical center of each projector for the three parameters of rotation, and a three-axis translation table for translations. Unfortunately such a system would have been both expensive and complicated to build. Looking at a typical alignment process and other approaches, such as the one described by Hereld [HJPS00], it became clear that cross-talk between translation and keystoning adjustments were the most problematic

Unlike other approaches such as the one described by Hereld [HJPS00], our system lets us correct keystone aberrations and tile positions independently (Figure 21c), for a fast, repeatable alignment process. Because we used lightweight projectors, we were able to use off-the-shelf optical bench equipment for most of its components, making it inexpensive to build.

4.2.1.1 Keystoning correction

If the projected image plane of a single projector and the screen plane are not parallel, the image tile will not appear rectangular but as a keystone, as shown in Figure 21c (yaw and pitch column). To make the two planes parallel one needs a way to adjust the main axis of the projector frustum. This can be done either by rotating the projector or by placing a mirror on the optical path and rotating the mirror to adjust the direction of the frustum. Because the mirror can be small and light, it is easier to rotate than a projector. In our design we adopted the latter solution, bending the

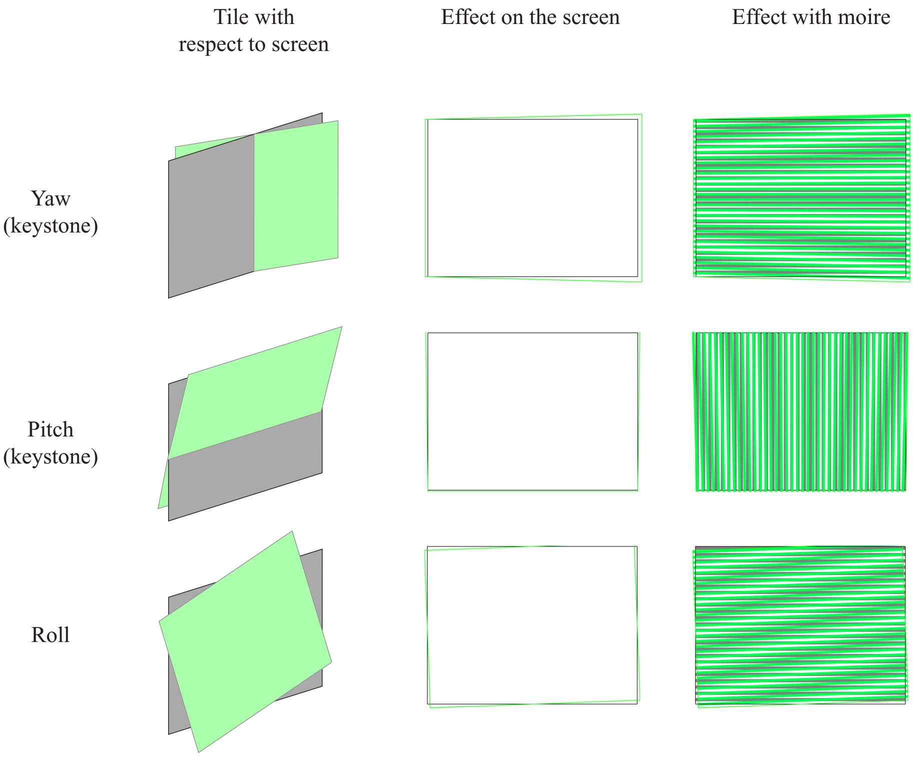


Figure 22: The use of moiré patterns to detect alignment problems. Using a moiré pattern it is possible to detect very slight misalignments. For each rotational parameter (yaw, pitch, and tilt) the figure shows the resulting configuration in space (left column), a misaligned rectangle projected against a reference rectangle (middle column), and the moiré pattern visible on the screen using our technique (right column). To adjust yaw, we use horizontal stripes to form a moiré pattern (top right). The moiré stripes diverge to the right for a projector pointing to the right. To adjust pitch, we use vertical stripes to form a moiré pattern (middle right). The moiré stripes diverge toward the top for a projector tilted upward. Either direction can be used to adjust tilt (bottom right). The orientation of the moiré stripes show the direction the tilt.

optical path with a mirror on a kinematic mount. Using the kinematic mount, the direction of the mirror and of the exiting optical path can be adjusted precisely using one knob for each axis. Using an 80 thread per inch (tpi) adjustment screw, the direction of the frustum can be adjusted to 7 min. of arc.

One of the main difficulties of keystone correction is accurately measuring the error of the current setting and knowing which correction should be applied to improve the alignment. To solve this problem we used a moiré pattern as a measuring tool. A moiré pattern is an interference pattern that is created when two striped patterns of similar frequency are superimposed on each other. Observing the resulting pattern, one can detect small variations of frequency, small distortions of one pattern, or the direction and amplitude of misalignment between the two patterns [ON63]. Figure 22 shows several examples of moire patterns. Moire patterns have been used for a long time in the printing industry to align consecutive colors in a print run [Lev93], and it was easy to adapt

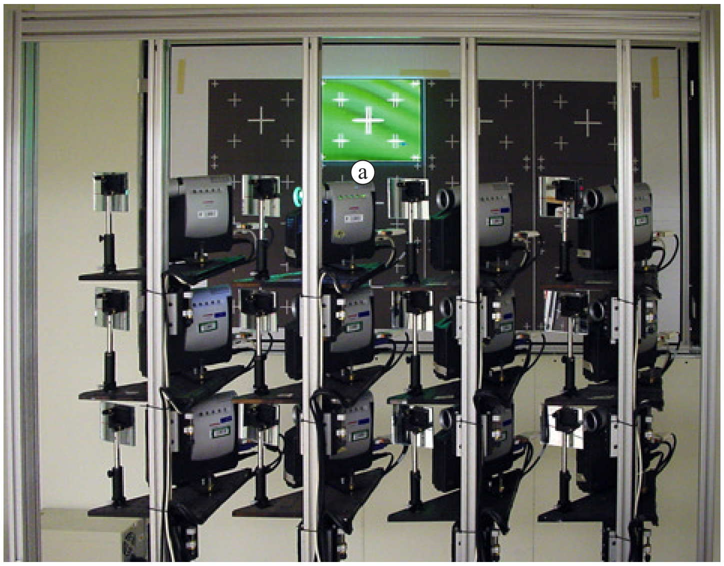


Figure 23: The reference alignment pattern in place. The pattern is printed on a large piece of foamcore and its distance from the screen can be adjusted using the mechanism shown Figure 20a. We show one projector projecting a set of vertical green strips and the resulting moire pattern can be seen (**a**). The crosshairs superimposed on the reference pattern provide a reference to detect parasite rotations and translations.

this technique to our use. We printed an exact reference pattern on a large piece of foamcore and placed it at a precise canonical position behind the screen facing the projector arrays (Figure 23). For each projector, the pattern presents a two-dimensional grid (one pixel on/one pixel off both vertically and horizontally) whose frequency is such that it would correspond to a 64-dpi pattern if it were projected on the screen. The moiré pattern forms on the foamcore (Figure 23a) when a projector projects a vertical or horizontal striped pattern (one pixel on/one pixel off). One can correct the pitch using vertical stripes, and one can correct the yaw using horizontal stripes. Because orthogonal stripes do not create a moire pattern, the same printed pattern grid can be used for both yaw and pitch alignments. Either pattern can also be used to detect rotation. Looking at the pattern it is easy to see which knob should be adjusted and in which direction. As one corrects for keystoning, one also translates and tilts the picture on the screen. These effects can be corrected without modifying the keystone correction settings, using the controls described below.

4.2.1.2 Rotation correction

Once keystoning is corrected, we must square each tile edge to the screen. Ideally we should have used a rotation stage, the axes of which would follow the projector optical axis, but this was too expensive and complicated. Instead we built a simple and inexpensive tilt stage that doubled as a support bracket for each projector. The stage is shown in Figure 24. It is a simple L bracket; the vertical side of the L is screwed to the projector mounting plate and the horizontal side rests on two ball bearings (used as an hinge) and a micrometer screw (used as an adjustment mechanism).

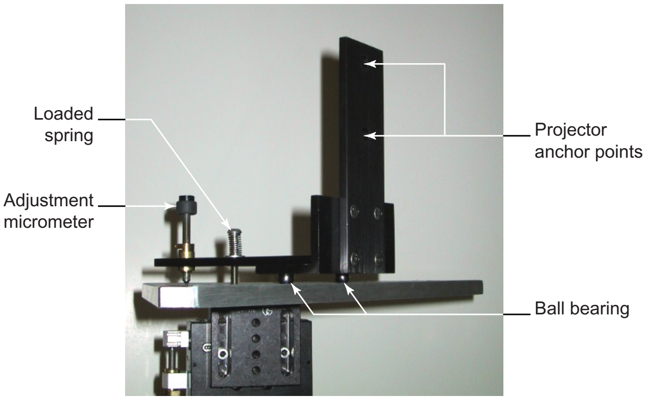


Figure 24: Projector rotation stage. To simplify the design and lower cost, we designed a simple rotation stage that doubles as a projector bracket. The two ball bearings at the bottom of the stage define the rotation axis while a micrometer screw is used to adjust the tilt. The compression screw holds the projector securely to the bench while allowing for adjustment.

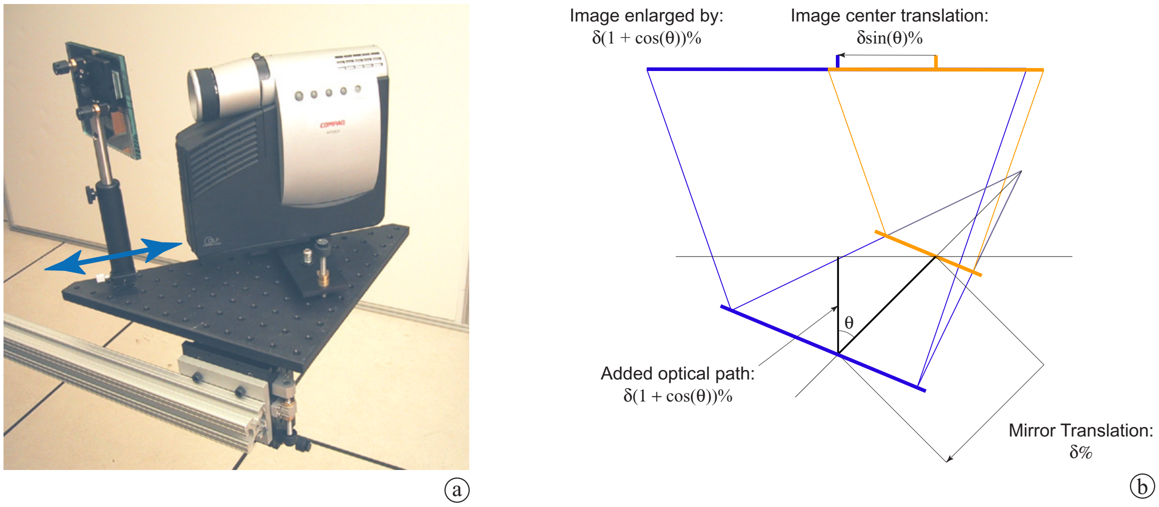


Image enlarged by: δ(1 + cos(θ))%
Image center translation: δsin(θ)%
θ
Added optical path: δ(1 + cos(θ))%
Mirror Translation: δ%
a
b

Figure 25: Modifying the tile size. To modify the size of a tile we modified the length of the optical path by moving the mirror along the optical path axis (**a**). Our bending beam configuration amplifies the adjustment by a factor of (1 + cos(θ)) and also creates a translation of the image that can easily be corrected with the XY translation stage (**b**).

The horizontal side of the L is held tight to the optical bench with a compression spring. Adjusting the micrometer screw tilts the L around the axis that is defined by the two ball bearings. Our design allows us to adjust the tilt of a tile to 1s. of arc which, given the tile size, corresponds to 1/6 pixel at the corner. Since the axis of rotation is not co-linear with the optical axis, adjusting the tilt of a projector translates the image. This translation can be corrected without modifying the yaw, pitch, or tilt using the translation procedure described below.

4.2.1.3 Tile size

It is important that each tile be precisely the same size so that each pixel will be the same size. To modify the size of a tile, one can either change the zoom setting of a projector or change the

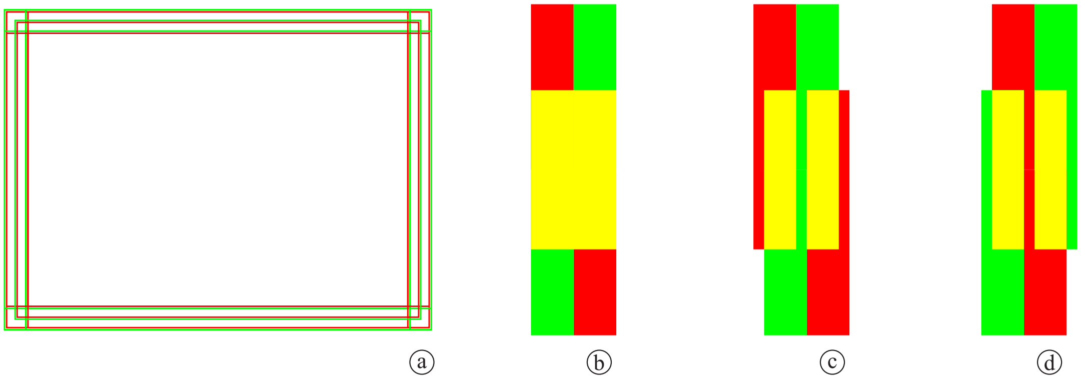


Figure 26: Tile alignment pattern. During the alignment process, each projector projects the pattern (**a**). The border in the pattern is two pixels wide with the innermost pixel red and outermost pixel green. The distance between the inner and outer borders corresponds to the overlap (32 pixels in our case). Because of color additivity (Red + Green = Yellow) when the alignment is perfect, the overlapping pattern look like a 2 pixel wide yellow line (**b**). If the tiles are too far from each other, two red lines appear on each side of the yellow line (**c**). If the tiles are too close to each other a red line appears on the center of the yellow line (**d**).

length of the optical path. Because the zoom adjustment of the projector we used was somewhat crude, we included a way to adjust the optical path length for each projector. Modifying the optical path by translating the projector would have been difficult, but since our design included a mirror to bend the optical path, we could move the mirror along the optical path as a way to refine the size of the image. As shown in Figure 25, the movement of the mirror is amplified: as we move the mirror away from the projector on the optical path, the mirror also moves away from the screen, extending the optical path even further. This system allows us to position the mirror to 10 microns, which means that the size of a tile can be adjusted to within a 1/100 of a pixel. Note that moving the mirror also translates the picture on the screen (Figure 25). This was not a problem in practice since this shift could be corrected easily with the final adjustment mechanisms.

4.2.1.4 Positioning the tiles

Once tile orientation and size had been set, we could align the tiles in relation to each other. In our design the projectors and all the previously discussed adjustment mechanisms rest on an optical bench that is connected to the support structure with an XY translation stage (shown in Figure 21b, page 38). A pair of standard 80-tpi precision screws allows us to adjust the tile position on the screen to within a couple of microns accuracy.

As in keystone correction, figuring out the error of the current setting was one of the main difficulties in aligning tiles. We needed to correct for even sub-pixel discrepancies, as these can disrupt the viewer's sense of a seamless image. To adjust tile alignments, we created a test pattern, shown in Figure 26a. The pattern is composed of a set of frames whose borders are two pixels wide: the inner pixel is red and the outer pixel is green. While aligning tiles with each other, we let adjacent

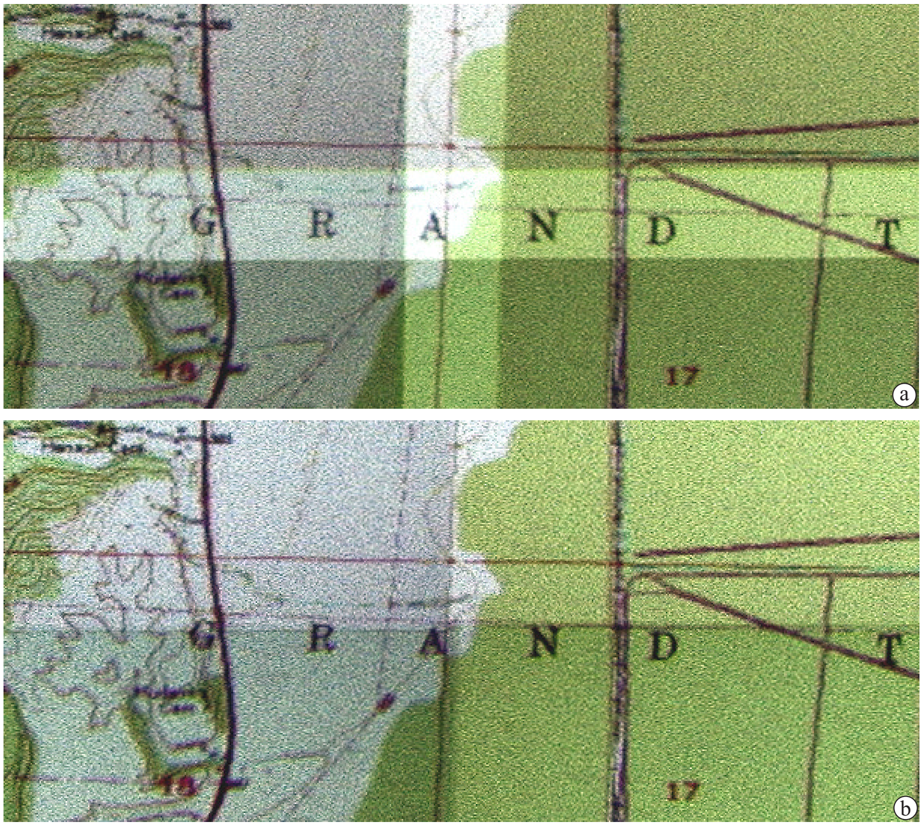


Figure 27: Resulting seams. In (**a**), the seams where four images overlap are shown without masking to show the quality of the alignment: the *A* in the word *GRAND* is projected from four projectors to virtually the same position on the screen. When the masking is in place (**b**), the seams are only visible in variations of color and brightness between tiles. Both pictures were taken 5' from the ground, looking downward at the center of the lower horizontal seam, and are shown near real size. The overlap area is 32 pixels wide.

tiles overlap and used color additivity to detect misalignment. As the tiles overlap, a perfect alignment results in a 2-pixel-wide yellow line (red + green = yellow). If there is a discrepancy in alignment, then the border will show red or green spots as shown in Figure 26c and d. For example if two tiles are too close to each other, we can see a sub-pixel-wide red line inset in the yellow border. This detection method allowed us to detect misalignments smaller than 1/5 pixel wide. Figure 27a shows the alignment of two tiles on a complex map picture.

4.2.2 Seam correction

Seams are the areas of the screen where the picture transitions from one tile to the next. Ideally we would have liked to abut the tiles together, but it was impractical: pixels at the edge of a projected tile can show distortions such as pin cushioning or chromatic aberration. There can also be variations in picture quality between tiles, and the seaming process is often used to smooth differences away. We experimented with several systems of seam correction including:

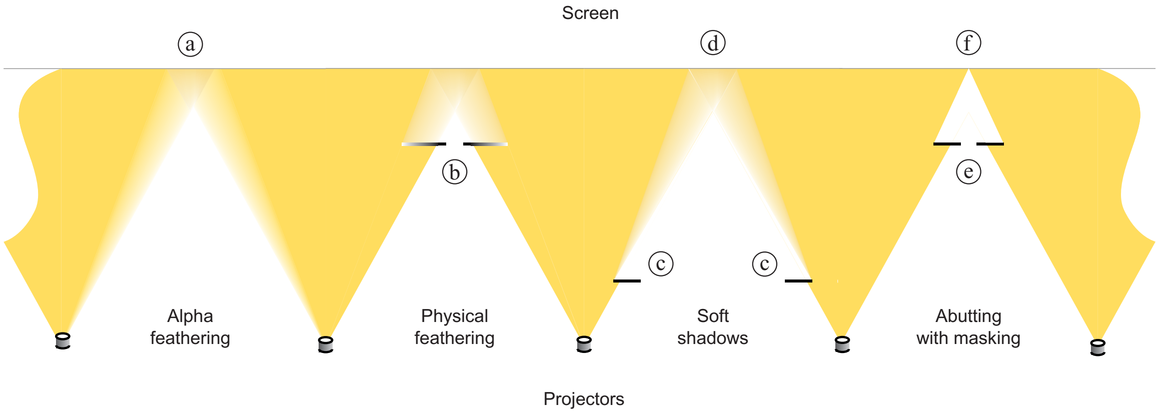


Figure 28: Seaming methods. In the area where images from two projectors overlap, brightness correction must be performed. The intensity of the area of overlap can be modified in software using an alpha mask, wherein the picture projected by each projector is pre-corrected to take into account the overlap (**a**). One can achieve similar results without post-processing the image between buffer swaps using a neutral density ramp placed in the optical path of the projector (**b**). Using a mask close to the projection center (**c**) creates a soft shadow (penumbra) (**d**) which creates an intensity ramp. Finally, a mask close to the screen (**e**) creates a sharp edge, making it possible to abut adjacent tiles without overlap (**f**).

- **Alpha feathering** (Figure 28a, [RBY+99, Sur99]). In this approach an alpha polygon is drawn on top of the scene just before swapping between the front and back buffers. The alpha polygon creates intensity ramps at each edge of the image, resulting in a uniform intensity along the seams. This method is very simple to implement but does not correct for a non-zero black level. Where tiles overlap, the black level leaked by each projector accumulates, increasing the black level where four corners overlap as much as four times. From a practical standpoint, this means that software seaming (using alpha ramps) forces one either to reduce the contrast by a factor of 4 or to leave a visible seam when black is rendered. Because alpha feathering uses a post-processing step at frame boundaries, this method also forces the system to run in double-buffer mode, preventing the use of direct feedback rendering in the front buffer (see Section 5.2.5).

- **Physical feathering** (Figure 28b). Neutral-density ramp filters are placed at the outer edges of the frustum to achieve the same effect as alpha feathering. Physical feathering is slightly more difficult to adjust than alpha feathering because it requires precise physical alignment, but it can correct the black level projected by a projector and create a seamless screen without decreasing image contrast. Creating the appropriate neutral-density ramp can be difficult and most techniques often result in a blurry region at each seam.

- **Soft shadows** (Figure 28c [LC99]). To avoid using a neutral-density ramp in the optical path, one can use a mask placed near the optical center of the projector to create soft shadows (penumbrae). This method is attractive in principle because it takes care of the black level, but in practice controlling the shape of the ramp can be as difficult as physical alignment.

- **Abutting with physical masking** (Figure 28d). In this method the outermost pixels are masked away to create a square tile without aberrations at its edges. The masking also helps reduce halo effects generated by the projecting element. The resulting tiles can then be abutted together. Given that we had methods for aligning the tiles very accurately, this approach was practical in our case.

Even though overlapping frames to create seams seemed tempting at first because one can create a smooth transition between slightly different tiles, our experience showed that this approach is difficult to implement in practice because the on-screen gain creates direction-dependent brightness variations (Figure 29). Furthermore, avoiding overlap would make it possible for us to use Fresnel lenses to reduce brightness aberrations (see Section 4.2.4.2)

In our final design, we masked the 16 outermost pixels with tape on a transparent screen placed a couple of inches in front of the screen's active surface. This configuration let us align tiles with one another easily and also remove the outermost pixel from each tile edge, which is often a source of geometric and chromatic aberration.

This solution is not perfect since it creates somewhat abrupt quality discontinuity (of color and brightness) between tiles, breaking the illusion of a unified screen. In practice we deemed this problem less important that the lost of contrast or the introduction of artifacts that would be created by overlapping tiles. Figure 27b shows the result of masking.

4.2.3 Alignment protocol

We will now present the steps for aligning tiled images projected onto our Mural. The full operation can be performed in less than an hour per projector and the alignment settings remain stable for several weeks.

1. We place the printed alignment pattern behind the screen, facing the projectors. The pattern can be held at the right distance from the screen using pins (Figure 23b).

2. For each projector, we correct the keystoning. During this phase, each projector projects a vertical (or horizontal) striped pattern. A small program makes it easy to switch between the two patterns for a given projector. We focus and sharpen the image on the printed pattern. Adjusting the mirror roughly, we center the projected pattern on the corresponding printed pattern. Using the zoom adjustment, we then find the position where no moiré is created by projecting the striped pattern onto the printed pattern. Then the zoom setting is

modified a little bit so that very clear bands (5-10) become visible on the screen. Alternating between projecting the horizontal and vertical patterns, we adjust yaw and pitch respectively. For each direction we adjust the mirror position until we can achieve parallel bands on the screen. Because we are using a kinematic mount, we must also correct tilt and translations as they appear so that the bands appear horizontal (or vertical) and centered on the target pattern. To make this process easier, large cross-hairs are superimposed on the moiré pattern (as shown in Figure 23). When the adjustment is done, we fine-tune the zoom again so that the moiré completely disappears. This step ensures that each projector has the same focal length and that each tile has the target dpi on the screen. Fine adjustments of pitch, yaw, and tilt are also performed at this point.

3. The alignment pattern is removed from the screen and replaced by a transparent screen for masking seams.
4. We turn on all the projectors simultaneously to align their fustrums to one another. During this phase, each projector projects the bicolored frame pattern shown in Figure 26. Using the translation control and the optical path-length control, we adjust the position and size of each tile on the screen. We use an arbitrary tile on the screen as our reference point and align the other tiles to the reference tile one at a time.
5. Using the same bicolored pattern, we use masking tape to mask the 16 outermost pixels on the bottom and left side of each tile.
6. Once each projector projects a uniform white rectangle, we mask 16 pixels from the top and right sides of each tile. The masking tape is applied so that one cannot see any line between tiles (Section 20 shows the final setting).

4.2.4 Color and brightness differences

Even though we can align the tiles with sub-pixel accuracy, the projected image on the Mural does not appear uniform. This is caused by slight differences between projectors as well as optical characteristics of the screen we used. Empirical evidence showed that even though color and brightness differences were noticeable at first, they did not affect the use of the display, so to simplify our design we simply use the projector controls to limit these differences. Here we will present an overview of these problems. More details can be found in [SB99] and [Sto01, Sto01a].

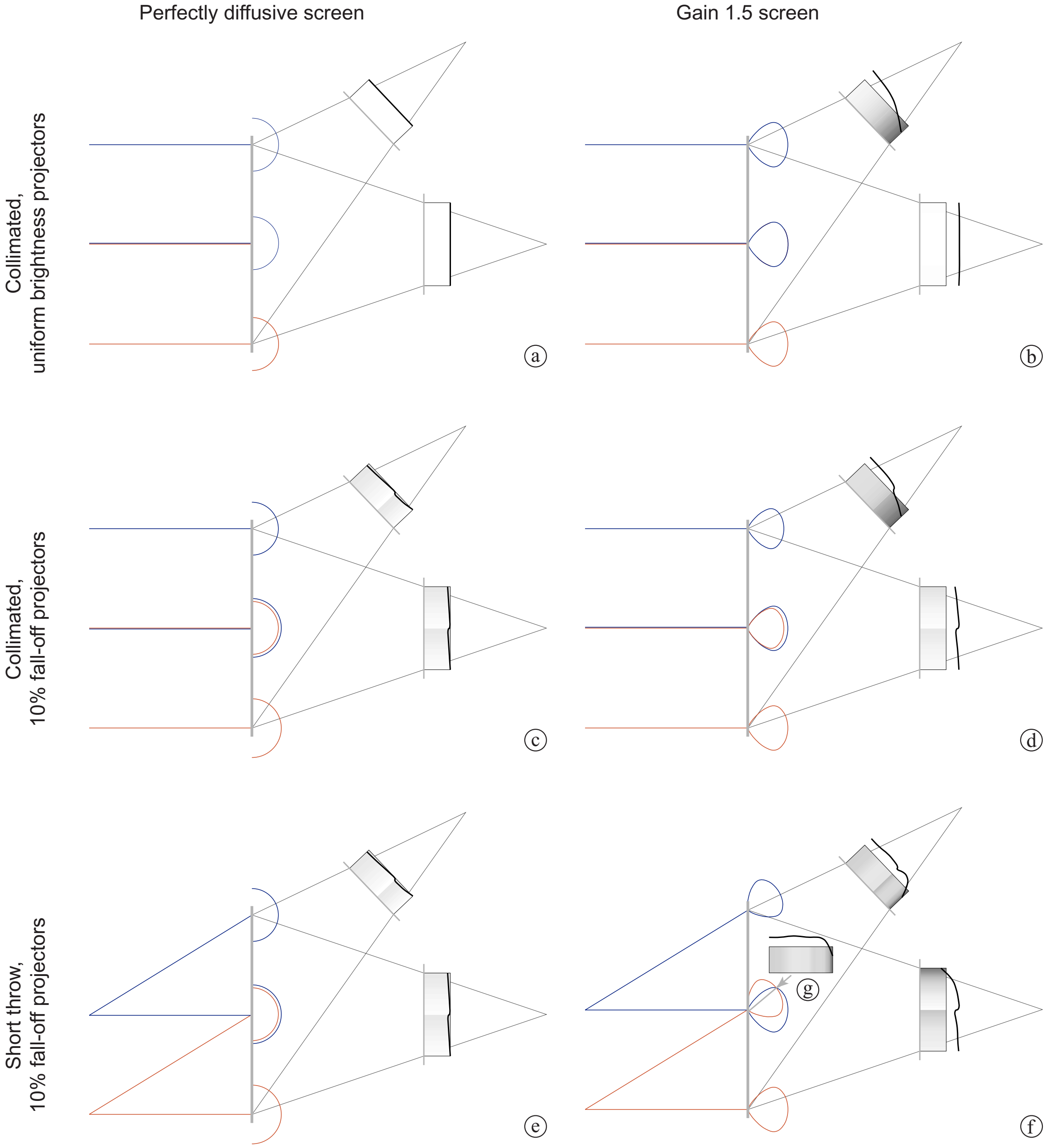


Figure 29: Screen brightness profiles. The brightness profile of the screen depends of the gain of the screen, the brightness uniformity of the projector, and the shape of the frustum. In the perfect situation the screen brightness is uniform from all directions (**a**). Because of the gain, looking at the screen from a side angle one can see large variations in the screen brightness (**b**) but cannot identify the tile boundaries. If the projector brightness is not uniform from the center to the edge of the tile, it becomes easy to distinguish tile boundaries on the screen both for gain 1.0 and gain 1.5 (**c**, **d**). Using non-collimated light amplifies the influence of the gain (**e**, **f**). Sometimes the different effects compensate for one another as in (**g**), which shows that at a specific viewing angle, gain and vignetting interact to create a seamless picture. This figure is only intended to illustrate phenomena at play and made numerous simplifications. It does not reflect experimental data.

4.2.4.1 Color difference

Both color uniformity within a tile and the color uniformity from one tile to the next are important for delivering a a seamless image.

4.2.4.1.1 Tile color uniformity

The color uniformity within a tile depends mostly on the technology used for the imaging elements. From our experience, LCD systems produce less uniform color than DLP systems. Because LCD systems use a dichroic filter to split the light into three primary colors, slight misconvergence of the incoming light can create color disparities on the screen. Furthermore LCD-based systems use light polarization to modulate the light intensity, which can cause each primary color to be polarized slightly differently. The presence of structural stress in any part of the optical path also creates slight color variations on the screen. In contrast DLP systems do not require polarization of the light source and often use a color wheel to generate color. This creates tiles that are very uniform in color.

4.2.4.1.2 Difference between tiles

Even though all the projectors we used are the same make and model, slight differences in each projector created perceivable color differences between tiles. Reports from users showed that even when those differences were noticed at first, they tended to become unnoticed as soon as the board filled up with content. This will probably hold true for whiteboard-type uses, but these discrepancies need to be corrected for uses such as industrial design or medical imagery. Color correction for large-tile displays presents many challenges. Stone presents an overview in [Sto01] and [Sto01a].

4.2.4.2 Brightness difference

Because of optical component imperfections, the image brightness is not uniform on the screen. First the projector’s internal optical system introduces some vignetting that causes the corners of the projected image to be dimmer than its center. But the main cause for the unevenness is the gain of the screen, as shown in Figure 29. When tiled together on a screen that is not perfectly diffusive, the difference in the angles of incidence between rays translates into differences in brightness. In our case the situation was made worse because each off-the-shelf projector we used was designed to sit on a table and project a square image at a height higher than the table, so it did not have a symmetrical frustum. In that case, the differences are greatest between the top of one tile and the bottom of the next.

Currently these discrepancies are more noticeable than the color discrepancies mentioned earlier and can be quite distracting when looking at a large data set. Unfortunately there is no simple solution to this problem for tiled displays. Using a screen with a lower gain can alleviate the problem but will diminish the screen contrast and perceived resolution. A method of placing Fresnel lenses in contact with the screen to collimate the image is often used for single projector displays, but we judged it too complicated to assemble for our prototype. As with color differences, reports from users showed that even when these brightness differences were noticed at first, they tended to become unnoticed as soon as the board filled up with content.

4.2.5 Discussion

Our design was successful for delivering a wall-size high-resolution display suitable to run our prototype applications and carrying out our user studies. Yet our prototype fell short of delivering a seamless display out of commodity hardware. Here are the lessons we learned from our experience.

4.2.5.1 Alignment

Overall, the combination of our adjustment stage and our alignment tools (the moiré pattern and the colored-frame alignment pattern) proved very successful for aligning the tiles. The alignment proved to be stable over several weeks, and users were impressed by the quality of our alignment.

The main problem we faced while assembling previous prototypes was that small variations in the alignment of one projector can have a ripple effect that makes the alignment of projectors on the opposite side of the Mural difficult if not impossible. Using a printed pattern (printed on a simple inkjet plotter) as an absolute reference proved critical to limit this ripple effect, and we believe that more refined printing techniques will limit this effect further. It is not clear at this point how our technique will scale as more and more tiles are added.

4.2.5.2 Appearance

Our original goal was to provide a seamless image. Even though we came very close to that goal with respect to alignment, our display still shows variations of color and brightness. This problem has a minimal impact during a brainstorming session but may be more of a problem for applications in medicine or biology. If brightness variation depends primarily on the optical setting or the imaging element used, color difference is inherent to tile display. Even when all projectors are the same make and model the variations introduced during the construction process lead to differences in gamut. In her papers Stone [Sto01, Sto01a] sets out a theoretical framework for correcting these brightness and gamut variations. At the same time, new capabilities of graphics hardware, such as programmable shader [NVI], make it possible to apply her framework during a fast post-rendering pass at each tile. We believe that this theoretical framework, in combination with new hardware capabilities, has the potential to deliver a relatively inexpensive tiled display with broad color uniformity.

4.3 Input System

We wanted our prototype to allow the user to interact directly on the surface of the screen. To do so, we had to choose between two broad classes of interaction styles: direct with bare hands or mediated through tools. While bare unencumbered hands are versatile and readily available, they lack the precision, task specificity, and robustness needed for many applications: one can draw more accurately using a pen or a brush than using one’s finger. Tools have also limitations. They are often task specific and often require some training before they can be used properly. Neither do they offer the feeling of direct manipulation that hands provide: manipulating clay with the hands is a completely different feeling than sculpting wood with tools. Let us consider the use of the whiteboard. Most of the time, a user works on the board with a tool such as a pen or an eraser. Both tools provide convenient advantages. With a pen one can write more precisely, whereas the eraser allows one to remove a large area at once. Nevertheless, users sometimes use their finger or fist as an on-the-spot eraser. This reveals another important aspect of tools: before a person can use a tool, they must get the tool into position, and in our everyday lives we weigh the pros and cons of this acquisition task against the advantages the tool will provide. To come back to our whiteboard example, people often elect to use their finger as an eraser to correct a small spelling error, so that they don’t have to switch between tools and be distracted from the task at hand.

In a recent paper Ringel [RBJW01] showed that it is possible to provide an interactive surface on which users are free to use either their finger or a pen to interact. We rejected this approach on two grounds: first, during meetings, one of the important uses of the finger is to point at items on the screen during discussion. Overloading this function of the finger with another seemed likely to be a source of error for many users—they could easily point at a touch screen just to discover that they were unintentionally creating a mark on their board; second it would have added too much complexity to our system because it required the use of a large touch-sensitive surface which would have had to have been custom made for our large screen size. For these reasons, we decided to fit our Mural with a tool-based interaction system that would not respond to touch. Given this constraint, we investigated two technologies that seemed most appropriate: optical tracking similar to the type used in the LiveBoard system [EBG+92] and the ultrasonic tracking often used in digital whiteboard capture systems.

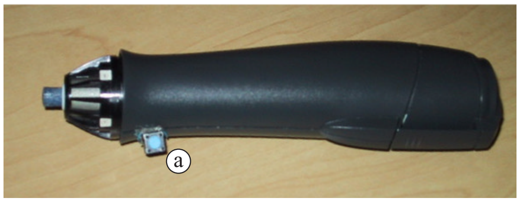


Figure 30: Pen with a command button used to interact with the Mural. Starting with a standard eBeam pen, we added a button on top (**a**), and had the firmware modified to increase the output power of the ultrasonic transceiver and manage the button.

4.3.1 Optical tracking

We first tried LumiTrack [CD02], a system developed by James Davis and Cindy Chen. LumiTrack consisted of a set of cameras installed behind the Mural screen, facing the screen to track a point of light created by an LED placed on the screen or by a laser pointer. The system was convenient because the laser allowed the user to interact with a screen from a distance. Unfortunately, latency made the simulation of ink on the screen difficult because it created a time gap between the ink trail and the pen. Furthermore, the sampling rate and resolution were insufficient to meet the requirements of the handwriting recognition system we were using.

4.3.2 Ultrasonic tracking

In recent years several commercial systems have been developed for tracking tools as they contact a digital whiteboard's surface [EFI, Mim]. Based on triangulation of an ultrasonic emitter, these systems are designed to record pen activity on a whiteboard for archiving or remote replay purposes. Most systems can recognize up to five tools (four colored pens and an eraser).

We chose eBeam from EFI [EFI]. The system comes with two pods attached at the upper corners of the screen. In our experimental setting we used one pen (Figure 30) that was modified in two ways:

- the power output of the emitter was increased so that it could accommodate the higher ultrasonic absorption characteristic of a rear-projection screen compared to that of a whiteboard,
- The firmware of the pen was modified so that the pen could be fitted with a button designed to trigger the command menu.

4.3.2.1 Calibration

Since the pen is a direct input device, it was important that the offset between the cursor and the pen be as small as possible. Our initial experience with the pen library showed that this requirement was difficult to maintain over the full surface of the screen.

To correct this, we added a calibration module to the existing library. After running a calibration program which asked a user to trace a coarse grid drawn on the screen with the pen, the library uses a simple bilinear interpolation method to create a calibration table for translating pen coordinates to logical screen coordinates. This simple method improved the uniformity of the calibration over the full surface of the screen.

4.3.3 Discussion

Overall our solution for the input system was functional, but user testing revealed some limitations. Because of the size of the screen, tracking was inaccurate at the very top of the screen, limiting user interaction in that area. Users also experienced difficulty with the pen in general since changes in the angle of the pen modifies the position of the cursor. But the main complaint expressed by users was that only one pen can be used on a screen at any given time. Even though our preliminary research showed that only one pen would be necessary, it was clear that often several people want to interact on the screen at the same time.

From our experience, it is clear that the current input technology falls short of the demands inherent in large display surfaces. Accurate tracking, multi-modal access (pen and finger, for example) and multiple access are the current areas of limitation. As this project nears completion the outlook is improving rapidly. New capacitive sensor technologies such as FingerWorks [Fin] and DiamondTouch [DL01] are providing new solutions for multi-modal, multi-access systems, while new digital pens such as Anoto's [Ano] may provide solutions for cursor accuracy and multiple access.

Chapter 5
System design

Whereas Chapter 3 presented our goals for the Stanford Interactive Mural user interface, and Chapter 4 presented the design of the display proper, in this chapter we will present the overall system design, highlighting key features that make it easier to develop applications for the Stanford Interactive Mural.

Since the Mural was developed for the visualization of large data sets, it is connected to a rendering cluster that uses OpenGL as its primary rendering language. Our first task was to implement *MilleFeuille,* a user interface toolkit, on top of OpenGL [Ope]. MilleFeuille provides window abstractions and interaction event dispatch. MilleFeuille differs from many toolkits in that it uses OpenGL as its primary rendering language, focuses on the use of transparency as a way to build interactions, provides support to implement ZoomScape, and helps the application programmer take advantage of distributed rendering API such as FruGL [ISH98].

The Stanford Mural was built to be one of the elements of the iRoom, the Stanford Interactive Workspace built to explore the interaction pattern of users in a technology-rich environment. The iRoom contains the Mural, three SMART Boards™, the Stanford Interactive Table, and an overhead scanner (Figure 31). Although these devices have different underlying implementations, the iRoom infrastructure [FJHW00] allows all of the devices to work smoothly together; information can be transferred from one device to the other, and one keyboard and mouse allow access to all the screens in the room.

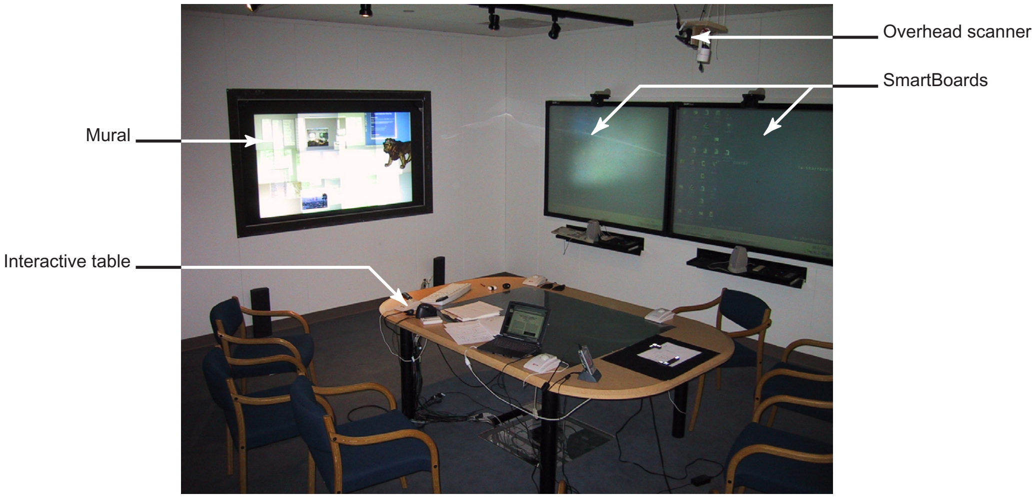


Figure 31: The interactive mural is part of the iRoom laboratory at Stanford. The lab presents a variety of interactive surfaces including three SMART Boards™ (.7 Mpixel each), an interactive table (.7 Mpixel), and the interactive Mural (~9 Mpixel). The room also provides easy laptop connectivity and an overhead scanner to quickly input non-digital content. The iRoom infrastructure gives the user access to all devices with one keyboard and mouse, allowing easy transfer of information between the interactive surfaces.

The Mural uses this infrastructure to receive information (such as pictures or presentation slides) from other devices in the room and to interact with FlowScan, the iRoom overhead scanner that we designed to support PostBrainstorm, our brainstorming tool (Section 6.2.2.1.3, page 87). Both features are important parts of PostBrainstorm.

In this chapter, after a brief overview of the overall system, we will describe the implementation details of the two parts of the iRoom system we were responsible for: MilleFeuille and FlowScan.

5.1 System overview

The system diagram is presented in Figure 32: the Stanford Mural is on the left. The screen uses an array of 12 projectors to produce a 9 Mpixel screen (4000x2240) and is connected to the graphics laboratory's rendering cluster. The cluster is built out of 32 dual Intel PIII processors running at 800 Mhz with 256Mb of memory. All nodes are connected through a Myrinet network providing a high-bandwidth/low-latency link. Twelve of the 32 computers are equipped with Nvidia GForce III cards that output Digital Video Interface (DVI) signals and are paired with the 12 mural projectors. The master computer that runs the application proper is also connected to the cluster via a Myrinet link. Each cluster processor and the master are running WireGL [HEB+01], a distributed version of the OpenGL API. Using WireGL, the master can use the Mural as a unified OpenGL context, even though the screen is rendered by 12 different computers whose output is tiled together to form the final image. WireGL also implements the distributed rendering API described

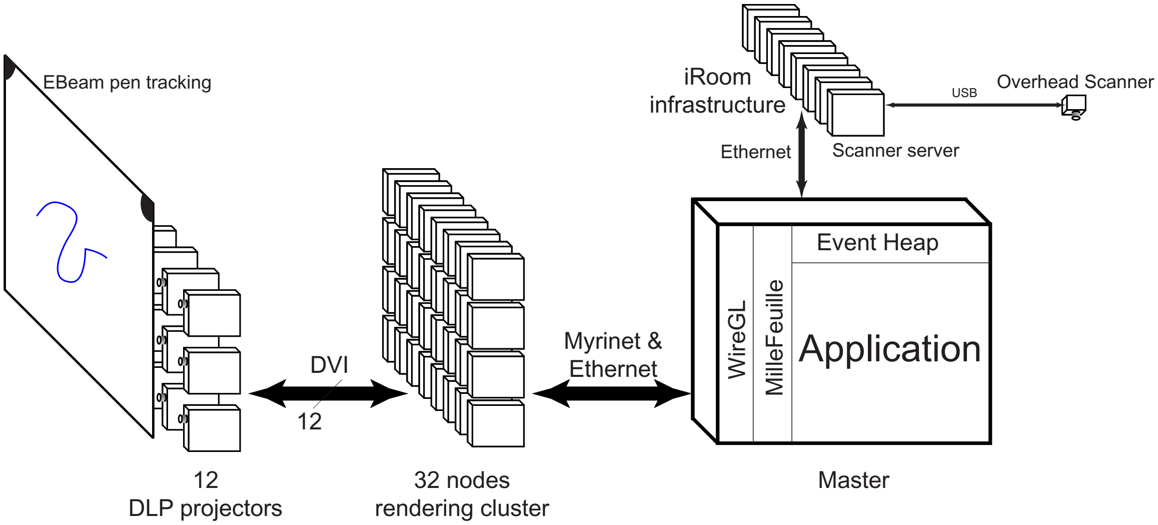


Figure 32: Overall system architecture. Left: The Mural is fitted with an eBeam pen tracking system connected to the mural master via a serial connection (not shown here). Center: The Mural itself is built out of an array of 12 projectors connected to 12 rendering nodes of a 32-node rendering cluster which uses Myrinet as a communication backbone. Right: The master is connected to the cluster through the Myrinet switch. The master is also connected to the room infrastructure. Through the infrastructure, the application can access the overhead scanner and exchange information with the rest of the iRoom infrastructure.

in [ISH98]. Using this API, several clients can render to the same OpenGL context and synchronize their OpenGL streams so that order-dependent operations such as transparency are ordered properly. In a typical configuration the master takes care of the light rendering tasks such as strokes, pictures, and interface widgets, while other cluster members share the rendering of complex 3D models (Figure 38 page 66). As Section 5.2.4 will explain, the MilleFeuille toolkit makes it easy for the developer to create a distributed-rendering user interface by synchronizing master and slaves and providing specific widgets as proxies for the rendering slaves.

The master is also part of the iRoom infrastructure, as shown in Figure 32. This allows users to drop information from any computer directly to the Mural. It also provides a simple interface between the mural and FlowScan, the overhead scanner we designed for the iRoom to support PostBrainstorm. Section 5.3 will present FlowScan in more detail.

5.2 MilleFeuille

When work on the Mural began, we knew that one of the major uses of large displays would include complex rendering, requiring a large amount of rendering power most probably provided by a rendering cluster running a distributed rendering library such as WireGL. WireGL, like OpenGL [Ope], merely provides a rendering context to the application, not windows or input management. MilleFeuille[1] came to life to fill this gap as an object-oriented user interface toolkit on top of WireGL. In many ways MilleFeuille is similar to other user interaction toolkits such as X11 [Nye95]. It provides a class hierarchy which programmers can use to handle interaction events received from the users and render objects in the frame buffer. Using MilleFeuille's containers and widgets, programmers can create complex graphical interfaces with minimal effort.

Combining transparent or translucent pieces of information is a common interaction pattern on nondigital wall displays, so one of our key goals for MilleFeuille was simplified implementation of stack metaphors. MilleFeuille provides an infrastructure to manage transparent layers (*feuilles* in French) and combine them together in stacks (*MilleFeuilles*). Members of such a stack can both display information and provide interaction mechanisms so that the user can easily change the behavior or look of a stack by pushing or popping new layers.

As explained in Section 3.3, one of the main limitations of today's digital whiteboards is their inherent real estate limitation. We addressed this problem with ZoomScape, a new interaction technique that lets the user manage a Focus+Context environment on the board. Because ZoomScape requires objects to be scaled dynamically as they move, it requires support from the underlying windows system to work properly. MilleFeuille provides this support and helps designers building ZoomScapes.

As the size of the screen increases, a more powerful rendering engine is needed to provide the interaction rate necessary for fluid interactions. As the use of large rendering clusters to provide such rendering power on tiled displays becomes more and more common, it becomes increasingly important for the user interface toolkit to help the application developers harness this power. MilleFeuille provides abstractions that let the application developer manage the transparency of layers whose content is rendered by different processors in the rendering cluster.

5.2.1 Basic components

Because we wanted to understand better how interaction and data visualization relate to each other and be able to change interaction modalities on the fly (on a per-user basis for example), MilleFeuille makes an explicit distinction between *Graphic* layers that render information on the screen and *Interaction* layers that handle interaction following the Model-View-Controller style [KP88]. Because transparency and layers of information play important roles in our system, MilleFeuille provides the *MilleFeuille* container, which can be used to stack different layers together. All the layers inside a MilleFeuille container always share the same screen position and size. The

[1] The French word MilleFeuille translates literally as “Thousand Layers”. It is also the name of a french pastry.

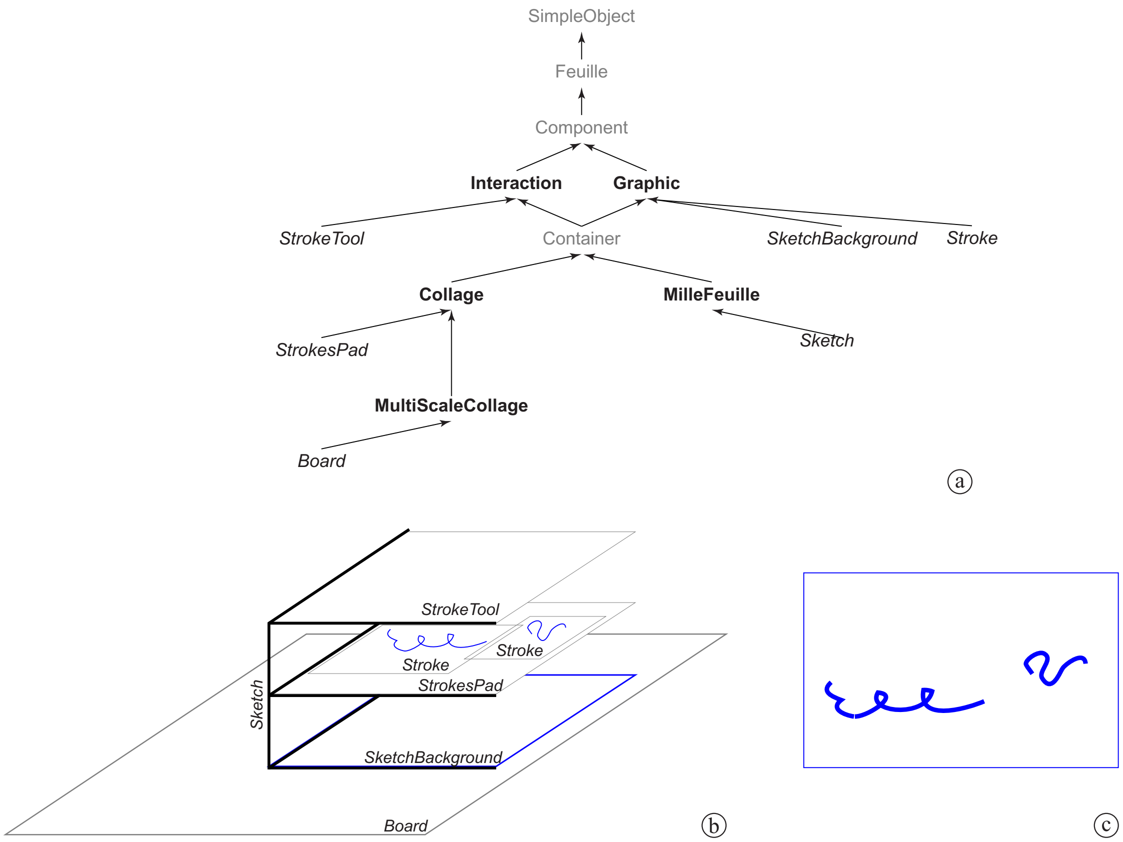


Figure 33: Basic MilleFeuille class hierarchy. Key classes explained in the text are shown in bold (**a**), while subclasses created to build a simple sketching application are shown in italic. Using these subclasses, one can create the structure shown in (**b**) which will shown on the screen as shown in (**c**).

MilleFeuille toolkit provides a standard canvas (called *Collage*) but we also extended the normal canvas semantic in *MultiScaleCollage,* which allows canvas elements to be scaled dynamically depending on their position. MultiScaleCollage is the super-class of all ZoomScape implementations. An overview of the MilleFeuille class hierarchy is shown in Figure 33a.

Using the objects shown in this hierarchy, one can build the simple drawing application depicted in Figure 33b. In this application, the Mural background is a MultiScaleCollage and the sketch consists of a MilleFeuille container stacking together a Graphic layer used to draw the sketch background, a SimpleSketch Graphic layer used to draw the strokes of the sketch, and an Interaction layer used to interpret user inputs and create new strokes as needed.

5.2.2 Event dispatch

Like other user interface toolkits, MilleFeuille dispatches two major type of events: graphic-related events and interaction-related events. But because MilleFeuille was designed to run on top of OpenGL in double buffer mode, uses scaling as a primary space management mechanism, and manages stacks of layers, it departs in significant ways from previous systems.

5.2.2.1 DrawEvent dispatch

Because MilleFeuille was designed on top of OpenGL and uses frame semantics, each frame must be redrawn from scratch. This simplifies the dispatch of graphic events: DrawEvent events are generated when the application calls for the method post_redisplay. DrawEvent events are then dispatched starting at the root Graphic layer and then propagated through the layer tree. Each subclass of Graphic layers implements the virtual function draw_action(DrawEvent &event) to handle this dispatch. Furthermore, since each frame must be redrawn from scratch, DrawEvent events do not carry any damaged region information.

As the traversal proceeds, the infrastructure updates the transformation matrix to reflect the current translation and scaling factor, so that layers can render their content directly without knowledge of the local transformation. Nevertheless, for some application such as semantic zooming [Per93] it is important for a layer to know the current scaling factor. For this reason the current scaling factor is propagated within each DrawEvent event.

5.2.2.1.1 Finalization code

Another peculiarity of our system is that the graphic context is only accessible to the dispatcher thread. This can be a problem when programmers rely on multiple threads to handle user requests in the background, thus hiding latency from the user. It is often the case that such a thread needs to access the graphic context to manage resources: for example if a large picture holder is deleted by a background thread, this thread will probably need to deallocate texture memory. If the object is no longer on the screen, the background thread cannot directly access the graphic context and deallocate resources. In that case, the client can register a finalization code to be executed by the graphic dispatcher thread at the end of the next frame. The code will be executed once and can perform any kind of OpenGL commands.

5.2.2.2 InteractionEvent dispatch

InteractionEvent dispatch in MilleFeuille is very similar to that of other toolkits. MilleFeuille maintains several input-device streams (one per connected device). The application can register one or more listeners to each of these streams. By default, the system registers the root window to all input devices present and the cursor layer to the pen input stream (by default there is a cursor in the system that tracks the movement of the pen at all times). When a new input is received from a stream, an *InteractionEvent* is dispatched first to the layer currently grabbing the focus (if there is one) then to any registered listener for that stream. To handle this dispatch, listeners should be of a

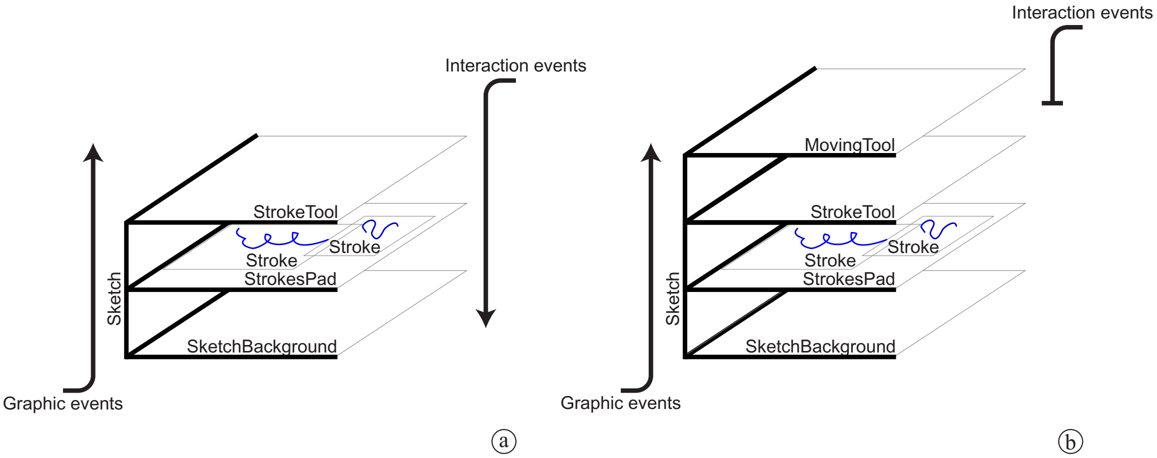


Figure 34: Event dispatch for a MilleFeuille container. Graphic events are dispatched from the bottom to the top of a MilleFeuille container, whereas interaction events are dispatched from top to bottom (**a**). This order makes it easy to temporarily change the behavior of a MilleFeuille by adding a new Interaction layer on top (**b**). In this case, adding a MovingTool layer lets the user move a sketch. The MovingTool intercepts the flow of pen events to let the stack follow the pen. At the end of the move interaction, the MovingTool automatically removes itself from the MilleFeuille stack, restoring the normal sketch behavior.

type which is a subclass of Interaction and implements the virtual function interaction_action(DispatchType type, InteractionEvent *event).

As the InteractionEvent is dispatched through the Interaction layers tree, the field describing the relative position of the event and the current scaling factor are updated. Although many layers do not need to know the scale at which they are rendered, pen interactions often perform a focus grab. As we explained above, in that case, the layers grabbing the focus receive their event before the tree is traversed. The event will then report only absolute screen coordinates with a scale factor of one. In that case it is important for a layer to know the scaling factor at the time of the grab, so that it can apply the correct scaling factor to each input coordinate.

5.2.2.3 MilleFeuille container dispatch

Because MilleFeuille containers organize information so that they behave as a stack of transparent layers, MilleFeuille containers handle dispatch of events in a slightly different way. DrawEvent events are dispatched from the bottom to the top of the stack so that layers will be rendered in a bottom to top order (Figure 34a), preserving transparency. This semantic can also be used to apply a local transformation to a subset of layers. To do so, one would first add a Graphic layer to a MilleFeuille container, pushing the current transformation matrix and applying the needed transformation to the current transformation. Next one would add to the container all the layers in the subset, and finally one would add a Graphic layer popping the current matrix.

From an interaction point of view, we wanted to be able to add a new Interaction layer on top of a stack to modify its behavior. For example, we might want to move the simple sketch shown in

Figure 33b by placing a “MovingTool” interaction layer on top of it. After the “MovingTool” is placed, it masks the previous interaction behaviors and the stack grabs the cursor and follows it as the user moves the pen around. When the “MovingTool” layer is removed, the sketch reverts to its normal behavior. To implement this behavior, MilleFeuille containers dispatch InteractionEvents from top to bottom. Events are dispatched in turn to each layer sub-classing Interactions until the bottom is reached. If one layer of the stack explicitly grabs the focus during the dispatch, the propagation stops at that layer.

Figure 34 illustrates this principle. In Figure 34a the stack behaves normally. Graphics events are dispatched from bottom to top; interaction events are dispatched from top to bottom. As interaction events pass through the stroke tool layer, a new stroke is created. If a “MovingTool” is placed on top of the stack while graphic events are still propagating through the stack, but all the interaction events will be intercepted by the “MovingTool” so that the stack can be moved around (Figure 34b). In this case, the MovingTool removes itself automatically from the stack when the pen is lifted from the screen.

5.2.2.4 Secondary interaction events

When interacting with an interactive surface, incoming events often need to be reinterpreted by the surface [Win01]. In MilleFeuille, this occurs, for example, in Quikwriting, the default character-entry mechanism. Strokes performed using Quikwriting are reinterpreted and generate character input to the underlying object. In such a case, it is important that the reinterpreted event be dispatched before any other events are processed: for example, if the user first draws on top of a text layer the Quikwriting stroke for *P* and then draws the command stroke to hide the layer, it is important that event order is preserved to assure that the text layer will receive the character *P* before being hidden. To handle this situation, MilleFeuille provides the application programmer with *secondary interaction events,* events that are created by the application during the reinterpretation process.

Secondary interaction events are posted by an application during the main interaction events dispatch loop and are always dispatched (using the normal dispatch mechanism) before any other event generated by the device streams is dispatched. In this example, each character is generated as soon as it is recognized and appears ordered correctly with respect to other event sources.

5.2.3 Support for ZoomScape

ZoomScape (Section 3.3.1, page 26) is a specialized form of canvas that allows each object it contains to be scaled dynamically depending on its location. This interaction metaphor requires

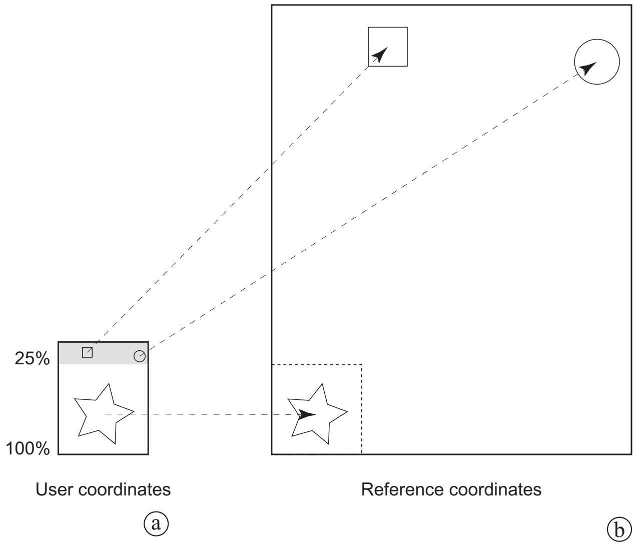


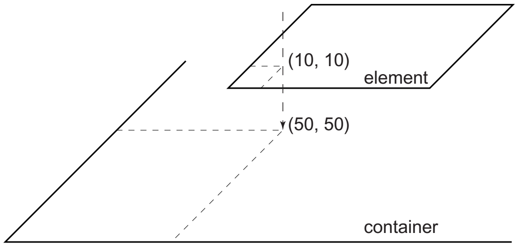
Figure 35: ZoomScape coordinate systems. Inside a MultiScale collage, each element can observe its size and position as if it were placed in a uniformly scaled window. While the user views configuration (**a**), the programmer views configuration (**b**), where the star is unchanged, but the circle and the square are now real size and placed as if they were part of a uniformly scaled window.

support from the windowing system to work. In MilleFeuille, *MultiScaleCollage,* a sub-class of Collage (Figure 33a), implements the ZoomScape semantics.

MultiScaleCollage looks quite different to the user than it does to the programmer. From the user's perspective, it appears as an implementation of a ZoomScape: objects are automatically scaled as they are moved around on the surface and groups distort according to the underlying zoom landscape.

From a programmer's perspective, MultiScaleCollage presents two coordinate systems. The first one is the *user coordinate system*. In the user coordinate system, each layer's scale varies depending on its position, and its position corresponds to the coordinate system perceived by the user. Layers contained in a MultiScaleCollage are rendered in this coordinate system. The second coordinate system is the *reference coordinate system*. In the reference coordinate system all layers share the same unit scale, and groups keep their shape as they are moved around. When layers contained in a MultiScaleCollage query their positions, they receive coordinates from this reference coordinate system. MultiScaleCollage provides a set of functions that makes it easy to switch between these two coordinate systems.

For static objects, it is easy to convert from one coordinate system to the other. Given an object O_1 located at coordinate (x_1, y_1) in the user coordinate system and shown at scale s_1, O_1 will be located at $(x_1/s_1, y_1/s_1)$ in the reference coordinate system. In the same way, an object O_2 whose local scale is$s_2$and whose location is (x_2, y_2) in the reference coordinate system, its



container.add_element(element, Point(10,10), Point(50, 50))

Figure 36: Using add_element. In MilleFeuille one can place an object on a ZoomScape surface such as MultiScaleCollage by specifying an anchor point on the object and a corresponding point on the canvas. This lets the user specify object position independent of local scaling factors.

location in the user coordinate system will be (x_2s_2, y_2s_2). Figure 35 shows an example of such a conversion. In Figure 35 we show both coordinate systems for a MultiScaleCollage, the lower section of which scales objects at 100% and the top section of which scales objects at 25%. While the left of the figure shows the configuration for the user, the right of the figure show the configuration for the programmer. Three objects were placed on the canvas by a user: a star in the center (in 100% area), a square at the top center, and a circle at the top right (both in a 25% area). When the program queries the star size and position, the answer corresponds to its size and position on the screen since the window is in a unit scale area. But when the program queries either the square or the circle size and position, the answer corresponds to four times its size on the screen and a position four times further away from the lower-left corner of the MultiScaleCollage as it appears on the screen since both are in a 25% scale area. The process is slightly more complicated for moving an object or group.

5.2.3.1 Moving objects

As objects are moved on a ZoomScape surface (such as MultiScaleCollage), they are dynamically rescaled around the dragging point of the pen tool. In that case, using the lower left corner of a window as a reference position no longer works because, as the object is scaled in and out, the distance between the anchor point and the lower left corner changes. Instead, in our system, the programmer can place an object on a canvas by specifying (in the user coordinate system) a point on the object and a point on the canvas that should be matched up, regardless of the scale of the two objects (Figure 36). Using this system, it is easy for the programmer to move objects around even when their size is dynamically updated.

5.2.3.2 Moving groups

As described in Section 3.3.1, page 26, when a user moves a group of objects on the canvas the objects are scaled according to the position of their center and the group is deformed to reflect the zoom landscape. As the group is moved around, the group geometry stays rigid in the reference coordinate system, while in the user coordinate system, the direction of each object relative to the center stays the same and the distance is updated dynamically.

To move a group with the anchor point A as a reference point, the application has first to recover the geometry of the group in the reference coordinate system at the beginning of the interaction. Then as the cursor moves, the application must recompute the group geometry in the user coordinate system using the precomputed geometry in the reference system and the current location of A, the anchor point, in the user coordinate system. To help in these steps, MultiScaleCollage provides two virtual functions: given two points in the user coordinate system, get_native_vector computes the corresponding vector in the reference coordinate system; and given a starting point on the user coordinate system and a vector in the reference coordinate system, get_collage_vector computes the resulting end point in the user coordinate system.

In the default implementation of MultiScaleCollage, general parametric representation of the form ax + by + c = 0 are used for performing the necessary computation. The integral (from Section 3.3.1, page 26)

$$D_{ref} = \int_{\overrightarrow{AO_i}} shrink(x, y)$$

then becomes:

$$(b \neq 0) \begin{cases} x_{ref} = \displaystyle\int_{x_A}^{x_O} shrink_x(a, b, c, x)dx \\ y_{ref} = \displaystyle\int_{\frac{(-ax_A + c)}{b}}^{\frac{(-ax_O + c)}{b}} shrink_y(a, b, c, x)dx \end{cases} \qquad (b = 0) \begin{cases} x_{ref} = 0 \\ y_{ref} = \displaystyle\int_{y_A}^{y_O} shrink_y(a, b, c, x)dx \end{cases}$$

The virtual functions x_scale_profile_primitive and y_scale_profile_primitive let subclasses provide an efficient implementation of the primitive functions needed to compute these integrals.

Similarly, as we move the anchor point A, we solve the following equation in O_i with the constraint that $\overrightarrow{AO_i}$ be co-linear with $\vec{u_i}$:

$$\int_{\overrightarrow{AO_i}} shrink(x, y) = D_{ref}$$

Using the same parametric representation the equations to solve become:

$$(b \neq 0) \begin{cases} \displaystyle\int_{x_A}^{x_O} shrink_x(a, b, c, x)dx = x_{ref} \\ \displaystyle\int_{\frac{(-ax_A + c)}{b}}^{\frac{(-ax_O + c)}{b}} shrink_y(a, b, c, x)dx = y_{ref} \end{cases} \qquad (b = 0) \begin{cases} x_{ref} = 0 \\ \displaystyle\int_{y_A}^{y_O} shrink_y(a, b, c, x)dx = y_{ref} \end{cases}$$

Using a standard Newton approximation method [Atk88], the virtual function x_scale_profile_primitive_inverse and y_scale_profile_primitive_inverse use functions x_scale_profile_primitive and y_scale_profile_primitive to solve these equations and compute the resulting vector in the user coordinate system.

This approach proved to work very well in practice. Its main drawback was that for an interaction starting with a group that span a transition area, the resulting group deformation is dependent on the grabbing point. To our knowledge users never complained about this problem.

5.2.3.3 Event dispatch

Event dispatch for MultiScaleCollage is more complicated than for a simple Collage since it is necessary to map the event coordinate from the user coordinate to the reference coordinate during dispatch. As a new event is dispatched, its position on the canvas is expressed in the user coordinate system, but the bounding boxes of layers on the canvas are expressed in the reference coordinate system. Given that MultiScaleCollage can have an arbitrary zoom landscape, we solved this problem by performing the conversion from user to reference coordinate system on a per-layer basis. In our system, each layer maintains its own scaling factor so that during event dispatch, before comparing an event position to a layer's bounding box, the event position is scaled by the layer's current scaling factor. In the case of a match, the user coordinate position of the event in the coordinate system of the target layer is computed by subtracting the layer position in the reference coordinate from the scaled event position. Using this simple method it is possible to interact with windows whose boundaries span across areas of non-uniform scaling factor. This means that one would get the correct dispatch even if a large window were moved to the top of the screen and part of the window were over a 25% area and part of the window were over a 100% area.

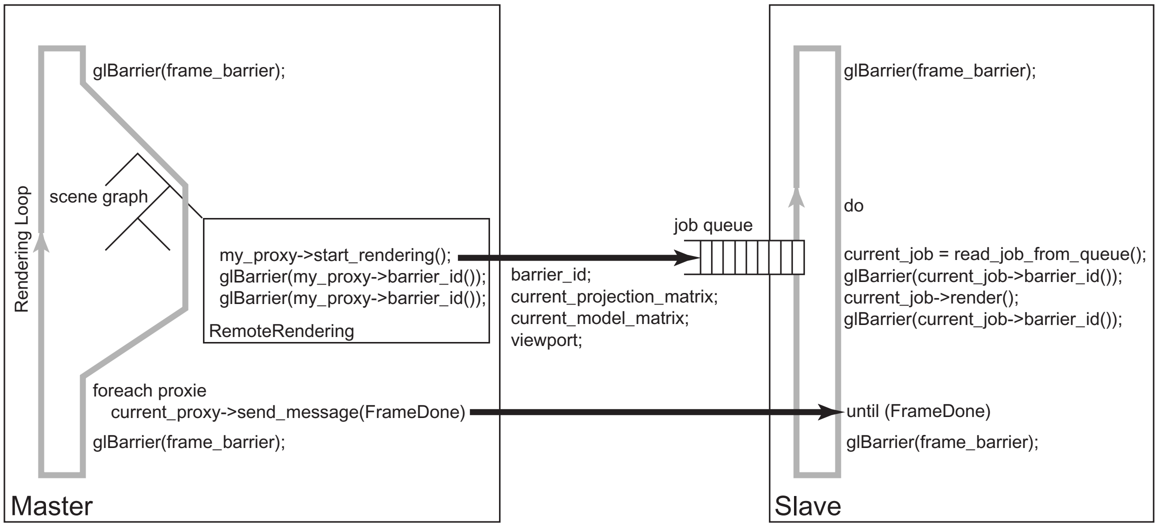

glBarrier(frame_barrier);
scene graph
Rendering Loop
my_proxy->start_rendering();
glBarrier(my_proxy->barrier_id());
glBarrier(my_proxy->barrier_id());
RemoteRendering
job queue
barrier_id;
current_projection_matrix;
current_model_matrix;
viewport;
foreach proxie
current_proxy->send_message(FrameDone)
glBarrier(frame_barrier);
Master
glBarrier(frame_barrier);
do
current_job = read_job_from_queue();
glBarrier(current_job->barrier_id());
current_job->render();
glBarrier(current_job->barrier_id());
until (FrameDone)
glBarrier(frame_barrier);
Slave

Figure 37: Master-Slave frame synchronisation in MilleFeuille. During each frame, the master and slave synchronize with each other using a combination of TCP/IP messages and graphics state synchronization. At the beginning of the frame the master and each slave call glBarrier to synchronize their graphic states. Then, during tree traversal, as a proxy is traversed, its updated rendering state (viewport, projection and model matrices) is sent to its corresponding slave. A proxy-specific barrier is also used between the proxy and the slave to guarantee correct rendering order of elements of the rendering tree. When the tree has been completely traversed the master sends a message to each slave indicating the end of the current frame so that all slaves can call glBarrier before buffers are swapped

5.2.3.4 Discussion

Our user study showed that ZoomScape provides a valuable response to the limited real estate of current digital whiteboards and helps people to manage and structure large amounts of data. As explained in Section 3.3.1, ZoomScape is not limited to a two region geometry but can accommodate arbitrary geometry such as the radial geometry shown in Figure 15b, page 28. This geometry would be well suited for a round digital table top. ZoomScape is currently limited to scaling but could easily be extended to accommodate other kinds of transformations, such as rotation. Rotation would be very useful in a digital table-top configuration such as the one just mentioned: using rotation, documents could reorient themselves as they were moved around the table, so that they would always be facing a user sitting at the table.

5.2.4 Support for distributed rendering

WireGL is a cluster rendering library. It presents to its client the abstraction of an unified display built out of several (in our case 12) rendering engines tiled together. It also provides a parallel API allowing several clients to render in parallel to the virtual screen and synchronize their rendering streams using FruGL [ISH98]. In our configuration, we designated one member of our cluster as the application master, which manages the dispatch of interaction events, as well as the user

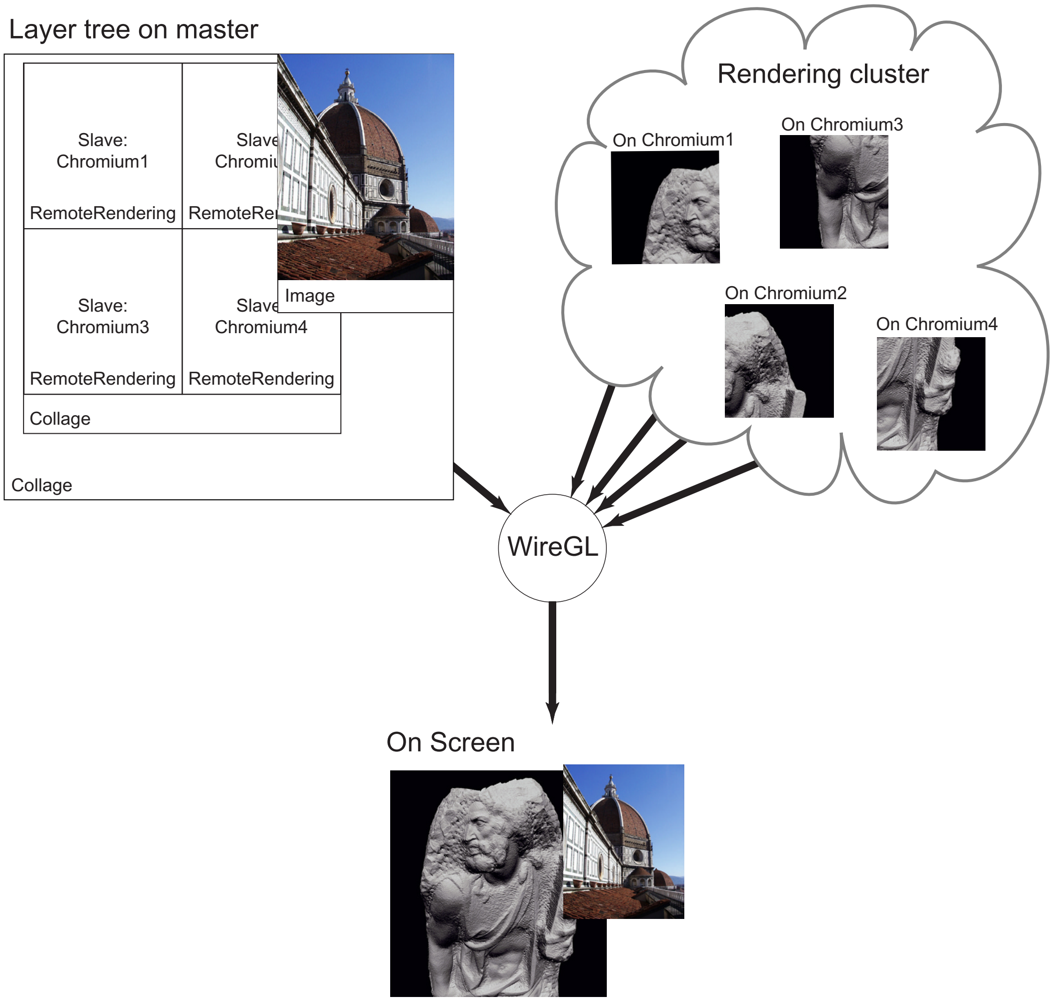


Figure 38: Using RemoteRendering layers to render complex 3D models. RemoteRendering layers make it easy to use several slaves to render one complex 3D model. Here, the rendering of the St. Matthew has been divided into a 2 x 2 tiling. By properly setting the viewport of each tile, each rendering slave only renders one quarter of the total model, and the complete image is properly reconstructed on the screen.

interface part of the application. As many as 32 slaves can be used to generate geometry for complex 3D rendering.

MilleFeuille exposes the distributed rendering API via the *RemoteRendering* layer so that applications using MilleFeuille can use the full rendering capability of the distributed graphic API provided by WireGL. A RemoteRendering layer behaves like a Graphic layer inside the rendering tree, but its graphics content is generated by one of the rendering slaves in the system. RemoteRendering layers can be added and removed from the tree dynamically, and each rendering slave can serve multiple RemoteRendering layers per frame, allowing for maximum flexibility.

At program start-up, after the initialization of the cluster is completed, the MilleFeuille dispatcher establishes a farm of rendering slaves. Slaves run on cluster members as daemons waiting for a startup message from the master. When the master starts, creating the farm of rendering slaves, MilleFeuille automatically synchronizes each slave in the farm with the master on a frame boundary. As shown in Figure 37, master and slaves use a combination of TCP/IP messages and glBarrier calls [ISH98] to synchronize their graphic state. A glBarrier is used at the beginning of a frame so that the clear buffer takes place correctly and at the end of a frame so that the buffer swap will take place only after all graphic calls for this frame have occurred. Since the load of each slave can change from frame to frame, when the tree traversal is completed we use a TCP/IP message to notify each rendering slave that the frame is completed. At this point all can run the outermost barrier safely without the risk of entering a deadlock.

As a RemoteRendering layer is added to the rendering tree, its job is added to the rendering farm: a slave and a unique glBarrier ID are assigned to the job, and the job description and barrier ID are sent to the selected slave. When traversed during rendering (Figure 37), each RemoteRendering layer sends to its slave an updated projection and transform matrix for the current frame and uses a glBarrier to guarantee the correct rendering order for the components of the tree. Upon receiving the new rendering parameter the slave proceeds with rendering using the same barrier for graphic synchronization.

Using RemoteRendering layer, it is easy to divide the rendering load of complex 3D models between several slaves by adjusting the viewport and projection matrix of each RemoteRendering layer to create a tiling. In Figure 38, each slave uses QSplat [RL00] as its rendering engine, and uses its own local copy of the full data set. We created a 2 x 2 tiling configuration by using an array of four RemoteRendering layers tiled together. While the four RemoteRendering layers share the same model matrix, they each set their viewport and projection matrix so that each tile renders only one quarter of the full model. More complex balancing schemas can be used if necessary, but in practice this approach was a good balance between complexity and efficiency.

5.2.5 Difficulty with the frame semantics

From the start, our system enforced a frame semantics: each time something changes on the screen, the full screen must be redrawn. This configuration is the typical for high-end interactive rendering systems and made sense for us, since we envisioned that applications for the Mural will use interactive 3D rendering and transparency extensively. Initially, this choice was not a problem since rendering loads were light and very high frame rate could be achieved. But as the complexity of material we showed on the screen increased, the frame rate dropped making the system less reactive and more difficult to use. The main problem was the lag time created by a low frame rate. From our experience, a frame rate of 10 frames per second (f/s) was acceptable to users when they were manipulating 3D objects on the screen [RCM89], but it created an unacceptable lag when drawing on the screen with a pen.

To solve this problem, we allowed interaction methods to draw on the *front* buffer during the interaction dispatch. That way, as a stroke is drawn on the screen, it can be updated right away without swapping buffers and causing a complete redraw. At the end of the stroke, the full image is regenerated as usual. To further improve quality, stroke points provided by the pen are buffered by the receiving thread and delivered to the applications as a list of pending events. This feature minimizes the number of redraws and prevents the application from falling behind: if for some reason the frame rate drops, the returned list gets longer, but the application does not have to generate a redraw for each new point in the list. Instead it processes them in batches. When combined, front-buffer rendering and stroke-point buffering reduced the gap between the pen tip and the digital ink to less than 0.4 inches while writing, an acceptable level for most users.

Nevertheless it is clear that the current frame semantics used in our system will not scale well. Because of the size of the screen, most of the screen content is often outside of the user locus of attention and will not change from one frame to the other. As more and more complex content is rendered on the screen, more and more time is spent rerendering content that does not change from frame to frame. During the development of our system, we experimented with several remedies to this problem. Our first solution was to use a governor [RCM89] to redirect the rendering power at the locus of interaction. In this approach the system evaluates the current locus of attention and assigns most of the rendering power to layers in this area. As a result, high-quality rendering is provided at the locus of attention, and low-fidelity rendering is provided elsewhere on the screen.

This approach has been very successful in workstation systems [RCM89], but does not work well on large displays. First, when more than one person is looking at the display, each person's focus of attention can be quite different and the different levels of rendering will be noticed. Second, the change in rendering quality creates small changes in the peripheral vision of the user, which can be very distracting. For these reasons, we did not pursue this approach.

Another solution would be to cache the 3D images that are not modified, and use the result as a texture map in subsequent frames [LS97]. This approach is currently feasible on hardware that provides a fast path between the buffer memory and the texture memory. Unfortunately, using the current implementation of WireGL, implementing this schema would require us to transmit the texture data from the rendering computer to the master and back. This cost was judged unacceptable, especially since textures are broadcasted. The new version of WireGL, called Chromium, lets

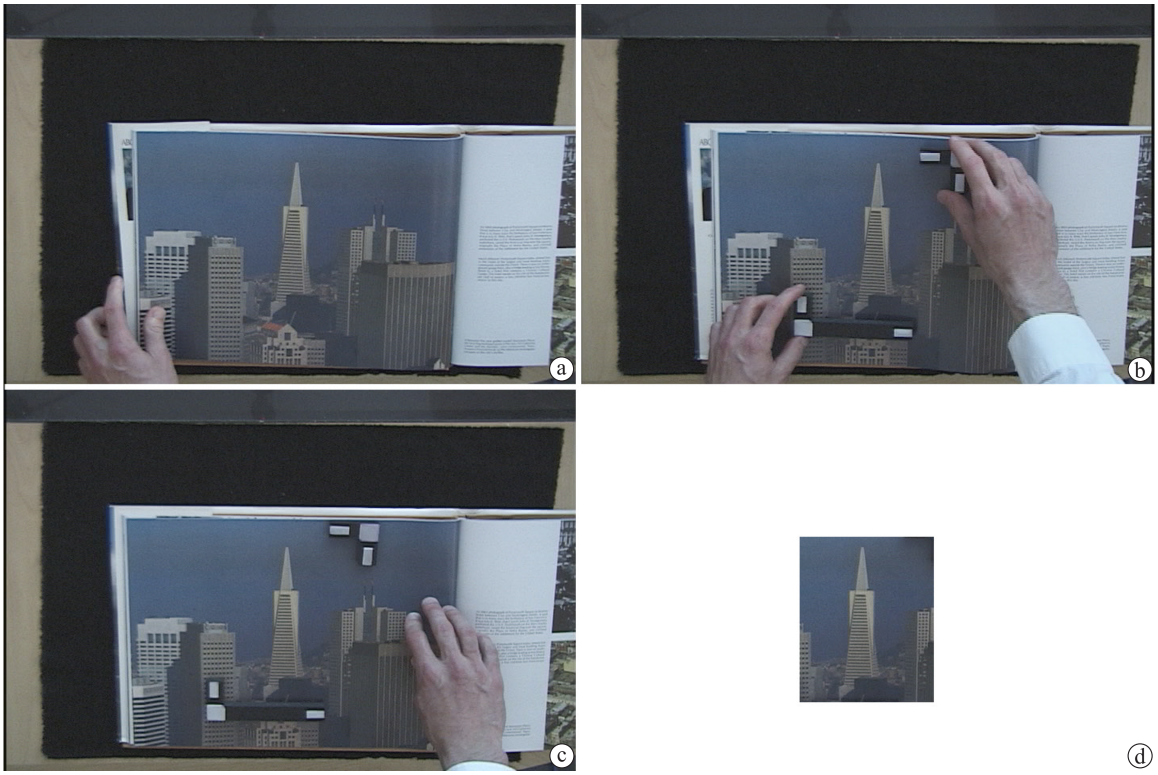


Figure 39: Using the FlowScan tangible interface. After laying the document on the active scanning area (**a**), the user uses two kinds of crop marks to indicate the area of interest and its orientation (**b**). After triggering the scanner (from the Mural or the scanning station) (**c**), the cropped picture appears on the screen (**d**).

programmers perform this transfer locally on the server side and makes caching the static part of the screen using a texture map a viable alternative to improve overall rendering speed.

5.3 FlowScan

Even in our increasingly digital world, people often bring printed documents to meetings, or sketch visual information on pieces of paper during meetings. Very early in our project, we knew that it would be important to provide a simple way to bring non-digital information from the physical world into the digital environment of the mural for meetings. An obvious approach was to provide a flatbed scanner. Yet this solution has a major drawback: flatbed scanners are somewhat difficult to use and require a lot of attention from the user. This fits poorly with a typical meeting situation, where the important point is to move the piece of information to the wall as rapidly as possible without falling behind in the meeting as a price of operating the scanner. Instead we designed *FlowScan,* an overhead scanner with a tangible interface on the meeting table, where a user can easily designate the area of interest. To use the scanner (Figure 39), the user first places

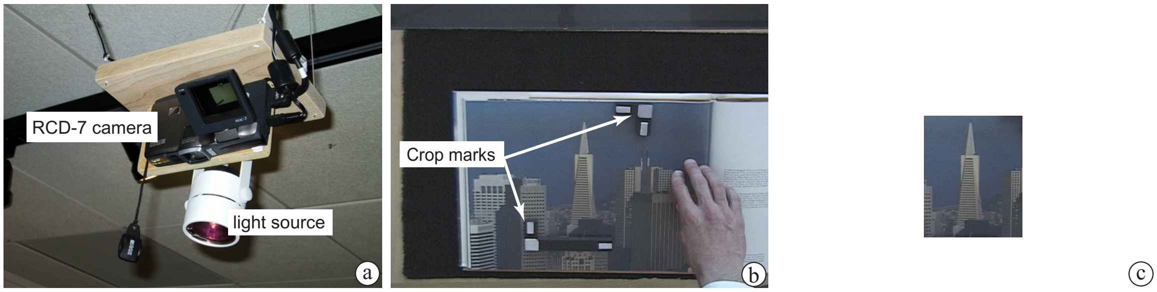


Figure 40: The FlowScan overhead scanner configuration. A Ricoh RDC-7 camera is hung from the ceiling and controlled through its USB interface. A light source is placed close to the camera to illuminate the retro-reflectors (**a**). After taking the picture, the scanner software identifies the retro-reflector positions (**b**) and produces a cropped and rotated image (**c**).

the object (or book or sketch) within the active scanning area and indicates the area of interest by using special crop marks. She then trigger the camera from either the scanning station or the Mural.

We used an off-the-shelf digital camera (Ricoh RDC-7) hung from the ceiling (Figure 40) and controlled by a computer. The main difficulty of this project was to reliably detect the crop marks. We first experimented with active crop marks containing LEDs, but managing the battery or providing a reliable way to switch on the LEDs when the crop marks were needed proved difficult. So we switched to using retro-reflecting tags placed on top of crop marks made out of foamcore (Figure 40b). To insure better detection of the retro-reflective tags, we placed a small source of light close to the camera (Figure 40a) so that the tag provides a strong signature without modifying the appearance of the picture (even when glossy materials were used).

Each tag is identified by thresholding the picture and using a simple region-growth algorithm [GW92] to identify possible candidates. Possible candidates are then grouped by size and the crop-mark geometry is used to discriminate further. The system uses two kinds of crop marks, one of which is always used to specify the lower-left corner of the scan. Upon request the camera takes a picture of the active area, identifies each crop mark's retro-reflector markers, crops and rotates the area delimited by the crop marks, and delivers the picture to the display surface specified by the user.

Preliminary testing showed that 100 dpi was the minimum resolution acceptable to render sketches. We designed the system so that our 3.3 Mpixel camera (2047x1535) captured a 20-x15 inch area, scanning this area at 100 dpi.

5.3.1 Discussion

FlowScan and its tangible interface proved to be extremely effective and popular. FlowScan provided a simple and direct way for the user to post new pieces of information onto the display. By eliminating the dialog between the scanner and its user, FlowScan also preserved the group dynamic in meetings.

Yet the current implementation has some limitations. The capture area is too limited, and because we are using an off-the-shelf camera, the cycle time between scans is large (around 10 seconds). In effect these limitations allow only one user to take one picture every 10 seconds. At the beginning of the meeting, this limitation prevents users from “priming” the board by rapidly scanning a stack of documents. During the meeting it limits the use of the scanner because whoever wants to scan a document needs first to interact with the person sitting in front of the scanning area.

These problems are not related to the interface per se but to the specifics of our implementation. We believe that inexpensive imaging elements such as CMOS sensors [Fos98], will soon make it possible to create large arrays of sensors placed on the ceiling that are capable of capturing a full tabletop. Using the same FlowScan interface, several users will be able to scan different areas of the tabletop simultaneously, or to scan elements of a stack of document in rapid succession.

Chapter 6
Applications

To validate the concepts presented in the previous chapters, we implemented two applications, each illustrating a possible use of our display. The first application, the *Geometer's Workbench,* illustrates how one can use a digital backboard to explore differential geometry. The Geometer's Workbench aims to provide the ease of use and flexibility of a blackboard while providing the computational resource of an algebraic engine such as Mathematica [Wol96]. As in many other scientific applications, the lag caused by computation time was an issue. The Geometer's Workbench illustrates how latency can be dealt with so that it does not prevent interactive exploration of a complex parameter space.

The second application, PostBrainstorm, is a new kind of brainstorming tool that illustrates how the display can be used by small groups of designers to gather and structure a wide variety of information. Brainstorms are very dynamic and engaging meetings, and well-trained brainstorm leaders are extremely efficient at managing the flow of information using walls and simple technologies such as Post-It™ notes and pens. The brainstorming process seems to be the perfect test ground for the feasibility of a fluid interface. First, the group dynamic is extremely important for the success of a brainstorm, so even a low level of disruption caused by the interface can greatly diminish results. Furthermore, brainstorms could profit greatly from a fully digital medium. Even when the paper-and-pen approach works very well during the brainstorming session itself, it causes serious difficulty when the leader has to distill the brainstorm and create a coherent report: pieces of

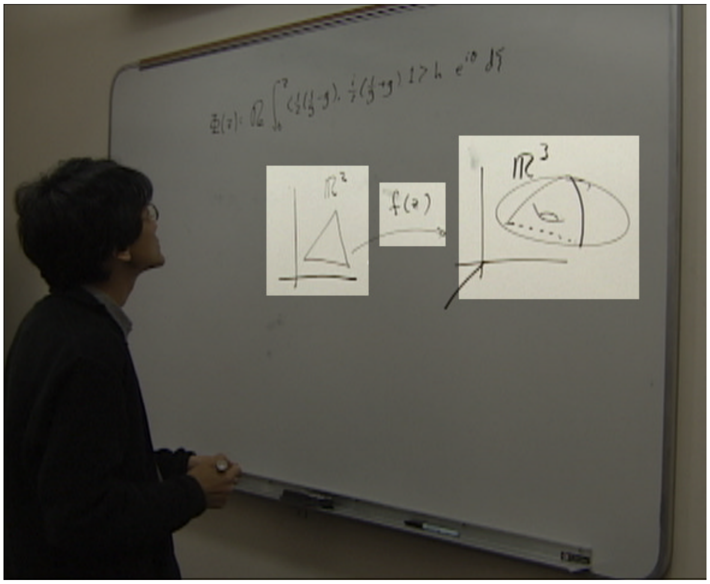


Figure 41: A whiteboard showing a common graphical idiom used by mathematicians. From left to right, highlighted regions are: a 2D patch to be mapped (the domain), a mapping definition, and the result of the mapping.

information created on paper or wall must now to be transferred manually (typed in or drawn into a drawing program) to the digital world at great cost of time and effort.

6.1 Geometer’s Workbench

Mathematicians working with differential geometry problems have two forms of tools: they can use a blackboard for casual problem exploration, or they can use an analytical engine such as Mathematica. Using the former, they can fluidly mix sketches, diagrams, formulas, and fragments but have no computational power at their disposal. Using the latter they produce precise results but are forced to deal with a far more rigid interaction style. The Geometer’s Workbench project was an attempt to bridge the gap between these two tools. The Geometer's Workbench was designed to encourage exploration and casual interaction while acting as a new front-end to the Mathematica kernel using the Interactive Mural. One of the prime advantages of an active display over a physical blackboard is the ability to do interactive “steering” in a simulation space. Here, high latency is the rule rather than the exception since the user typically pushes the envelope of the simulation capability. Providing fluid interaction in this context was one of the main goals of this project.

For this prototype, we put aside the formula recognition problem and limited the scope of our investigation to geometrical mappings between differential manifolds. Rather than entering a symbolic formula, users choose a mapping function by picking a parametric family (from a library that

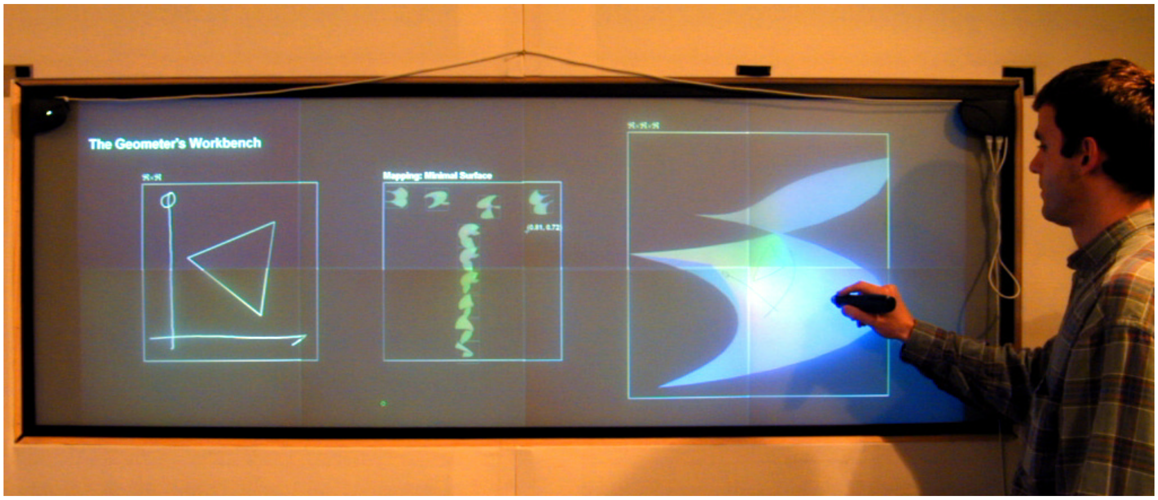


Figure 42: Overview of the Geometer's WorkBench. The screen is divided into three sections (from left to right): the 2D patch on which the user can draw arbitrary sketch, the MultiPoint area used to select a mapping, and the 3D model which shows the resulting mapping of the sketch. The system is shown here running on the previous version of the mural.

includes sphere, saddle, and minimal surface) and specifying the values of two parameters. They then can explore how arbitrary sketches drawn on the board will be mapped by the function onto the specified 3D manifold. This work was conducted in collaboration with Sha Xin Wei.

6.1.1 Blackboard interface

The design of the Geometer's Workbench interface was inspired by the graphical idiom often used by mathematicians studying differential geometry (Figure 41). From left to right, the mathematician describes a 2D patch to be mapped (the domain), the mapping to use, and the result of the mapping. Because of the size and resolution of the Mural screen, we were able to provide a very similar setting, displaying contextual information along with work focus, as shown in Figure 42. At the left of the screen is a representation of the 2D patch to be mapped (the mapping domain) (Figure 42 left). In the center, we provide a graphical way to specify the desired mapping using the MultiPoint area (Figure 42 center). On the right is a 3D model generated from the information supplied in the 2D patch and the MultiPoint areas (Figure 42 right).

6.1.1.1 The 2D patch

The 2D patch represents the mapping domain (Figure 43 left). The user can draw arbitrary freehand shapes or select among a menu of simple shapes (such as circles, equilateral triangles, and rectangles) that can be dynamically adjusted in size and orientation. In our experience, users prefer to create the shape they need by sketching, rather than using predefined shapes. If the user has

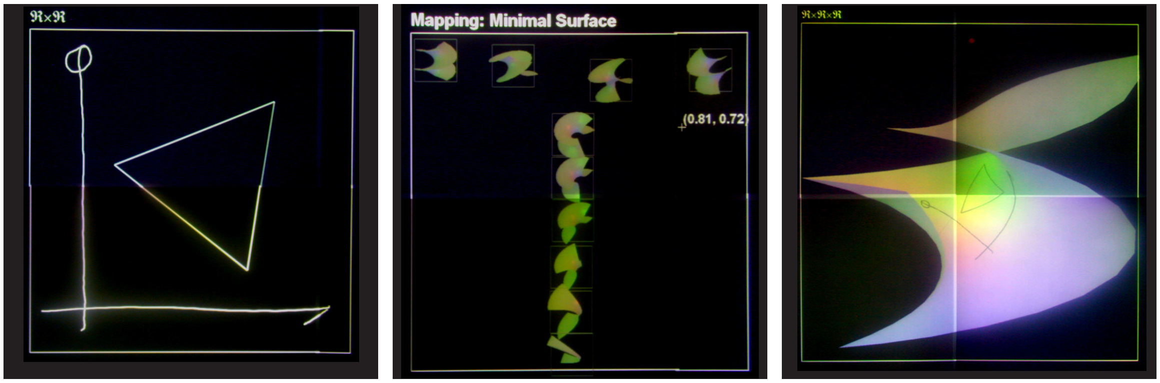


Figure 43: Close-up of the different areas on the Geometer's Workbench board. **Left**: the 2D patch used to draw sketches. Using the menu, the user can also draw a single geometrical shape such as the equilateral triangle shown here. **Center**: the MultiPoint area is an explicit representation of the underlying parameter space. The parametric family in this example is Minimal Surface. Pointing in the area at (s, t) will show the resulting mapping for this pair of parameter values. In this case s represents the twist of the surface and t represents how much of the surface is used. The values selected for the current mapping are shown by the cross hair with the corresponding value attached $(0.81{,}0.71)$. Because samples stay visible, the user can easily compare different areas of the parametric space even in presence of high latency. **Right**: the 3D model area shows the result of the current mapping. The 3D object can be manipulated using the pen as a virtual trackball.

already chosen points in MultiPoint, their strokes are automatically mapped onto the selected surface in the 3D model area.

6.1.1.2 Mapping selection

The middle segment (Figure 43 center) is used to select the current mapping using MultiPoint (see Section 3.4.1, page 29) rather than a hand-written equation. Each mapping is defined by a family and two parameters (s, t) that are specific to that family. Once a family is chosen, MultiPoint provides an explicit representation of the underlying 2D parameter space. For example, in the MultiPoint area of Figure 43, the family "Minimal Surface" was chosen from a menu, and then the user clicked on a dozen or so points (s, t) on the screen to discover the resulting mapping for each pair of parameter values. For this family, the horizontal axis s represents a variation in twist and the vertical axis t represents the amount of the surface shown. Instead of going point by point, the user can also draw a stroke and the system will resample it appropriately, leaving behind a trail of non-overlapping sample images. When the user is satisfied with the exploration, she can use the menu to specify any point in the parameter space as the value to be used for mapping. If she selects a new parameter, the 3D model view window at the right of the screen is updated and the strokes present on the 2D patch are automatically remapped to the new surface in the 3D model view window.

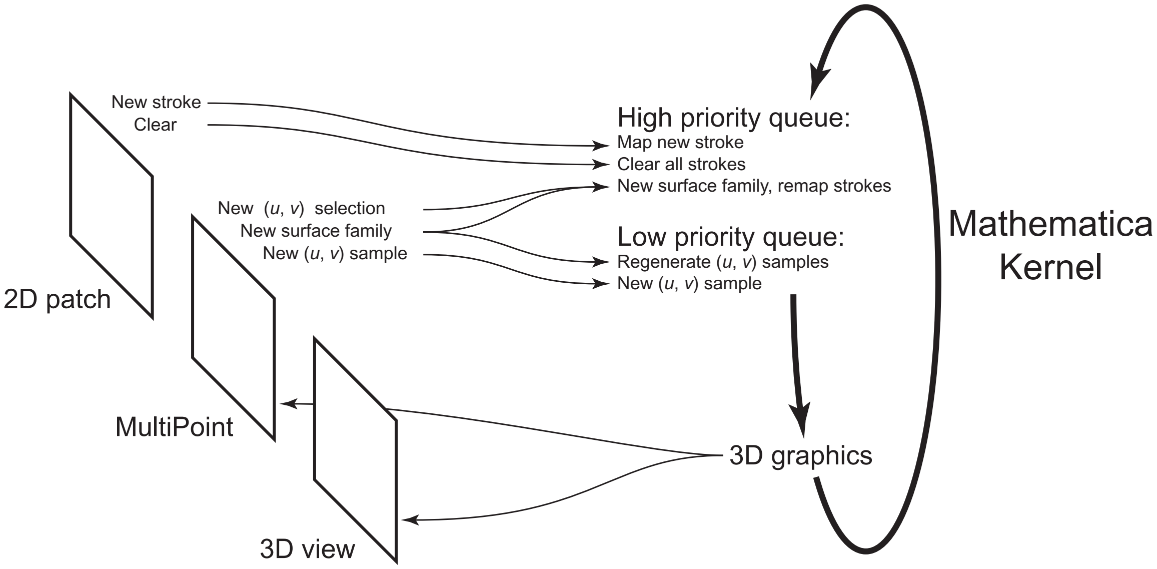


Figure 44: Geometer's Workbench system architecture. To handle computation latency, the application provides a proxy as immediate feedback while posting requests to the Mathematica kernel. Requests maintaining the coherence of the screen are posted to a high-priority queue; sample requests are posted to the low-priority queue. The kernel processes requests asynchronously and sends the resulting graphics back to the front end for display.

6.1.1.3 3D model

The right segment (Figure 42 right) shows the shaded three-dimensional model of the surface currently selected in MultiPoint with the shape in the 2D patch mapped onto it. The pen tool can interactively rotate this model in three dimensions, allowing the user to see all parts of the curve and get a better sense of how it maps onto the surface.

6.1.2 System architecture

One of our key goals for this system was to create a fluid interface to the Mathematica kernel. We reached this goal by carefully designing our system architecture so that latency could be kept out of the main interaction loop. To avoid blocking behavior we decoupled interaction and computation: whenever the user interacts with the system, the system provides simple immediate feedback before forwarding the requests to the computational engine. Requests are queued in two priority queues, as shown in Figure 44. The high-priority queue is used for requests that will help restore a coherent screen to the user: for example, when a new family is selected the 3D view is updated and all the strokes of the 2D patch are remapped to the new surface. The low-priority queue is for those requests that establish the exploration context (MultiPoint interactions). Results are displayed as soon as they become available.

In Geometer's Workbench communication with Mathematica is handled by the standard MathLink protocol [Wol93]. The link is used to send requests to the kernel as well as to retrieve Mathematica's Graphic3D descriptions. Graphic3D descriptions are transformed to OpenGL calls using the MathView3D library [Kus]. We modified this library extensively to accommodate multiple pending contexts and improve network performance.

6.1.3 Discussion

One of the major promises of the large high-resolution display wall is the visualization of complex data sets. In the recent year advances in cluster rendering have increased the rendering capability of large displays manyfold, making it possible to use them not only for data visualization but also for computational steering of the simulation producing the data set under investigation. The Geometer's Workbench provides a glimpse of what driving such a system can feel like. By retaining the casual interface of a blackboard and using latency-safe interaction with MultiPoint, the Geometer's Workbench provide an environment favorable to exploration and analysis of "What if..." scenarios. Geometer's Workbench only scratches the surface of how one can design tools that provide a powerful computational resource in an informal exploration-friendly interface. We believe that by providing powerful simulation resources early in the design process, these tools can play an increasingly important role in the near future.

6.2 PostBrainstorm

PostBrainstorm is a fully digital brainstorming tool. In designing PostBrainstorm, we wanted to deliver an experience as fluid as current brainstorming experiences using Post-It™ notes and pen can be, while providing an easy way to access and collect digital and non-digital material onto the display. By getting all the gathered information into fully digital form, PostBrainstorm helps designers during the post-meeting report phase (hence its name). Using a fully digital system it becomes easy to transfer the information that was on the Mural to a document that can be reshaped to create the final report for the meeting. Using the log of actions performed on the board, it is also easy to call back a previous meeting or send the content of a meeting to another location. PostBrainstorm is well suited to the style of brainstorming advocated by IDEO [KL01], where strict social rules are observed by the group and the computer serves as a medium to record the group's thought process but does not take part in the process itself.

We will begin by presenting the results of our study of how brainstorming sessions are carried out at IDEO, then point out the major breakdowns in the current practice before presenting Post-Brainstorm and how it addresses these breakdowns.

6.2.1 The brainstorming process

To better understand the requirements for a digital brainstorming tool, we interviewed designers and conducted observations at IDEO, an industrial product design firm in Palo Alto [KL01]. We conducted further interviews with other firms, such as Speck Design, to validate our findings.

6.2.1.1 Interview with IDEO

We had several discussions with David Kelley, IDEO co-founder, over the course of this project. Most of the discussions focused on the brainstorming technique used at IDEO. Here we present a synthesis of these discussions.

6.2.1.1.1 Use

Brainstorming is intrinsic to IDEO design culture. It is used throughout the design process to generate the stream of new ideas necessary to address challenges at each phase in the development of a new product.

6.2.1.1.2 Technique

Brainstorming groups at IDEO vary in size from small (4 to 5 persons) to large (20 persons or more). Large groups are often used at the beginning of a project and usually break into small subgroups as the session progresses. During the session the tools of choice are a large whiteboard, large Post-It™ notes, and pens. Each participant has access to paper and pen to sketch or jot down ideas they want to keep in mind. The leader (or sometimes two leaders for a large group) commits ideas to large Post-It™ notes that she places on the wall.

Depending on the topic a lot of outside material may be brought in by the brainstorming leader: this can include photos, printouts, and often objects or material to inspire new ideas. IDEO maintains a repository of interesting materials in a large tool chest called the “Tech Box” [KL01, page 143]; material can be drawn from it as needed for each meeting.

Meetings are generally split into two distinct phases. The first phase emphasizes idea generation. During this phase the main goal is to produce as many ideas as possible. One-hour meetings creating 100 or even 150 ideas are common. This phase is in general very dynamic and lively but nevertheless, the group follows a strict set of rules designed to encourage creativity [KL01]:

- One conversation at a time,
- Stay focused,

- Encourage wild ideas,
- Defer judgment,
- Build upon ideas from others.

The second phase emphasizes idea selection. During this phase, ideas are sorted out so that fewer than 10 are selected for further investigation. The selection process relies on discussion between the participants. It often takes the form of a vote during which each specialty (mechanical design, human factors...) is assigned a color and group members vote according to their core competence using small multi-color Post-It™.

The end of the meeting does not really end the brainstorm for the leader. After the crowd leaves the room, the leader must gather the pieces of information necessary to document the brainstorm. In general, all the Post-It™ notes are collected for archiving, and the selected ideas are set aside. During this process, the leader often redraws these ideas to incorporate the latest improvements to the original idea. Leaders also gather pieces of information that help document how an idea came into existence, as well as some rejected ideas that seem interesting to document.

On longer-term projects, a whole room is sometimes allocated to a project, becoming the project room. The project room can be seen as a brainstorming session in progress: it represents a repository of the information gathered about the project in one setting so that one can get the global picture of the project. In a project room, large pieces of foamcore are used as panels to pin material on (as an example, see Figure 3, page 3). Sketches, Post-It™ notes, photographs, and overlays on tracing paper are common. This type of room is of great value and can promote interaction with remotely located team members: the remote part of the team can fly in from time to time to be immersed into the room. The room can then be used as a 3D map of the project during conversations (most commonly conference calls) that include team members inside the room and team members at remote locations. Members can then use a spatial location (such as “about eye level on the wall next to the door”) to describe a piece of information. Having a virtual copy of this room at remote locations would be of great help for large, multi-month projects.

6.2.1.1.3 Wish list

IDEO had several experiences with digital whiteboards in the past, few of them successful. The culprits were the limited real estate on this display and its incapacity to incorporate nondigital information.

Despite these bad experiences IDEO is still looking for a computer-assisted way to carry out brainstorming sessions. For IDEO, moving to a digital platform opens the possibility of collecting more information about the design process and how the meetings unfold. This information could be used later on to better understand how a new idea came to life, to observe the reaction of a client at a key point in the meeting, or simply to provide a readily available source of material for the leader while she is preparing her report.

An ideal system should include not only the strokes and documents created during the meeting, but also an audio and video recording of the proceedings (as in the Recall system [Yen00]). At the core of the system would be a browsing tool that makes it easy for a user to query a meeting log with a request such as: “Go to the frame where Bob said ‘Make it purple.’ ”

6.2.1.2 Interview with Speck Product Design

Speck Product Design is another industrial design studio in Palo Alto. We interviewed David Law, one of Speck’s principals. As with IDEO, the goal of our interview was to identify the current brainstorming practice at Speck.

6.2.1.2.1 Use

Brainstorming is a very important part of the design process at Speck; it is not only used as an idea generating process but also as a confirmation tool to validate design.

6.2.1.2.2 Technique

Brainstorming sessions are performed in small groups of four or so people, producing around 100 ideas in an hour. The technique is similar to the one used at IDEO but at Speck, words are often preferred to sketches because they are easier to capture on the board and are more open to interpretation—leaving the potential meaning open can help generate more ideas. Speck most often uses a whiteboard augmented with a Mimio, a digital whiteboard capture device [Mim].

During its brainstorming sessions, Speck put a lot of emphasis on maintaining the flow of the meeting. The role of the leader is essential in that respect. A good leader should be able to understand, capture, and organize ideas rapidly so that he does not interrupt the flow of the group. It is another reason that words are preferable to sketches: because they can be put down more rapidly. The leader also should assure that the different participants stay in the same track and do not drift in and out of the flow of the meeting. These transitions are disruptive and limit the idea generation of the group.

6.2.1.2.3 Wish list

Bringing digital media such as CAD drawing into the brainstorming process would be a plus for Speck. But probably the most valuable feature would be the capability to record the session. While video is not so valuable because of the time it takes to process the footage, synchronized recording of strokes and audio using a system such as Recall [Yen00] would be best.

6.2.1.3 Observation

After our interviews at IDEO and Speck, we participated in several brainstorming sessions at IDEO. Our goal was to observe how people use wall surfaces during a typical brainstorming session. Because of intellectual property issues, we got a somewhat limited exposure; we observed three settings: a large group of more than 15 people, a small group of five persons or so, and a project room dedicated to an ongoing project.

For each session we shared the room with the brainstorming team. After being introduced by the leader and given the opportunity to describe our research goal to the attendees, we were encouraged to take an active part in the meeting. We recorded our observations by taking notes during the meeting and during discussions with leaders at the end of the sessions.

6.2.1.3.1 Large group session

The large group session involved 15 or so persons on a very open topic. During the first phase of the meeting the two leaders were collecting the ideas of the whole group. Both leaders were working in parallel using their own writing surfaces, yet they collectively enforced the five rules described in Section 6.2.1.1.2. Participants sat in a circle around the leaders.

As high-level directions started to emerge, leaders decided to split the group in two. Each group was assigned an area of the room and a topic and started its own brainstorm on this topic. Our group was sitting at a table with paper and pen in front of us. After selecting a leader, we started our own brainstorming. Again the leader collected ideas on a large easel using large Post-It™ notes. Most of the ideas were described using text but often small sketches were use to describe some detail. From time to time the participants used the paper in front of them to sketch an idea while the leader was busy at the board. The sketch was used later on to explain the idea to the leader (and the rest of the group), letting the leader commit it to the easel. When each Post-It™ note was full, it was stuck on the wall, first in front of the participants then later behind them. As a few major ideas emerged the leader decided to split the group again for a short period of time to generate more concrete ideas for each major idea. Finally a short wrap-up session took place with all the ideas assembled in front of the full group.

A large quantity of information was collected during this brainstorming session and we spent time with the designer after the meeting to ask them what they would do with it. Once the brainstorming meeting was over, the first job of the leader was to identify the 10 or so main ideas that will be explored further. She also identified ideas that were not selected for one reason or another but that may be interesting in the future as well as any important issues that came up in the discussions. In all cases, the information was then collected on a pad or cut away from the Post-It™

notes. Sometimes idea were sketched again to get a better rendering of the final idea under consideration. Finally all the Post-It™ notes were archived either physically or via scanning. Back at her desk the leader often needed to create a report to summarize the findings and present them to the client. This report would incorporate elements from the meeting—sometimes as is, sometimes after regenerating them.

6.2.1.3.2 Small group session

The small group session was conducted with five or so people on a narrower topic. The meeting took place in a small conference room, one wall of which was covered with a large whiteboard. During this brainstorm, the leader numbered in sequence each idea and structured the information in several vertical lists using bullets and sub-bullets. The debriefing process was similar to that of the large session.

6.2.1.3.3 Project room

Our last observation was not a brainstorming session per se, but a visit to a project room used for an ongoing project. The walls of the project room were covered with large foamcore panels on which a wide variety of pieces of information was pinned. Information ranged from printouts, sketches, and pictures to annotations on tracing paper. The team managing the conference room emphasized again how important this room was for the success of their project, which involved two teams: one in San Francisco and one in Grand Rapids. They all regretted that it could not be moved around and was by its nature very static.

6.2.1.4 Analysis

During the sessions we were surprised by the fluidity of the interaction and the enjoyment of the group during the brainstorming. Using very simple tools, such as Post-It™ notes and a pen, a leader can gather and structure a very large number of ideas, even as the group adhered closely to the five brainstorming rules. From a human interaction point of view, these tools are very successful because they are extremely reliable and do not get in the way of their user by imposing a structure on the interaction. To be successful a digital tool should provide the same reliability and simplicity of use, letting the user control the interaction.

The tool should also provide ample space for gathering large amounts of data. When asked about using a current digital brainstorming tool such as the LiveBoard system [EBG+92], leaders always pointed out that they are too small for most brainstorming. An ideal tool should accommodate at least 100 ideas, enabling it to be used in a standard one-hour brainstorming meeting.

The enjoyable social time spent during the brainstorm time contrasted sharply with the dreaded debriefing time. Even though it does not take place under the spotlights of the meeting, debriefing

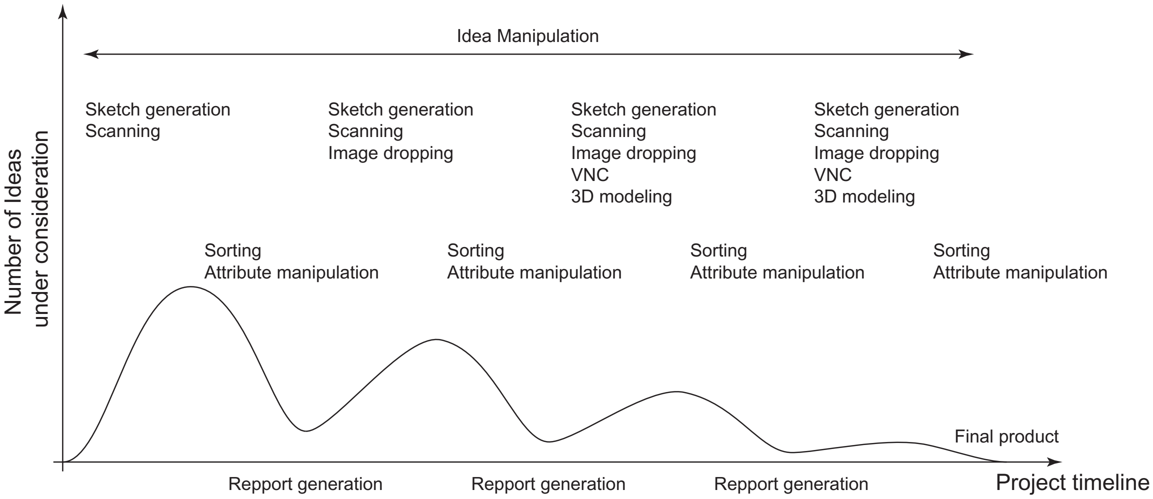


Figure 45: Brainstorming as part of the design process. Brainstorming can be used several times during the design process. During each brainstorm the same cycle of idea generation, idea selection and report creation is repeated but each cycle has its own content requirement. As the project progresses, fewer and fewer options are considered.

is a very important aspect of a brainstorming session during which not only the main ideas are collected but also key insights into how these ideas came to life. It is also during this debriefing phase that we identify the major flaw of the current brainstorming process: the disparity between the mainly paper-oriented brainstorming process and the computer-based post-meeting process of reporting and distribution. It is during this phase that the use of a digital brainstorming tool would provide the most rewards.

As the visit to the project room at IDEO San Francisco reminded us, having an ongoing brainstorming space is a very efficient way to keep track of the status of the project. The capability to create virtual project rooms that can then be called up on demand to the screen or easily transmitted to another location is also a great advantage of having a fully digital brainstorming tool. Furthermore, stroke and timing data collected from such a system can be integrated with other sources such as audio and video to help analyze the idea generation process of a brainstorming session.

The brainstorming session is key to IDEO's design process. Brainstorming is of course used in the very early stage of idea generation for new projects but also subsequently to refine projects as they progress. Even though all of these meetings are brainstorming per se, they have very different sets of requirements (as shown in Figure 45). At the beginning of a project most of the information that comes together in the meeting takes the shape of sketches or handwritten notes.

As the project makes progress, the same cycle can be repeated many time, but the sources of information will become more diversified. After the first meeting, participants will come back with pictures taken during ethnographic study or information found on the internet. Further along

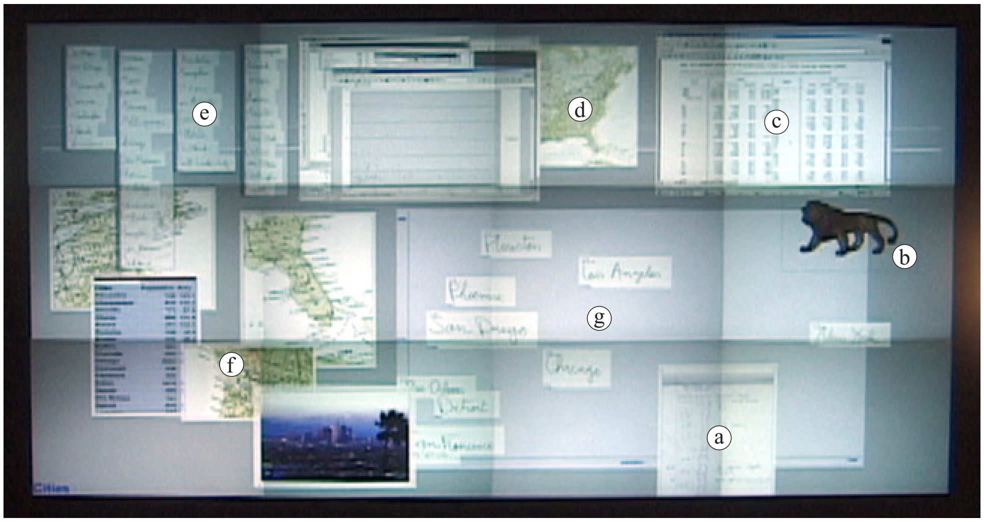


Figure 46: A typical PostBrainstorm screen. The user has accumulated content from various sources (counter-clockwise from the bottom right): a scan of a sketch (**a**), a 3D model of a lion (**b**), a VNC connection (**c**), an annotated map of the USA (**d**), several lists of handwritten material (**e**), several snapshots (**f**), and in the center a scatter plot container (**g**).

in the project's development, computer-generated data (such as spreadsheets) and 3D models become more and more important.

6.2.2 PostBrainstorm brainstorming tool

PostBrainstorm is a fully digital brainstorming tool that users can use to capture strokes, digital content, and scans of physical objects on a whiteboard-like surface. As the application starts, it appears as a large digital whiteboard empty except for two lines showing the limit of each ZoomScape area (Chapter 3.3.1). Using a digital pen, users can write or sketch directly on the board. Pressing the command button on the pen, they can bring up FlowMenu to bring in more content or manipulate existing information on the screen. Figure 46 shows a typical screen as the session develops: large graphics such as this map of the United States have been dropped onto the board, material has been scanned in using FlowScan, a remote-screen viewer gives access to any computer connected to the internet and running a Virtual Network Computing (VNC) [RSWH98] server, 3D models can be brought to the screen, and lists and containers are used extensively to structure content.

We will now present in turn the various ways of bringing in content and structuring information on the display.

6.2.2.1 Bringing content in

PostBrainstorm provides access to a wide variety of content:

- **Sketches and handwriting**, which can be created directly on the board,

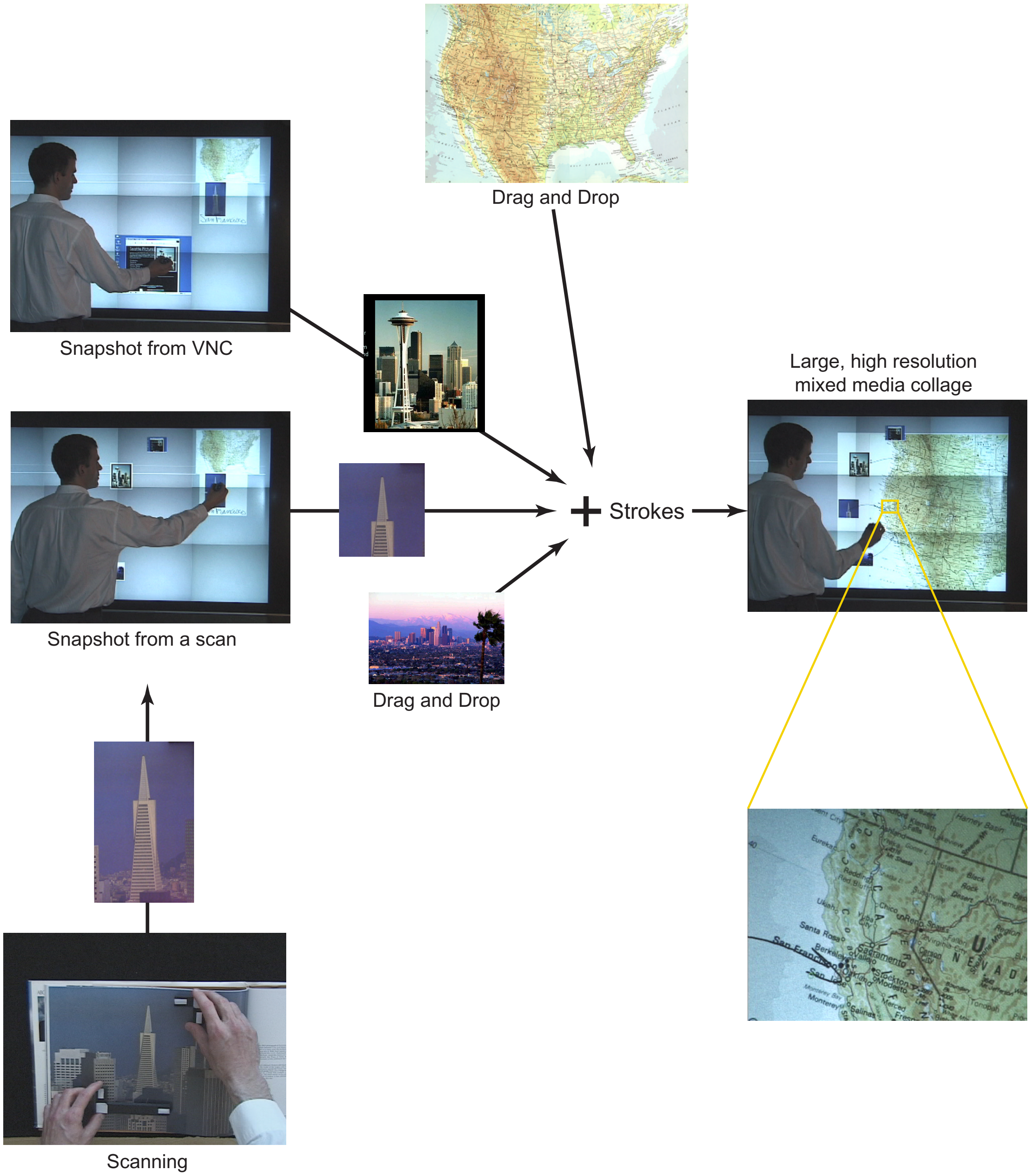


Figure 47: Assembling a digital mixed media collage using PostBrainstorm. In this example the user assembles an annotated map from the United States using sources from various origins. The map and the picture from Los Angeles were dropped to the screen from another computer, the picture of the Transamerica tower was first scanned from a book before taking a detail from the scan using the snapshot technique. The picture of the space needle was found on the web using the VNC viewer and was brought to the screen by making a snapshot of a page. Using the pen, all the pictures were assembled together as a large, high resolution annotated map.

- **Digital pictures**, which can be brought to the board from any computer in the iRoom (Chapter 5, page 53) using the iRoom controller,
- **Scan of physical media** such as paper-based documents (such as books, printouts, sketches) and three-dimensional artifacts that can be scanned using the FlowScan overhead scanner,
- **VNC** viewer [RSWH98], which can be used to interact with and take snapshots of the content of any computer running on a VNC server,
- **3D models,** which can be manipulated on the screen.

PostBrainstorm was designed so that people can easily assemble on the fly mixed-media collages on the screen using snapshot from different sources, sketching, and handwriting. A typical assembly is shown Figure 47.

6.2.2.1.1 Sketching and Writing

Using FlowMenu, users can create a virtual sheet of paper and draw on it with the pen. By default the system automatically expands the piece of paper to encompass all the strokes added to the sketch. In a way similar to FlatLand [MIEL99], our system also automatically merges drawings with bounding boxes that intersect the bounding box of the last drawn strokes. This feature makes it easy to create one drawing from disconnected parts. Experiences with several merging techniques have shown that this approach was the least error prone for users, because it is easy for the user to see which sketch will be affected by adding a new stroke. In the few instances where a merge occurred by mistake, it was easy for the user to recover using the undo mechanism.

It is also possible for the user to write on the board using *writing paper.* In that case, PostBrainstorm sends writing paper strokes to the Paragraph [Par] handwriting recognition engine. The handwriting recognition is done in the background and does not interfere with ongoing interactions. When recognized, the text is displayed on the lower-left corner of the piece of paper. We increased the size of the right and top borders of the writing paper so that there is room to cross any *t'*s and leave white space at the end to provide the expected segmentation for western handwriting. Contrary to the techniques used by FlatLand [MIEL99], we did not increase the size of the lower border since, most of the time, users prefer to write just one idea per piece of paper.

By default, the system interprets strokes made directly on the board without use of virtual paper to be handwriting, a natural choice since most of the content on the screen is likely to be words. Experiences showed that users seldom created a sheet of paper to sketch, but just start sketching on the board, ignoring the result of the recognition. To avoid spurious interpretations from sketches or a mix of text and graphics, writing recognition is automatically stopped if a sketch becomes taller than a specified threshold. In that case an ellipsis is added to the text already recognized.

Users can also copy any axis-aligned rectangular section of a sketch using FlowMenu. After selecting *Item... → snapshot* at one corner of the area of interest, the user can in the same interaction adjust the size of the selection area by dynamically adjusting the opposite corner. Strokes inside the rectangular selection area will be copied into a new sketch or writing paper depending on the original target class.

6.2.2.1.2 Digital images

As shown in Figure 31, page 54, the Mural is part of the iRoom infrastructure [FJHW00]. A user can use the iRoom controller to drop a JEPG image to the mural. When dropped from another connected computer, images appear in the lower-left corner of the screen and can be manipulated or annotated directly. Like sketch containers, image containers adjust their size so that they always contain all their strokes. Furthermore, the same merging algorithm used for sketches is used with digital images, making it easy to create a digital collage either by positioning these elements with the pen to overlap each other or making a stroke to join them.

Users can also copy any axis-aligned rectangular section of an image: after selecting *Item... → snapshot* at one corner of the area of interest, the user can in the same interaction adjust the size of the selection area by dynamically adjusting the opposite corner. The image as well as any strokes in this area will be copied.

6.2.2.1.3 Scanning

Early in the design process it became clear that PostBrainstorm should provide easy pathways between the physical world and the digital content manipulated on the Mural: given the current technology, people still prefer sketching on a piece of paper to sketching on a digital tablet (see Section 7.2.1). It is also true that many printed materials (books, document printouts, etc.) are used during meetings. Although a conventional flatbed scanner often provides a valid option for these kinds of document, these scanners can be bulky as well as attention consuming. Also, our studies showed that 3D artifacts are very often brought to meetings. At the early stages of a project, these might be materials or objects brought to trigger discussion. Later on, it could be a model or an early prototype. In all cases it is very important to integrate at least a snapshot of this artifact with other relevant information presented on the Mural. 3D artifacts cannot be effectively scanned with a flatbed scanner.

We decided to install in the room FlowScan (see Figure 39, page 69), an overhead scanner that can be controlled from the board. To scan an object the user places it anywhere on the active scanning area of the scanner. Then she places crop marks to delimit the area of interest. For establishing orientation, the long hand of the large mark is always placed in the bottom-left corner of the area of interest. Then the user can trigger the scan either from the Mural or from the scanning station. It takes around thirty seconds to process the image using our current driver, so a proxy is delivered to the Mural display right away. The user can manipulate and annotate the proxy before the picture appears on the screen. The picture created is equivalent to a full-color, 100-dpi scan. Empirical studies have shown that this value was necessary to provide a smooth rendering of freehand sketches on the screen.

6.2.2.1.4 Remote computer access

As more and more people are using laptops during meetings to search, gather, and process information, it seemed important for our system to provide a way to capture information from a laptop. We focused on a solution that makes it easy for people to transfer content from their laptop to the screen without worrying too much about the underlying format and provide an easy understanding of what is shared between participants of a meeting who may not fully trust each other. Our solution relies on VNC [RSWH98], a remote display system. Using a simple VNC client as part of the Mural software, Mural users can interact with any laptop on the network running a VNC server. After writing the name of the laptop on the screen, Mural users can use FlowMenu to create a new connection to the corresponding VNC server. Then, using the pen as a mouse, they can interact directly with that laptop via the Mural interface. Text can be entered by dropping a piece of writing paper with recognized handwriting inside the VNC viewer. If meeting participants do not trust each other, the VNC server can also be set in a “view only” mode that prevents direct interaction from the Mural. In that case only the visual content of the laptop’s screen becomes accessible to the group, making it easy for participants to understand and control how much information is disclosed to other participants.

At anytime the user can take a snapshot of any axis-aligned rectangular part of the VNC screen. After selecting the *Item...* → *snapshot* at one corner of the area of interest, the user can in the same interaction adjust the size of the selection area by dynamically adjusting the opposite corner. The snapshot created behaves like any other picture brought into the system.

6.2.2.1.5 3D model access

PostBrainstorm supports interactive manipulation of complex 3D models using the scalable rendering support of the WireGL distributed-graphics system. Users can call up a model by writing its file name on the screen and using the FlowMenu. The current implementation uses a small database to store rendering parameters such as the number of slaves involved, the viewpoint, and so on

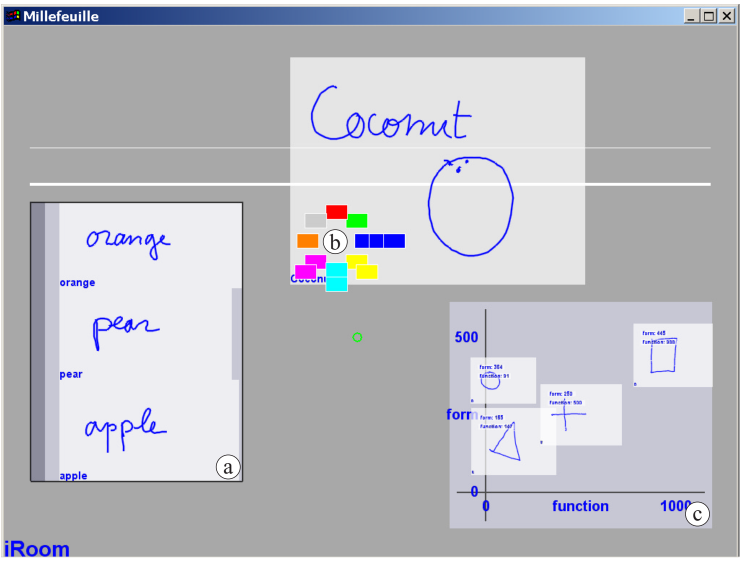


Figure 48: PostBrainstorm idea management tools. PostBrainstorm provides a set of containers such as the vertical list shown here (**a**). Users can drag and drop ideas to and from the list. A handle bar is provided on the left so that the list can be manipulated as a whole. Using the menu, users can also vote on ideas. Eight different votes can be cast. Each vote appears as a small virtual Post-It™ note at the bottom left of each object (**b**). Using Typed Drag and Drop (see Figure 10 page 23), one can assign attributes to objects. Using the ScatterPlot container (**c**), one can visualize the relationship between two attributes for a group of objects. The ScatterPlot parameters (minimum values, maximum values, and attribute names) can be modified by dropping new values on their respective labels.

for each model. More details about the distributed-rendering model provided by MilleFeuille and WireGL can be found in Section 5.2.4, page 65.

Currently, we use QSplat [RL00] as our underlying guaranteed frame-rate rendering engine. Since 3D models are often not brought to meetings until quite late in the design cycle (see Figure 45), we judged it acceptable to have users convert 3D models to the QSplat format and push the result to the cluster for maximum rendering performance during the meeting.

6.2.2.2 Managing ideas during the brainstorming session

Once created, items can be moved, erased, scaled using FlowMenu. One can also use FlowMenu to select items (either one by one or using a lasso metaphor) and move selected items at once. As the number of ideas on the board increases, these simple techniques become too limited. This section will describe other techniques used in PostBrainstorm to help users manage a large number of idea on the screen.

6.2.2.2.1 Containers

To manage ideas on the screen during the idea-generation phase, PostBrainstorm provides a variety of containers such as vertical and horizontal lists to group ideas together. Containers can be created from a selected group of objects or an idea can be dropped in or removed by a simple drag

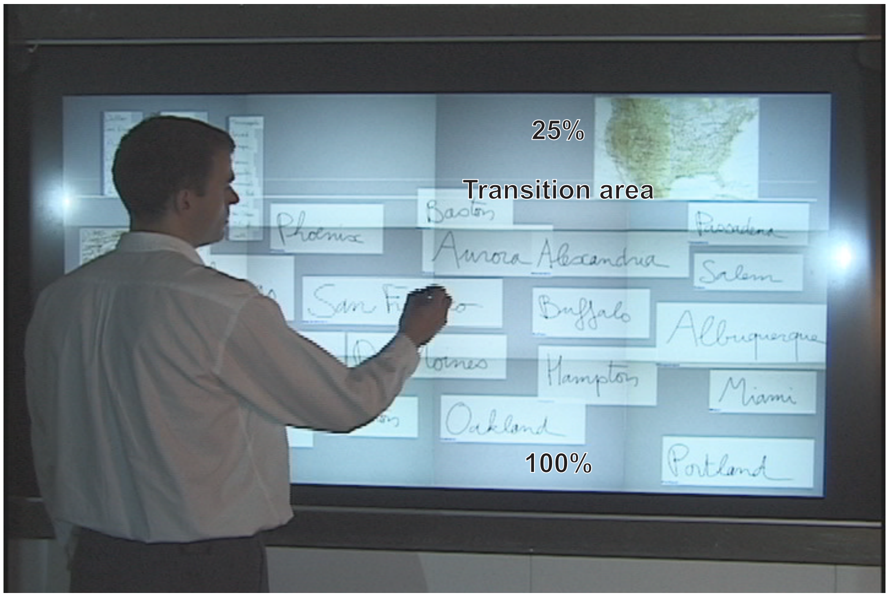


Figure 49: ZoomScape configuration for PostBrainstorm. For PostBrainstorm, the screen is divided into two main areas: The bottom four-fifths of the screen shows objects at their actual size, whereas the top fifth displays them at 25% of their actual size. A small transition zone in the middle allows for smooth transitions between the two areas.

and drop. A handle bar on the side of the list serves both as a visual cue of the grouping and as a way to manipulate the elements in the group as a whole (Figure 48a).

6.2.2.2.2 Space Management

During discussion with designers it became clear that the biggest problem with the current electronic whiteboard system is the lack of screen real estate. A closer look at designer practices helped us better understand why. IDEO brainstorms routinely generate 100 ideas in one hour, most of them written directly on large Post-It™ notes that are spread around the room as the meeting progresses. Most of the time, entries are written quite large so that participants in the meeting can see them, but also because the leader is writing on a vertical surface. Writing vertically involves the arm muscles, which leads to larger writing—typically 96 points (Figure 11, page 25).

As the meeting proceeds, one of the main roles of the leader is to keep important ideas front and center while maintaining other ideas in context so that they can be referenced during the meeting. Context ideas are in general pushed toward the sides of the room, further and further out of sight of the participants. This approach cannot work for systems with limited screen real estate, but the Focus+Context technique [FB95] of *scaling* is well adapted to our high-resolution display. In fact, on our display, the same 96-point handwriting is still readable by the user of the board when scaled down four times.

We used ZoomScape to provide a fluid transition between the large central content and the smaller contextual content. Our Mural settings use a two panel arrangement where the top fifth of the screen scales objects to 25%. The rest of the screen remains at 100% scale except a small transition area between the two areas (Figure 49). Using this system, new ideas are written in the main area of the screen and dragged to the top if they become contextual rather than central. Everything on the screen remains active so that the user can modify elements as needed. This 25% scaling area is particularly powerful for managing the screen real estate when working with containers such as lists. By moving a list inside the scaled-down region, the user frees up 75% of the list's real estate while still being able to add or remove items in the list. As a scaled-down list grows in length downward into the 100% area, the software, in effect, lets the user extend the 25% area of the screen to the expanded length of the list. Using this technique, the board can accommodate content representing up to 16 boards full of handwritten material.

6.2.2.3 Idea selections and post-meeting tools

After the idea-generation phase, the brainstorming session moves to an idea-selection phase where the best ideas are selected (see Figure 45). These ideas will be the centerpieces of the report created by the leader after the meeting. PostBrainstorm offers its users tools to select ideas and create reports by assembling material created during the meeting. It also provides persistence, so that sessions can be played back or transmitted to distant sites.

6.2.2.3.1 Idea selection

During the idea selection phase, each participant, using a small colorful Post-It™ note to indicate their specialty (e.g. mechanical engineering, human factors engineering), votes on the potential of each idea. Often the design group wants to factor in other attributes such as the estimated cost or the design complexity. PostBrainstorm provides support for both of these practices.

Our system provides a digital equivalent for voting by color-coded Post-It™ notes. Using FlowMenu, one can attach a vote to any idea by selecting one of the eight colors available. Votes appear at the lower-left corner of the idea as small colorful stamps laid out in an eight-segment wheel (Figure 48b).

Assigning attributes is somewhat more complicated since in general neither the name nor the value of attributes are known at the beginning of the session. This prevented us from using FlowMenu to directly assign attributes. We also wanted to avoid the use of a dialog box, which disrupts the interaction flow, introduces a temporal mode, and clutters the screen. Instead we rely on the Typed Drag and Drop (Section 3.2.2, page 22) to assign attributes to objects. To assign an attribute, the user first writes on the board a name-value pair such as “Cost 100”. Then, opening FlowMenu on top of the writing, she can designate the writing as an attribute definition and in the

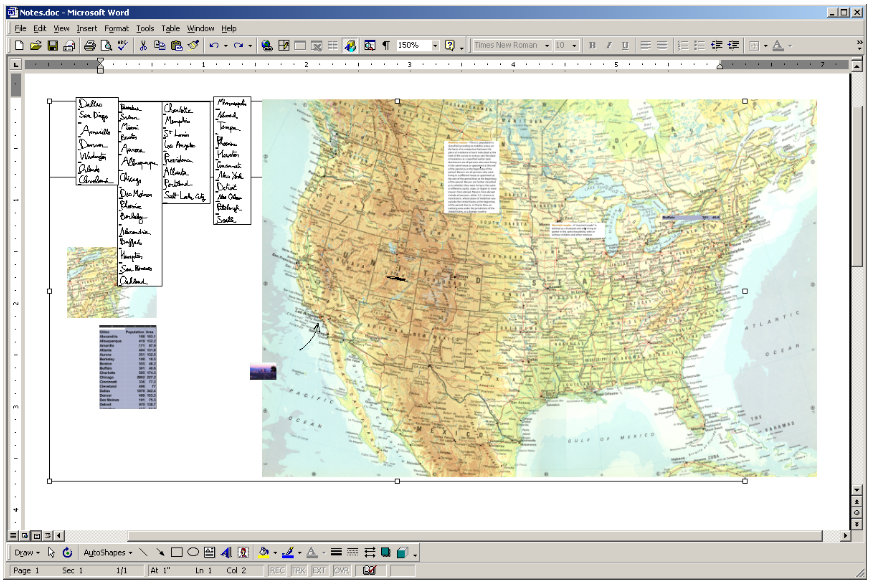


Figure 50: Exporting the board as a Microsoft Office™ document. At any time during the brainstorming sessions the user can capture the content on the board as a Word™ document. The document generated contains all strokes, images, and handwriting-recognition information present on the board and uses grouping as a structure mechanism. The user can edit this document or copy parts of it to generate her report.

same interaction drop it on top of the target idea. Using the recognized handwriting, PostBrainstorm sets the corresponding attribute for the target (Figure 10 page 23).

Once assigned, the relationship between two attributes of a group of ideas can be visualized using the ScatterPlot container. As objects are dropped inside a ScatterPlot, they assume the place corresponding to their attribute (Figure 48c). As with other containers, objects can be added or removed from the container with a simple drag and drop. To adjust the ScatterPlot parameters, such as attribute names or extremum of each axis, the user can simply drop new values onto the corresponding labels.

6.2.2.3.2 Report generation

Discussion with several designers made it clear that the system usability would be greatly increased if it provided an easy way to assemble the information on the board into a report. To address this need we provided a filter that creates a Microsoft Office™ document containing a snapshot of the screen (Figure 50). The snapshot is a drawing inside the document, which includes objects for each of the strokes, recognized text, images, and containers present on the board at the time of the snapshot. All these elements can be edited and/or copied into other documents to create customized reports for different purposes or audiences.

6.2.2.3.3 Persistence

One of the main attractions of a fully digital brainstorming tool is that it can provide easy storage retrieval and transfer of the information collected during brainstorming. To simplify the management of the information gathered, PostBrainstorm does not provide an explicit save function but instead logs all interactions performed by the user. The log is self-contained and includes all the content (other than 3D models and VNC screens) brought to the board. The log is the backbone of the undo system, which can go backward or forward.

The meeting log can be called to the board by name and will display the screen layout that was left at the end of the meeting. Since there is no save function per se, users entering the room to start a new meeting do not have to worry about saving the content present on the board since any state of the board can always be reconstructed from the log. Since the log is self-contained, it is easy to save the meeting on portable media such as CD-ROM or to transmit it to another site.

As in FlatLand [MIEL99], the implementation of the logging mechanism was made more complicated by two aspects of our system:

- **Non-reversible operations**. Many user operations are not reversible. For example, adding a stroke to a sketch changes the size of the sketch to accommodate the new stroke. Since sketches only expand, this operation is not reversible from the point of view of the sketch abstract data type. As each of the user's new strokes is logged, the system must also log all the information necessary to recover the state of the sketch before each additional stroke. We are using a simplified version of the transaction-based system described in FlatLand [EILM00] to indicate the semantic boundary of each user action in the log.
- **Asynchronous events**. Scanning a picture or adding a stroke to a handwriting sketch causes the state of the board to change asynchronously when the result of the scan or the handwriting-recognition engine is available for display. Since the asynchronous actions are not part of any transaction, they are logged directly as part of the pending transaction. That simple approach seems to make sense since our logging system is based on user action and the exact time of an action is not very important. If an asynchronous event arises while the user was doing nothing, the event was not there in the previous action, and will be there in the next.

6.2.3 Discussion

As the user study presented in Chapter 7 will show, the system was very well received by our target user group. In this section we will present some observations about some key features of our design and how they can be improved further.

6.2.3.1 Issuing commands

The combination of the FlowMenu and Typed Drag and Drop proved to be a very efficient way to get closer to our goal of a fluid interface stated in the introduction. FlowMenu provides a smooth transition from the structure of a menu to a set of gestures, helping the user reach the autonomous stage of learning. Using strokes as a scoping mechanism also proved very powerful to limit mode errors in the demanding environment of a brainstorming session. Typed Drag and Drop takes advantage of FlowMenu capability to mix command generation and interaction, letting the user create complicated commands on the fly, without relying on dialog boxes to enter parameters. This technique works very well in practice, but was limited in our experience by the performance of the handwriting-recognition software used by the system [Par]. Using a better dictionary and an engine designed to better recognize printed characters will help in that direction.

An unexpected difficulty in using the system with the pen was that people often did not press the command button before pressing the pen to the screen. That proved to be quite confusing since the pen would then leave a dot on the screen at the location where the command started. To correct this problem, we modified the stroke-generation tool to undo its work if the command button is pressed during the creation of a stroke. This proved to be extremely successful and prevented further generation of points.

6.2.3.2 Scanning

Our approach to scanning documents and objects proved very successful, even though delay and single-user access are still issues for most users. In particular, users often wanted to scan pictures in rapid succession, and the cycle time of our system (~ 15 seconds) was too slow to accommodate this kind of use. As explained in Section 5.3.1, page 71, the rapid progress of the CMOS imaging element should make it possible to address this limitation in the near future.

6.2.3.3 Space management

ZoomScape was probably the most successful part of our design. People used it extensively after only a very short exposure. Although the default two panel layout proved useful, it lacks versatility and limits to about 100 the number of ideas that can be managed on the board. Future versions of the system should include a higher zoom level at the edge of the screen and a way for users to define zoom landscapes suited to their specific needs.

6.2.3.4 Managing viewport

Most of the individual pieces of content manipulated during a typical PostBrainstorm session are small relative to the dimensions of the screen. That being the case, the capability to scale individual objects and groups using ZoomScape is effective for managing the screen's real estate. For this reason our interface does not offer a way to observe a small part of a larger document through a viewport. Nevertheless, viewports provide a convenient way to manage the screen footprint of a document regardless of its size. As the scope of fluid interface expands, finding a reliable way to manage viewports will become more and more pressing.

Many windowing systems rely on a scrollbar to manage a document's viewport, and the scrollbar was one reason for their success. Scrollbars, however, have drawbacks: they introduce visual clutter and place control of the viewport away from the locus of attention. We believe that scrollbars are unnecessary for controlling viewports. Our observations of how people manipulate large quantities of information suggest that using a combination of zooming and panning may be more efficient to control viewports [FB95]. A recent implementation of this principle proposed by Igarashi [IH00] could easily be extended to completely replace scrollbars for managing viewports.

Chapter 7
User study and testing

To evaluate our design we ran user tests. First we ran a controlled experiment to measure the performance of FlowMenu. Second, as we were starting to implement PostBrainstorm, we gathered as much feedback from users as we could and conducted informal usability testing to identify problems early in the implementation cycle. Finally, to evaluate the overall usability of our tool we ran a series of formal experiments in which professional designers performed several brainstorming sessions in our lab.

7.1 FlowMenu study

In this study, we wanted to investigate users' command selection speed with FlowMenu. Because of FlowMenu's design, we wanted not only to know the average speed but also speed variations depending on the complexity of command selection. Like Kurtenbach in his study of the marking menu, [KB93], we wanted to simulate expert performance by taking out the cognitive time associated with remembering the location of a specific menu item. While Kurtenbach [KB93] used cardinal directions to simulate expert performance, we found in early testing that this approach sometimes confused users. Instead we showed the user the exit and reentry directions needed for each menu choice using simple arrows.

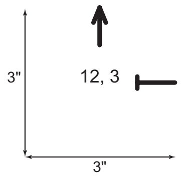


Figure 51: FlowMenu experiment prompt. To simulate expert behavior by the test users, we presented arrows showing the optimal exit and reentry directions. These arrows were labeled with corresponding clock directions as well (12 o'clock for an arrow going straight up, for example). The prompt fit inside a 3-inch square. The test program presents the requested selection prompt so that its center is 23" from the Mural edge and 62.75" from the ground. Users were able to perform menu selection anywhere on the screen.

7.1.1 Settings

All testing was performed on the Mural running a small application designed specifically for the purpose of the test. Initially the Mural was white. When the user touched the screen with the pen tip while pressing the command button, a FlowMenu appeared at the point of contact (labeled with eight clock directions: 12,2,3,4,6,8,9,and 10—one for each octant of the FlowMenu) and a new selection request was presented on the screen. The selection request took the form of a visual prompt indicating the motions that an expert user would make to quickly select the target item. The prompt consisted of two arrows and their corresponding clock directions shown at the center (see Figure 52). The first arrow, with a pointed arrowhead, showed the requested exit direction, whereas the arrow ending in a **T** indicated the requested reentry direction. The prompt was drawn inside a box 3 inches square that was displayed 62.75 inches from the floor (a comfortable height for users) and 23 inches from the left side of the Mural. If the user made the right selection, the prompt disappeared and the system waited for the user to touch the screen again for her next selection. If the user made a mistake, her selected input was shown below the prompt using the same arrowhead symbols so that she could compare her selection and the requested selection. When she pressed the pen to the screen again, the system repeated the prompt which caused an error and timed her new response. In all cases the user could trigger a selection by pressing the pen anywhere on the screen.

During each trial, the system measured both the time from the moment the pen touched the screen (and the prompt was presented) to the moment the selection was made and the time from the moment the pen left a 3-pixel radius around the touch position to the moment the selection was made. The first measurement includes the cognitive time to identify the target and perform the action, whereas the second evaluates the time it takes for the user to move her pen to the target.

The experiment was structured in sets. Each set covered all possible menu selections (64) twice. The 128 selections in each set were presented to the user in random order.

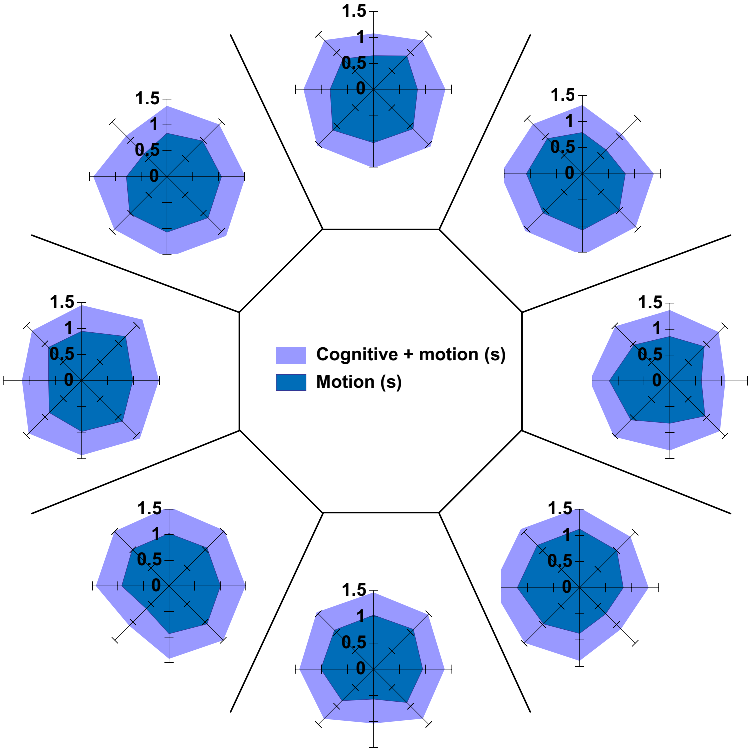


Figure 52: Selection speeds for FlowMenu. In each octant is a radar graph that shows data collected for selection starting with this octant. Here we show both total selection time from prompt to selection (light shading) and stroke time from center to selection (dark shading). Data shows no bias in any direction and, as expected, the simple in-and-out gesture is the fastest. Selection time increases as the distance between starting and ending octant increases. This data was collected from five subjects. Each point represents 40 samples. (Data sets used for these radar graphs can be found in Appendix 3.)

7.1.2 Users and protocol

For this study, our five users were fellow students from the computer science department who had very little or no experience in using the FlowMenu. After being given a brief tutorial about the goal of our research and explaining how the system worked, they were given the opportunity to practice an entire set of menu selections. Then their performance was recorded for four sets. Users were allowed to rest between sets.

7.1.3 Results

The full data set collected during the experiment has been tabulated in Appendix 3. The average time for selection was 1.4s from prompt to selection, while the average time for moving the pen from the center of the menu to selection was 0.94s. More detailed results are shown in Figure 52. In this figure, each octant contains a radar graph that shows the time it takes to reach a given octant from this octant. For each radar graph, the inner ring (dark tint) shows the motion times while the

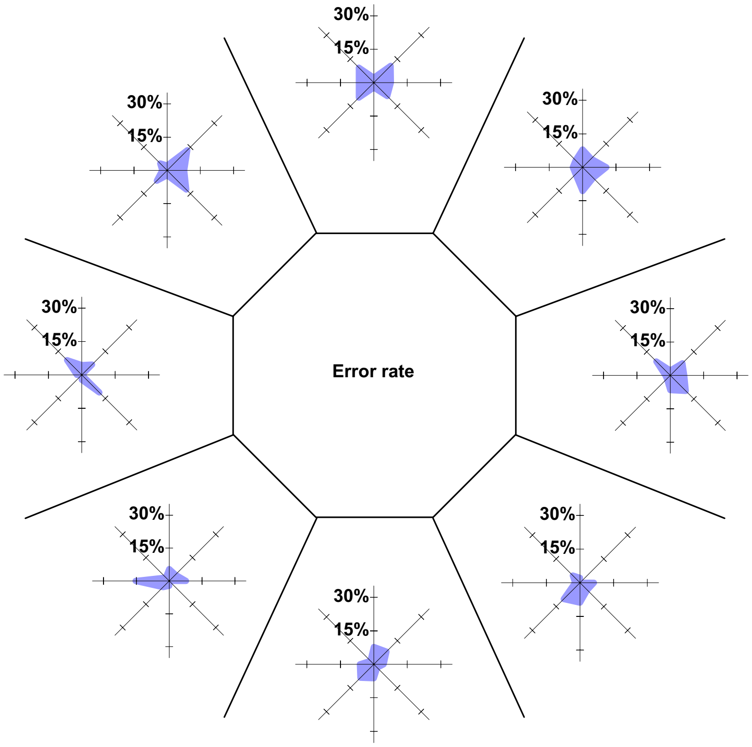


Figure 53: Selection error rates for FlowMenu. In each octant is a radar graph that shows error rates for selection starting with this octant. The error rate shows no bias in any direction. This data was collected from five subjects. Each point represents 40 samples. (Data sets used for these radar graphs can be found in Appendix 3.)

outer ring shows the times from prompt to selection. As expected, the flicking gesture corresponding to entering an octant and exiting through the same octant is the fastest. After that, the speed of selection increases slowly as the selection path increases. Figure 52 also shows that no starting direction is better than any other and that there is no asymmetry of speeds corresponding to going left or right during selection.

During this experiment, we also recorded the error rates (see Figure 53). On average the error rate was 6.8%. As in the case of timing, the error rate does not show significant bias toward any direction, but the octant to the right of the starting octant seems slightly more error prone.

These results are encouraging since they are very similar to the published values for Marking menus of similar complexity. Kurtenbach [KB93] reports an average speed of 1.4s and an error rate of 6.6%. But care is needed in comparing these results. On one hand, Kurtenbach's [KB93] results are based on a test performed with a tablet computer—a setup in which interaction does not require movement of the full arm. This setup should give the Kurtenbach's setting an advantage over our setting, with its vertical input interaction. Kurtenbach [KB93] also told the user how they scored, another factor which enhances user performance. On the other hand, our setup does not use

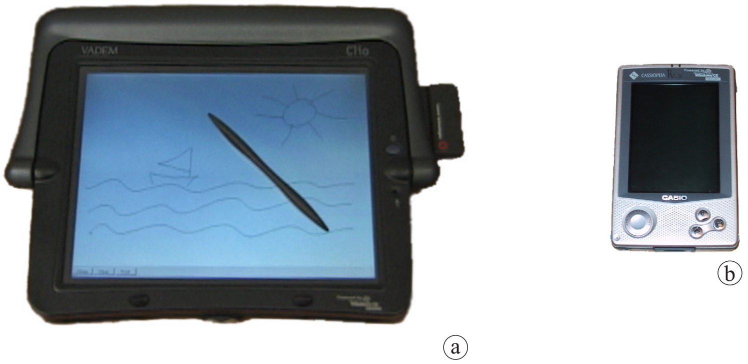


Figure 54: PDA used in the experiment. 3 Clio 1000 (**a**) and 3 Cassiopeia (**b**) were used as digital sketch pad. The Clio is shown running the simple sketching application.

cardinal directions but arrows, which are cognitively less demanding. Even though we believe that our directional arrows simulate "expert use" more accurately, it also puts us at an advantage. It is still not clear how the advantages and disadvantages of each aspect in the two experimental setups balance out, but it seems safe to say that the differences between the two systems may have little impact in practice.

7.2 PostBrainstorm

The development of PostBrainstorm stretched over a year and a half. During this period, we had numerous contacts with our target users to guide our design effort. At the end of the project, we conducted a formal user study to evaluate the final version of the tool.

7.2.1 Preliminary testing

As we were assembling our system, we gathered as much early-user feedback as possible to identify major design issues and where we should focus our research. Most of the time this took the shape of impromptu demonstrations when a group of designers from IDEO would stop by our lab. But we also carried out an experimental session at an early stage of the final application design. The main goal of this session was to understand the limitations of conducting a brainstorming session mediated by the computer system (rather than a human leader) and using a graphics tablet as the sole input device.

Several authors (see for example [PLD01]) report that computer-mediated brainstorming has great advantages over human-mediated brainstorming, including:

- All postings are treated as equally valuable,
- Each posting on the board remains anonymous, but the system can log the originator of a

given idea for future reference (patents, etc.),

- The computer system performs better arbitration and in fact handles several ideas at the same time, something a human facilitator cannot do.

So it seemed important to evaluate this approach in the context of IDEO practice. Furthermore given our goal of a fully digital brainstorming tool, and the importance of sketching during brainstorming, we wanted to investigate how designers would respond to using a simple tablet computer as a sketching tool.

7.2.1.1 Setting

For this preliminary test, we used the Mural as a common posting surface and six personal device assistants (PDAs) for sketch pads. Three PDAs were Clio C1000 equipped (Figure 54) with wireless network connections, and three PDAs were Cassiopeia equipped with an ethernet card and a tether. All were running a simple sketching application that managed the PDA screen as well as its proxy on the mural. Besides a large drawing surface, the sketching application provided three buttons: *quit* to close the application, *clear* to clear the drawing on the PDA screen (but not on the Mural), and *post* to start a new drawing. After the post button was pressed the PDA screen was cleared and a new proxy area was allocated on the Mural as soon as the first stroke was drawn on the PDA screen. During the drawing, each stroke was transmitted to the Mural as it was made and rendered in the proxy area.

The Mural screen was divided into two areas. The bottom area displayed the most recent sketches, shown 66% of their original size in pixels, laid out in columns starting from the left and progressing to the right. When this area filled up, the sketches in the left-most column were transferred to the top area of the screen. and all others columns were then shifted one step to the left. The top area contained the oldest drawings shown 16.6% of their original size. Like the main area it laid out the sketches in columns starting from the left side. More than 150 sketches could fit on the display at a given time, with about 50 of these shown at the 66% size. Given the screen resolution, someone standing at arm’s length from the screen could read all the cells on the screen. Because the lag time and noise in this early system prevented using an eBeam [EFI] marker directly on the surface of the screen, the mural was used as a passive display during that experiment.

Finally the Mural software logged all strokes as they were sent from the PDAs. A simple tool read this log and produced a Powerpoint show, assigning each sketch to its own slide.

7.2.1.2 Participants and organization

All the participants for this test were designers from IDEO. Half of them were primarily designers, and half of them were trained brainstormers. One of them had studied different digital brainstorming systems.

After a short presentation of our system, each participant was tutored in how to use the tablet. Then the group started a brainstorming session without a leader. We chose the topic “new design for light switches” so that sketches could be used extensively. During that phase of the meeting, we took notes without commenting.

After the brainstorming session, the group started an open discussion about the problems they encountered.

7.2.1.3 Result

Despite its apparent simplicity this experiment yielded insights about the brainstorming process and how the use of technology can affect it. We will explore each point in turn.

7.2.1.3.1 Using the tablet computer

The overall reaction of the group toward a graphics tablet was negative. Current technologies for graphics tablets do not offer a good substitute for paper. Tablets are considered too bulky, and people who are used to sketching on paper feel that the surface is too slippery and without the control they enjoy with paper and pen. In the current state of the art it seems difficult to remedy this lack of sensitivity.

The simple tablet interface, which sent each stroke as it was created caused major problems. First it prevented each user from finalizing their drawing before sending it to the board, leaving them with little control on how their idea would be perceived by the group. More importantly it drew too much of the group’s attention toward the sketching process, limiting their generation of new ideas. Furthermore, the automatic stacking of ideas into columns often made it difficult for each participant to understand where his or her idea was displayed on the board.

During the discussion several options were proposed to create a more symmetrical interface with which each participant could access all the information on the board on her own tablet or have a remote-control capability over the board. These ideas were very attractive but unfortunately difficult to implement in the current state of the art, since current graphics tablets do not have the required resources to support these kinds of interfaces.

7.2.1.3.2 Group dynamics

Probably the most valuable input came from the group’s discussion of the dynamics of a leaderless brainstorming session as compared to a standard brainstorming session at IDEO. Almost all our participants mentioned that in this setting it was very difficult for them stay focused on the topic. Part of this problem came from the characteristics of the drawing program, as explained above, but during the discussion it became clear that the main culprit was the lack of a leader.

During a traditional brainstorming the leader plays four major roles:

1. Enforce the standard brainstorming rules as described in Section 6.2.1 page 78,
2. Provide a steering force, making sure that everyone in the session shares the same focus,
3. Provide a consistent depiction of ideas. Since all the ideas are committed to the board by the same person, they all get on a single footing with respect to drawing capability and style and metaphors used.
4. Force participants to share their ideas with the group. Because participants have to explain their idea to the leader, they must "broadcast" their idea to all members of the group. This aspect is very important because it keeps all the members of the group informed about the current "state of mind" of the session.

Computers can easily substitute for the first item by providing the correct protocol between participants, but it is very difficult for a computer system to emulate the other leadership roles. It was a common remark that brainstorming is supposed to be fun and be performed in a friendly and enjoyable way. Also, none of the people present believed that brainstorming meetings led by the anonymous computer were as successful as the current practice at IDEO.

The results of this study prompted us to give up the use of tablet computer and explore more suitable means to inputs sketch into the system. Our solution for the final system was to use FlowScan, a non-intrusive overhead scanner described in Section 5.3, page 69.

7.2.2 Final study

To validate our design, we asked several groups of designers to use the final version of our system during brainstorming sessions on topics they encounter in their everyday practice. The goal of the experiment was to provide qualitative data about the user experience during a typical session.

7.2.2.1 Settings

All the sessions took place in the iRoom, our augmented meeting place that we used as a laboratory. The room is shown in Figure 55. Besides the Mural, it contains three SMART Boards™ and the Stanford interactive table. Thanks to the room's infrastructure it is easy for users to access all the devices and move information between computers or screens. The room also provides several network access points so users can connect their own laptop to the network.

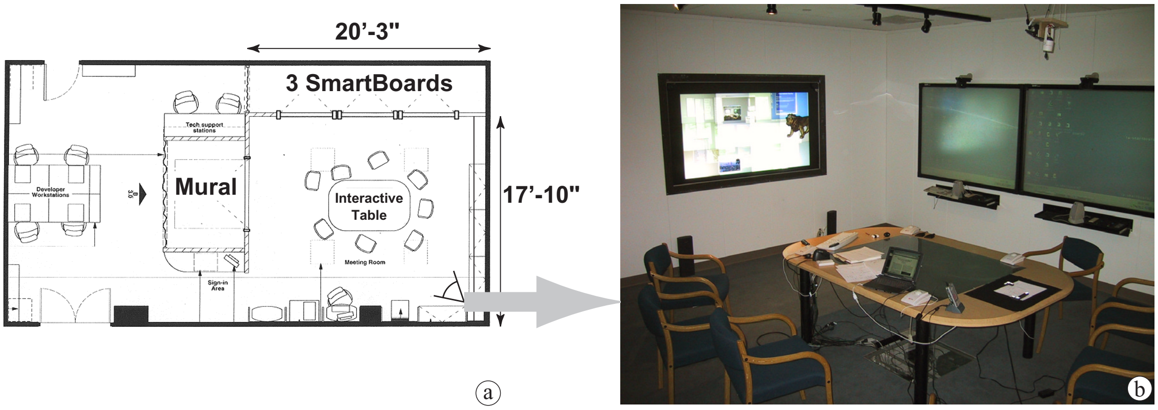


Figure 55: Layout of the iRoom used for the PostBrainstorm experiment. The iRoom floor plan (**a**) and view taken from the corner across from the entrance looking at the mural (**b**) as indicated by the eye symbol in the lower-right corner of the floor plan. Besides the Mural, the iRoom contains three SMART Boards™ and one Interactive table, none of which were used for the experiment.

The floor plan of the iRoom is depicted in Figure 55a. During the test, neither the SMART Boards™ nor the interactive table were used. At their request, we provided some users with laptops to let them access the internet. One member of the group was leading the meeting and operating the Mural, while other members sat around the table facing the Mural. Often the user at the far end of the table was scanning documents as needed, by setting the crop marks and signaling the leader when they were set. The Mural was running the PostBrainstorm software described in Section 6.2 with the following restrictions: the group was unable to annotate images or create mixed-media collage that combined images and sketches (such as the one shown in Figure 47, page 85).

7.2.2.2 Participants

To have insight into best practices, we chose trained designers from IDEO and Speck Product Design, two Palo Alto design firms. Attendance at the sessions ranged from 4 to 7.

Prior to each meeting, a person was trained as a leader for one hour. The training covered basic features of the software and let the user practice on his or her own. The training focused on basic features and other features the leader thought they would need during the session. Because, most of the brainstorming sessions were exploratory in nature, and the use of digital information is not yet common practice, features such as dropping images, VNC use, and attributes management were not used very often during this test. Session topics were picked by each leader.

7.2.2.3 Observations

Our observation covered five sessions and a total of 26 participants. For each session except the first one, they used topics from a project their studio was currently engaged with. One long session (three hours) had to be split in two because of performance problems with the software.

We acted as observers during each session and, when needed, we helped the leader find a feature or perform a given task. As far as possible, we provided help without performing the action but rather by explaining to the users what they should do.

At the end of each session all participants were ask to fill out a questionnaire covering specific aspects of the interface as well as general questions regarding how the system changed the group dynamic. Leader and participants filled out different forms, both of which are reproduced in Annex 2. All quantitative results were gathered using a scale ranking from worst (1) to best (7).

7.2.2.4 Results

The data gathered from this questionnaire is shown in Figure 56 and the full data set collected during the experiment has been tabulated in Appendix 1. Overall, results from the study were very encouraging. Participants found our system enjoyable to use (5.9 out of 7) and believed it would be useful in their work on an everyday basis (5.8).

ZoomScape, our technique to address the space crunch of current digital blackboards received (6.5), the highest score of all the questions; yet many users believed that the space was still insufficient: the question “Was the board providing enough space?” received a comparatively low score (5.0).

Our strategies of importing pieces of information via FlowScan (6.0) and exporting information to a Word document (6.1) were very well received. Further discussion with the leaders confirmed that other capabilities of the system, such as dropping images from a laptop or taking snapshots from the VNC viewer, would have great potential use in their everyday work practices.

At the other end of the scale, we found major problems with the pen—which did not feel right as a sketching tool (4.2) and did not have a comfortable button (3.3)—and with the handwriting recognition (3.4), which was easily be fooled by specific handwriting styles such as the printed characters often used by brainstormer.

We observed these two problems at the beginning of the experiment. We were unable to redesign the pen casing to accommodate a new command button, and even though Calligrapher [Par] delivered very high performance, it was not accurate enough for our use. We suspect that using capitals instead of cursive upset the system because the system was originally designed for cursive recognition. Furthermore, using printing exposed an imperfection of the pen tracking system, which caused a small hook to appear at the end of each stroke. These hooks confused the handwriting engine even further.

It is still unclear how the system will be perceived in the long run. Because of constraints on the designers’ schedules, training was limited and it was clear during the sessions that the leaders were

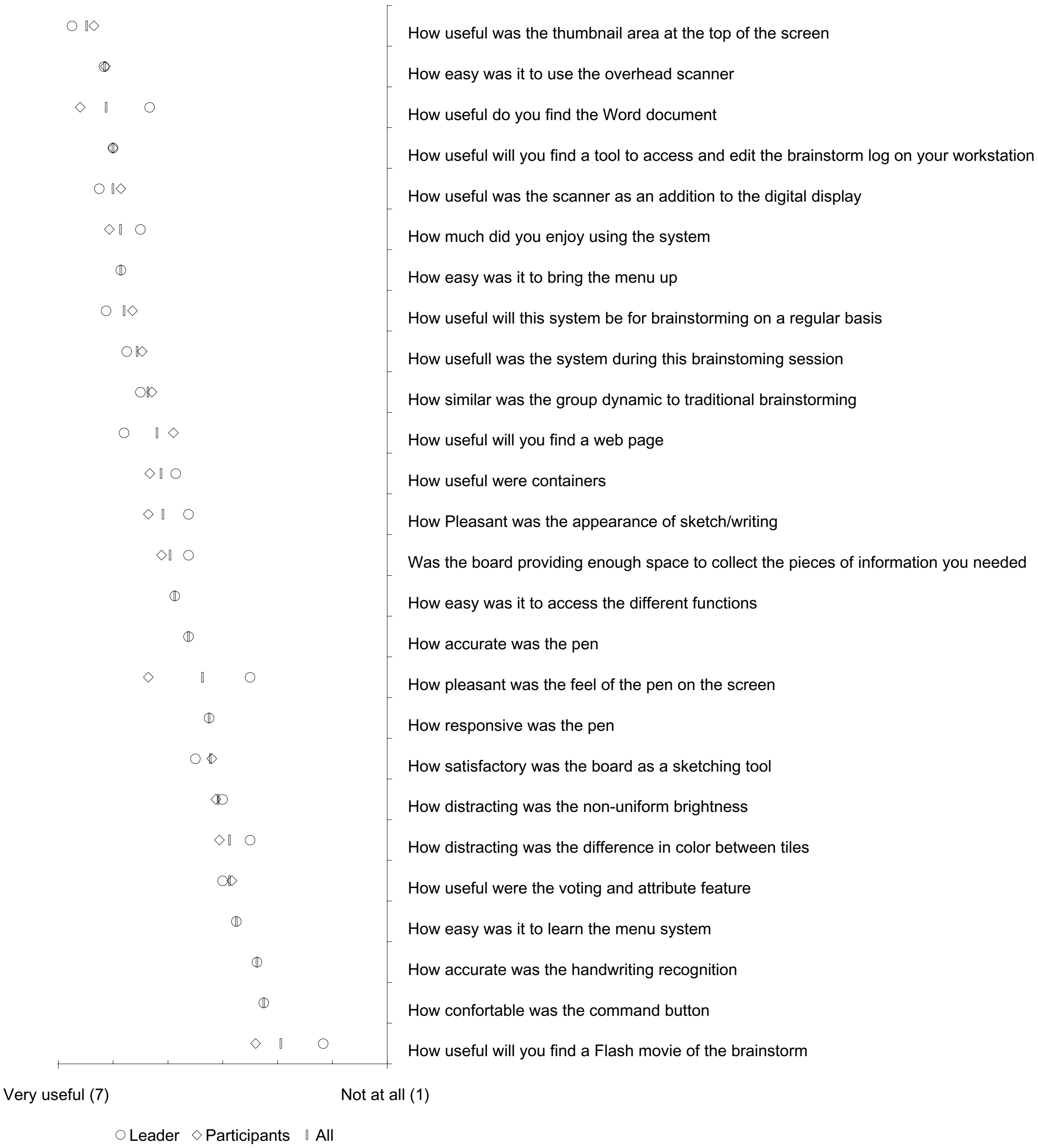


Figure 56: PostBrainstorm study results. Questions are shown at the right. On the left, the graph shows each question's average score for the leaders, the participants, and the average of all meeting attendees. The study involved 26 users of which only 24 returned the questionnaire (8 leaders and 16 non-leaders). The data set used to create this chart can be found in Annex 3.

Figure 57: The Chrysler Design Award 2001. PostBrainstorm was used in the early stage of the design of this award.

still learning the system. The situation was made even more difficult for them by the fact that they had to practice a freshly acquired skill in front of their peers, exposing themselves as less than expert in this regard.

The evaluation of the design is made even more complicated by the fact that even though the system was designed to mimic the use of a whiteboard, different patterns of use are needed to take advantage of its features. Looking at a typical pattern of use, leaders were using the system as a whiteboard to be filled with Post-It™ notes. For example, they were leveraging the fact that information can be moved around easily, but they did not use list containers. Instead they created one big sketch with all their list items, which made it difficult to move items on the list around. It is still unclear if this pattern of use was caused by a shortcoming of the interface or simply the lack of practice with the system.

7.2.2.5 Discussion

The result of the user studies we performed using PostBrainstorm were extremely encouraging and showed that being able to bring non-digital content, manage large quantities of information using ZoomScape, and export the content of the board to an easily manipulated form (an Office™ application) were keys to our success. Our approach of focusing on strokes and images as primary media delivered the flexibility we needed for successful brainstorming. But the system still can be improved. In particular we should expand opportunistic character recognition of strokes to include character recognition on scanned images and snapshots created from our VNC viewer. This would increase the range of content to be used as data for further manipulation after the brainstorming session.

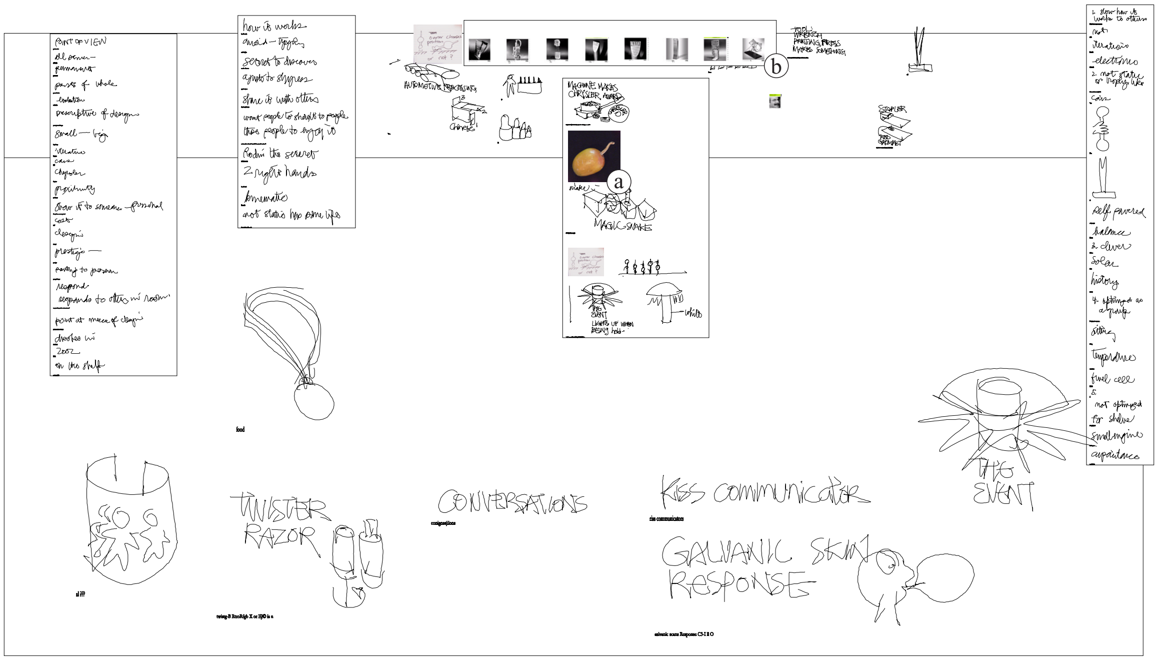


Figure 58: Snapshot of the board at the end of the first Chrysler Design Award meeting. Note the heavy use of lists in conjunction with ZoomScape. During this meeting FlowScan was used to scan a "Snake in a Mango" toy (**a**). At the end of the meeting snapshots of the previous awards were collected from the Chrysler Design Award web site (**b**) [Chr].

Our user study also has some limitations of its own. Time constraints limited training, which limited the number of features the leaders could comfortably use during any given session. Because they had such limited exposure to the tool, most of the users were unable to adapt their work patterns to the tool. At the end of this study, it was clear that a longitudinal user study would be needed to better understand the pattern of use of this tool.

A small step in that direction was taken when PostBrainstorm was used during the early design stage of the Chrysler Design Award 2001 (Figure 57). The Chrysler Design Award "celebrates a commitment to innovation, excellence and sustained vision" [Chr]. David Kelley, the co-founder of IDEO, was one of the six designers to receive the award in 2000 and was asked to design the 2001 edition. PostBrainstorm was used during a series of three brainstorming sessions. Figure 58 and Figure 59 reproduce snapshots of the screen for two of the sessions. They illustrate typical use of the system. In Figure 58a one can see that the scanner was used to capture a "Snake in a Mango" toy, while in Figure 59a one can see how the scanner was used to prime the board with sketches at the beginning of the meeting. One of these sketches was used later as a basis for discussion and as a shape to be drawn on top of and developed (Figure 59b). In both figures, one can see the heavy use of lists in conjunction with ZoomScape to structure ideas and manage board space.

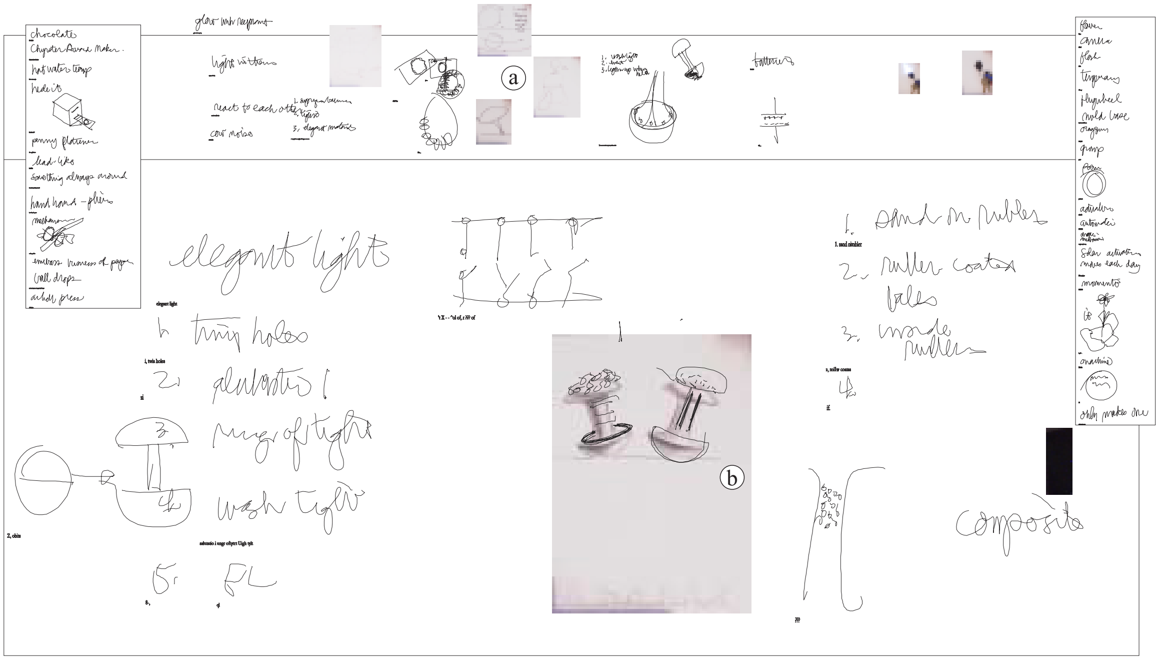


Figure 59: Snapshot of the board at the end of the second Chrysler Design Award meeting. During this meeting FlowScan was used at the beginning of the meeting to prime the board with design ideas (**a**). Note that one of the sketches was used later on as a basis to for discussion and visual development (**b**).

These results were very encouraging and we hope to be able to conduct more studies to observe how long-term (on the scale of months) use modifies patterns of use as people master the full gamut of the system and discover new uses unanticipated by its designers.

Chapter 8
Conclusion

Over the last 25 years the desktop interface pioneered by the Xerox Star [JRV+89] system has proven to be very powerful and versatile. It is credited with many improvements in user access and simplicity of use. Yet in many areas where interaction style is casual and fluid interaction is important, people still rely on the tools of pen and paper. Maybe no area expresses this contrast better than the brainstorming process in product design. Although most of the industrial design process relies heavily on computers, leading industrial design firms are still using Sharpie pens and large Post-It™ notes to conduct brainstorming sessions. Similar examples can be found in architecture, engineering, and mathematics.

Using the informal use of a whiteboard as an example we showed that it was possible to bridge the gap between the casual use of this tool and access to the powerful features of today's computers. Our approach in designing such a *fluid* interface, which limits workflow disruption, relied on five principles:

1. **Reducing cognitive load**
2. **Avoiding temporal modes**
3. **Avoiding dialogs**
4. **Developing graceful latency management**
5. **Avoiding visual artifacts**

These principles led to the design of several interaction techniques: FlowMenu, Typed Drag and Drop, ZoomScape, and MultiPoint. They were put into practice by designing the Stanford

Interactive Mural, a large high-resolution interactive wall display and developing two applications for it. The Geometer's Workbench let users study differential geometry with a new front end for Mathematica using an interaction style is reminiscent of idioms used by mathematicians on a standard whiteboard. PostBrainstorm, a fully digital brainstorming tool, let designers gather and structure pieces of information from a wide variety of sources such as sketches, photographs, artifacts, web content, and 3D models. PostBrainstorm was used by professional designers during real life brainstorming and was very well received.

Even though these techniques were illustrated in two domain-specific application contexts, they are applicable to a broader scope of activity. For example, FlowMenu and Typed Drag and Drop can be used on many different types of interactive surfaces such as PDA and tablet computers. The ZoomScape technique can be generalized to provide not only position-dependent scaling, but a more general position-dependent affine transform. A combination of scaling and rotation is particularly interesting since it will naturally extend the ZoomScape principle to an horizontal surface such as a table.

Another venue for large, high-resolution display surfaces is scientific visualizations. Observing users using the mural to visualize the result of large simulations, it was apparent that new interaction paradigms are needed so that users can fully exploit the advantages of the display. Current tools are ill-equipped to deal with large displays. Often they don't allow the user to interact directly with the model on the screen, providing only a remote control interaction paradigm. But their most important limitation may come from the fact that they do not deal with latency gracefully. Often they force the user into a "change view, wait, observe" cycle. This cycle makes exploratory behavior almost impossible. This limitation will be even more acute in cases where the user not only visualizes simulation results but also steers further computations. In both cases we hope that approaches similar to MultiPoint will lead to new interaction styles in scientific computing that encourage exploration and take into account the inherent latency present in computation-intensive modeling.

It is yet to be seen how the techniques we have presented will be accepted in the long run. The results of our user study were very encouraging, but its scope was limited because of the hardware requirements for our tool. One of the main limitations was the short training period designers were able to afford. Looking at participants using our board it was clear that they were drawing heavily from their whiteboard skills and did not take advantage of all the features of the electronic tool. We hope that in the near future we will be able to conduct longitudinal studies to understand better how this new feature can modify the dynamics of brainstorming activities and the design process at large.

Looking further into the future, the scope of this work goes beyond providing interfaces for large high-resolution interactive surfaces. By providing a more casual interface to computer resources, fluid interaction interfaces enable designers to use computer tools in the early stages of the design process. During this phase designers have to deal with ill-defined, poorly understood problems. To address this situation they use tools that deliver a fluid, flexible interaction style, such as sketching, relying on the computer only when a solution becomes concrete. Yet gaining access to computational resources early in the design cycle can help designers explore a larger set of options, reuse their work further along in the design process, and document their process more effectively. Mixing the flexibility of early design techniques with the expressive power of today's desktop interfaces will provide a new class of *interactive cognition* [Ged98] tools well suited to the creative process.

In turn, designing a successful interface for interactive cognition will help us better understand the mechanisms underlying casual use of complex tools. These mechanisms will be key to creating human computer interfaces well adapted to information appliances of all sorts. We expect that the fluid interaction interface principles and techniques described in the pages above will be a first step to delivering successful interfaces for these devices.

Appendix 1
FlowMenu user study data

In this appendix, we report the result of the FlowMenu user study described in Section 7.1

		Ending Octan							
		N	**NE**	**E**	**SE**	**S**	**SW**	**W**	**NW**
Starting Octan	**N**	1.07 (0.12)	1.33 (0.24)	1.39 (0.20)	1.55 (0.19)	1.52 (0.20)	1.54 (0.24)	1.35 (0.17)	1.34 (0.24)
	NE	1.32 (0.18)	1.08 (0.13)	1.36 (0.23)	1.45 (0.17)	1.56 (0.23)	1.55 (0.18)	1.57 (0.20)	1.34 (0.15)
	E	1.37 (0.16)	1.33 (0.25)	1.07 (0.20)	1.36 (0.25)	1.34 (0.15)	1.52 (0.19)	1.54 (0.18)	1.49 (0.17)
	SE	1.53 (0.21)	1.37 (0.14)	1.32 (0.18)	1.12 (0.15)	1.39 (0.24)	1.49 (0.24)	1.67 (0.23)	1.58 (0.24)
	S	1.47 (0.22)	1.48 (0.21)	1.36 (0.18)	1.34 (0.21)	1.04 (0.15)	1.35 (0.19)	1.42 (0.21)	1.54 (0.18)
	SW	1.54 (0.20)	1.53 (0.21)	1.49 (0.17)	1.50 (0.24)	1.42 (0.27)	1.10 (0.15)	1.43 (0.24)	1.51 (0.24)
	W	1.45 (0.21)	1.66 (0.32)	1.49 (0.18)	1.59 (0.23)	1.44 (0.23)	1.44 (0.24)	1.15 (0.19)	1.35 (0.16)
	NW	1.37 (0.22)	1.42 (0.23)	1.52 (0.22)	1.62 (0.27)	1.57 (0.23)	1.44 (0.18)	1.42 (0.27)	1.12 (0.14)

Table 1: Total selection time for each combination of starting and ending octants. Each cell shows the time between the moment the pen touches the screen and the stimulus is shown and the moment the selection is performed. 5 users took part in this study, collecting a total of 40 data points per menu selection. All times are in second, standard deviation are shown in parenthesis.

		Ending Octan							
		N	**NE**	**E**	**SE**	**S**	**SW**	**W**	**NW**
Starting Octan	**N**	0.65 (0.08)	0.91 (0.24)	0.84 (0.13)	1.08 (0.13)	1.03 (0.15)	1.11 (0.16)	0.84 (0.11)	0.83 (0.14)
	NE	0.79 (0.10)	0.64 (0.09)	0.82 (0.17)	0.99 (0.10)	1.08 (0.20)	1.06 (0.11)	1.07 (0.16)	0.95 (0.10)
	E	0.86 (0.08)	0.94 (0.17)	0.61 (0.17)	0.96 (0.17)	0.82 (0.06)	1.07 (0.14)	1.17 (0.05)	0.98 (0.09)
	SE	1.12 (0.14)	1.00 (0.12)	0.84 (0.06)	0.71 (0.16)	0.87 (0.18)	0.98 (0.18)	1.19 (0.12)	1.15 (0.23)
	S	1.03 (0.13)	1.08 (0.11)	0.94 (0.08)	0.90 (0.11)	0.57 (0.13)	0.86 (0.07)	1.01 (0.11)	1.06 (0.04)
	SW	1.02 (0.15)	1.05 (0.11)	1.00 (0.14)	1.05 (0.13)	0.94 (0.20)	0.63 (0.18)	0.93 (0.17)	1.02 (0.14)
	W	0.95 (0.06)	1.20 (0.21)	0.99 (0.14)	1.11 (0.09)	0.99 (0.12)	0.88 (0.07)	0.64 (0.17)	0.90 (0.12)
	NW	0.84 (0.12)	0.97 (0.17)	1.06 (0.16)	1.12 (0.19)	1.07 (0.15)	1.02 (0.19)	0.79 (0.07)	0.62 (0.08)

Table 2: Gesture times for each combination of starting and ending octants. Each cell shows the time between the moment the pen is more than 3 pixels away from the center of the menu and the moment the selection is performed. 5 users took part in this study, collecting a total of 40 data points per menu selection. All times are in second, standard deviation are shown in parenthesis.

		Ending Octan							
		N	**NE**	**E**	**SE**	**S**	**SW**	**W**	**NW**
Starting Octan	**N**	2.2 (5.0)	10.7 (7.1)	7.7(12.0)	8.4 (8.5)	2.2 (5.0)	9.5(13.2)	6.7 (6.1)	9.5(13.2)
	NE	8.0(11.0)	5.5(12.2)	10.7(15.3)	6.7(14.9)	10.7 (7.1)	4.4 (6.1)	4.4 (6.1)	4.4 (6.1)
	E	2.2 (5.0)	7.7(12.0)	6.2 (9.1)	9.9(16.7)	6.7 (6.1)	2.2 (5.0)	2.2 (5.0)	8.9(14.5)
	SE	2.2 (5.0)	0.0 (0.0)	6.2 (9.1)	4.4 (6.1)	8.9(14.5)	10.2(10.0)	2.2 (5.0)	4.4 (6.1)
	S	7.7(17.2)	8.0(11.0)	4.4 (6.1)	2.2 (5.0)	6.2 (9.1)	8.4 (8.5)	6.2 (9.1)	2.2 (5.0)
	SW	5.5(12.2)	4.0 (8.9)	7.7(12.0)	0.0 (0.0)	2.2 (5.0)	4.0 (8.9)	15.4(11.8)	2.2 (5.0)
	W	4.4 (6.1)	6.7(14.9)	0.0 (0.0)	11.1(13.6)	2.2 (5.0)	0.0 (0.0)	2.2 (5.0)	9.5(13.2)
	NW	2.2 (5.0)	12.9(14.2)	7.7(12.0)	12.1 (9.7)	2.2 (5.0)	6.7 (6.1)	2.2 (5.0)	4.4 (6.1)

Table 3: Error rates for each combination of starting and ending octants. 5 users took part in this study, collecting a total of 40 data points per menu selection. All data in percent, standard deviation in parenthesis.

Appendix 2
PostBrainstorm study questionnaires

We reproduce here the questionnaires used for the PostBrainstorm study.

Digital Brainstorm Study Questionnaire (Leader)

Meeting ID:........................... Your ID .... Date:.....................................

Overall usability of the system:

9 How useful was the system during this brainstorming session?

NA ○ Not at all ○ ○ ○ ○ ○ ○ ○ Very much

10 How much did you enjoy using the system?

NA ○ Not at all ○ ○ ○ ○ ○ ○ ○ Very much

11 How useful will this system be for brainstorming on a regular basis?

NA ○ Not at all ○ ○ ○ ○ ○ ○ ○ Very much

Comments?...

....

....

12 How similar was the group dynamic to traditional brainstorming?

NA ○ Not at all ○ ○ ○ ○ ○ ○ ○ Very much

What was different?..

....

....

What was identical?..

....

....

13 For you, what are the strong points of the system?

....

....

....

....

14 For you, what are the weak points of the system?

....

....

....

....

Bringing information in

15 How useful was the scanner as an addition to the digital display?

NA ○ Not at all ○ ○ ○ ○ ○ ○ ○ Very much

Comments?..

....

....

16 How easy was it to use the overhead scanner?

NA ○ Not at all ○ ○ ○ ○ ○ ○ ○ Very much

Comments?..

....

....

....

17 Suggestions for improving access to pieces of information (sketches, objects, digital images, web access) needed during the meeting?

....

....

....

....

Managing items on the board

18 How useful was the thumbnail area at the top of the screen?

NA ○ Not at all ○ ○ ○ ○ ○ ○ ○ Very much

Comments?..

....

....

19 Was the board providing enough space to collect the pieces of information you needed?

NA ○ Not at all ○ ○ ○ ○ ○ ○ ○ Very much

Comments?..

....

....

20 How useful were containers (lists, scatter plot...)?

NA ○ Not at all ○ ○ ○ ○ ○ ○ ○ Very much

Comments?..

....

....

21 How useful were the voting and attribute features?

NA ○ Not at all ○ ○ ○ ○ ○ ○ ○ Very much

Comments?..

....

....

Post-brainstorm access to information:

22 How useful do you find the Word document representing the state of the board?

NA ○ Not at all ○ ○ ○ ○ ○ ○ ○ Very much

Comments?..

....

....

23 How useful will you find a tool to access and edit the brainstorm log on your workstation?

NA ○ Not at all ○ ○ ○ ○ ○ ○ ○ Very much

Comments?..

....

....

24 How useful will you find a Flash movie of the brainstorm?

NA ○ Not at all ○ ○ ○ ○ ○ ○ ○ Very much

Comments?..

....

....

25 How useful will you find a web page description of the state of the board at the end of the meeting (including sketches, images and handwriting recognition information)?

NA ○ Not at all ○ ○ ○ ○ ○ ○ ○ Very much

Comments..

....

....

Appearance of the Display

26 How distracting was the difference in color between tiles?

NA ○ Not at all ○ ○ ○ ○ ○ ○ ○ Very much

27 How distracting was the non-uniform brightness?

NA ○ Not at all ○ ○ ○ ○ ○ ○ ○ Very much

28 How pleasant was the appearance of sketch/writing?

NA ○ Not at all ○ ○ ○ ○ ○ ○ ○ Very much

Comments?..

....

....

Using the pen

29 How pleasant was the feel of the pen on the screen?

NA ○ Not at all ○ ○ ○ ○ ○ ○ ○ Very much

Comments?..

....

....

30 How responsive was the pen?

NA ○ Not at all ○ ○ ○ ○ ○ ○ ○ Very much

31 How accurate was the pen?

NA ○ Not at all ○ ○ ○ ○ ○ ○ ○ Very much

Comments? ....

....

....

32 How satisfactory was the board as a sketching tool?

NA ○ Not at all ○ ○ ○ ○ ○ ○ ○ Very much

Comments? ....

....

....

33 How accurate was the handwriting recognition?

NA ○ Not at all ○ ○ ○ ○ ○ ○ ○ Very much

Comments? ....

....

....

34 How comfortable was the command button?

NA ○ Not at all ○ ○ ○ ○ ○ ○ ○ Very much

Comments? ....

....

....

Menu system

35 How easy was it to learn the menu system?

NA ○ Not at all ○ ○ ○ ○ ○ ○ ○ Very much

Comments? ....

....

....

36 How easy was it to bring the menu up?

NA ○ Not at all ○ ○ ○ ○ ○ ○ ○ Very much

Comments?..

....

....

37 How easy was it to access the different functions (move, zoom, snap shot...)?

NA ○ Not at all ○ ○ ○ ○ ○ ○ ○ Very much

Comments?..

....

....

Previous experience

38 Did you use other electronic board for brainstorming before?

Which one?...

What were its shortcomings compared to this system?...

....

....

....

What were its strengths compare to this system?..

....

....

....

Miscellaneous

39 Did you need a tool not provided by the system during the session?

....

....

....

....

40 Others comments?

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

Thank you!

Thank you for participating in this study and taking the time answering these questions.

Digital Brainstorm Study Questionnaire (Participant)

Meeting ID:........................... Your ID.................................... Date:.......................................

Overall usability of the system:

1 How useful was the system during this brainstorming session?

NA ○ Not at all ○ ○ ○ ○ ○ ○ ○ Very much

2 How much did you enjoy using the system?

NA ○ Not at all ○ ○ ○ ○ ○ ○ ○ Very much

3 How useful will this system be for brainstorming on a regular basis?

NA ○ Not at all ○ ○ ○ ○ ○ ○ ○ Very much

Comments?...

....

....

4 How similar was the group dynamic to traditional brainstorming?

NA ○ Not at all ○ ○ ○ ○ ○ ○ ○ Very much

What was different?...

....

....

What was identical?...

....

....

5 For you, what are the strong points of the system?

....

....

....

....

6 For you, what are the weak points of the system?

....

....

....

....

Bringing information in

7 How useful was the scanner as an addition to the digital display?

NA ○ Not at all ○ ○ ○ ○ ○ ○ ○ Very much

Comments?...

....

....

8 How easy was it to use the overhead scanner?

NA ○ Not at all ○ ○ ○ ○ ○ ○ ○ Very much

Comments?...

....

....

9 Suggestions for improving access to pieces of information (sketches, objects, digital images, web access) needed during the meeting?

....

....

....

....

Managing items on the board

10 How useful was the thumbnail area at the top of the screen?

NA ○ Not at all ○ ○ ○ ○ ○ ○ ○ Very much

Comments?...

....

....

11 Was the board providing enough space to collect the pieces of information you needed?

NA ○ Not at all ○ ○ ○ ○ ○ ○ ○ Very much

Comments?...

....

....

12 How useful were containers (lists, scatter plot...)?

NA ○ Not at all ○ ○ ○ ○ ○ ○ ○ Very much

Comments?...

....

....

13 How useful were the voting and attribute features?

NA ○ Not at all ○ ○ ○ ○ ○ ○ ○ Very much

Comments?...

....

....

Post-brainstorm access to information:

14 How useful do you find the Word document representing the state of the board?

NA ○ Not at all ○ ○ ○ ○ ○ ○ ○ Very much

Comments?...

....

....

15 How useful will you find a tool to access and edit the brainstorm log on your workstation?

NA ○ Not at all ○ ○ ○ ○ ○ ○ ○ Very much

Comments?...

....

....

16 How useful will you find a Flash movie of the brainstorm?

NA ○ Not at all ○ ○ ○ ○ ○ ○ ○ Very much

Comments?..

....

....

17 How useful will you find a web page description of the state of the board at the end of the meeting (including sketches, images and handwriting recognition information)?

NA ○ Not at all ○ ○ ○ ○ ○ ○ ○ Very much

Comments..

....

....

Appearance of the Display

18 How distracting was the difference in color between tiles?

NA ○ Not at all ○ ○ ○ ○ ○ ○ ○ Very much

19 How distracting was the non-uniform brightness?

NA ○ Not at all ○ ○ ○ ○ ○ ○ ○ Very much

20 How pleasant was the appearance of sketch/writing?

NA ○ Not at all ○ ○ ○ ○ ○ ○ ○ Very much

Comments?..

....

....

Using the pen

21 How satisfactory was the board as a sketching tool?

NA ○ Not at all ○ ○ ○ ○ ○ ○ ○ Very much

Comments?..

....

....

Previous experience

22 Did you use other electronic board for brainstorming before?

Which one?...

What were its shortcomings compared to this system?..

....

....

....

What were its strengths compare to this system?...

....

....

....

Miscellaneous

23 Did you need a tool not provided by the system during the session?

....

....

....

....

24 Others comments?

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

Thank you!

Thank you for participating in this study and taking the time answering these questions.

Appendix 3
PostBrainstorm user study data

In this appendix, we report the result of the PostBrainstorm user study described in Section 7.2.

Questions (numbered as for the designer questionnaire (Appendix 2))		8 Leaders	16 Participants	All
1	How useful was the system during this brainstorming session	5.8 (0.7)	5.5 (0.6)	5.6 (0.7)
2	How much did you enjoy using the system	5.5 (0.9)	6.1 (0.8)	5.9 (0.9)
3	How useful will this system be for brainstorming on a regular basis	6.1 (0.6)	5.6 (0.9)	5.8 (0.8)
4	How similar was the group dynamic to traditional brainstorming	5.5 (1.3)	5.3 (1.2)	5.4 (1.2)
7	How useful was the scanner as an addition to the digital display	6.3 (1.5)	5.9 (1.8)	6.0 (1.6)
8	How easy was it to use the overhead scanner	6.2 (1.0)	6.1 (0.7)	6.2 (0.8)
10	How useful was the thumbnail area at the top of the screen	6.8 (0.7)	6.4 (0.8)	6.5 (0.8)
11	Was the board providing enough space to collect the pieces of information you needed	4.6 (1.5)	5.1 (1.4)	5.0 (1.4)
12	How useful were containers	4.9 (1.6)	5.3 (2.1)	5.1 (1.9)
13	How useful were the voting and attribute feature	4.0 (1.4)	3.8 (2.3)	3.9 (2.0)
14	How useful do you find the Word document	5.3 (1.5)	6.6 (0.5)	6.1 (1.1)
15	How useful will you find a tool to access and edit the brainstorm log on your workstation	6.0 (1.7)	6.0 (1.2)	6.0 (1.3)
16	How useful will you find a Flash movie of the brainstorm	2.2 (1.9)	3.4 (1.3)	2.9 (1.7)
17	How useful will you find a web page	5.8 (1.3)	4.9 (1.7)	5.2 (1.6)
18	How distracting was the difference in color between tiles	3.5 (1.5)	4.1 (1.5)	3.9 (1.5)
19	How distracting was the non-uniform brightness	4.0 (1.6)	4.1 (1.7)	4.1 (1.6)
20	How Pleasant was the appearance of sketch/writing	4.6 (1.7)	5.4 (1.4)	5.1 (1.5)
21	How pleasant was the feel of the pen on the screen	3.5 (1.6)	5.4 (1.3)	4.4 (1.7)
22	How responsive was the pen	4.3 (1.7)		4.3 (1.7)
23	How accurate was the pen	4.6 (1.4)		4.6 (1.4)
24	How satisfactory was the board as a sketching tool	4.5 (1.0)	4.2 (1.7)	4.2 (1.5)
25	How accurate was the handwriting recognition	3.4 (1.8)		3.4 (1.8)
26	How comfortable was the command button	3.3 (1.7)		3.3 (1.7)
27	How easy was it to learn the menu system	3.8 (0.7)		3.8 (0.7)
28	How easy was it to bring the menu up	5.9 (1.1)		5.9 (1.1)
29	How easy was it to access the different functions	4.9 (1.1)		4.9 (1.1)

Table 4: Quantitative results of the PostBrainstorm user study. For each question with a quantitative answer (from 1 to 7), the table next page show the average score for the 8 leaders, 16 Participants, and finally for all subjects. Standard deviations are shown in parenthesis.

Bibliography

[80/20] 80/20. *http://www.8020.net.*

[And94] Anderson, J. 1994. Learning and Memory: an integrated approach. (1994) John Wiley & Sons.

[Ano] Anoto. *http://www.anoto.com.*

[App87] Apple Computer 1987. Human Interface Guidelines: The Apple Desktop Interface. Addison-Wesley, 1987.

[Arv] Arvo, J. The smart board project: Human-computer interaction for direct manipulation of formal systems. *http://www.cs.caltech.edu/~arvo/projects/SmartBoard*.

[Atk88] Atkinson, K. 1988. An introduction to numerical analysis (2nd edition). John Wiley &Sons. 1988.

[Bar] BARCO Projection Systems. *http://www.barco.com/projection_systems.*

[BH94] Bederson, B., and Hollan, J. 1994. Pad++: A Zooming Graphical Interface for Exploring Alternate Interface Physics. *In Proc. UIST '94*, pp. 17-36.

[Bog86] Bogen, R. and the Computer-Aided Mathematics Group 1986. *Macsyma Reference Manua*l, 1986.

[BSP+93] Bier, E., Stone, M., Pier, K., Buxton, W. and DeRose, T. 1993. Toolglass and magic lenses. *In Proc SIGGRAPH'93*, pp. 73-80.

[Cam] Cambridge Display Technology. *http://www.cdtltd.co.uk.*

[Car81] Carlton, L. 1981. Processing Visual Feedback Information for Movement Control. *J. of Exp. Psychology: Human Perception and Performanc*e, 7(5). pp. 1019–1030.

[CD02] Davis, J., Chen, X. LumiPoint: Multi-User Laser-Based Interaction on Large Tiled Displays 2002. *Displays*, 22(1), (March 2002). To appear.

[Chr] Chrysler Design Award web site. *http://www.chryslerdesignawards.org.*

[CHWS88] Callahan, J., Hopkins, D., Weiser, M. and Shneiderman, B. 1988. An Empirical Comparison of Pie vs. Linear Menus. *In Proc. CHI'8*8, pp. 95–100.

[CMN83] Card, S., Moran, T. and Newell, A. 1983. The Psychology of Human-Computer Interaction. Lawrence Erlbaum Associates, 1983.

[Com] Compaq. *http://www.compaq.com.*

[CS91] Carr, R. and Shafer, D. 1991. The Power of Penpoint (1991).

[DL01] Dietz, P. and Leigh, D. 2001. DiamondTouch: A Multi-User Touch Technology. *In Proc. UIST '01* pp. 219-226

[EBG+92] Elrod, S., Bruce, R., Gold, R., Goldberg, D., Halasz, F., Janssen, W. Jr., Lee, D., McCall, K., Pedersen, E., Pier, K., Tang, J. and Welch, B. 1992. Liveboard: A Large Interactive Display Supporting Group Meetings, Presentations, and Remote Collaboration. *In Proc. CHI '92.* pp. 599–607.

[EFI] Electronics for Imaging. *http://www.e-beam.com*.

[EILM00] Edwards, K., Igarashi, T., LaMarca, A. and Mynatt, E. 2000. A temporal model for multi-level undo and redo. *In Proc. UIST'00*, pp. 31-40.

[FB95] Furnas, G. and Bederson, B. 1995. Space-scale diagrams: understanding multiscale interfaces. *In Proc CHI'95*, pp. 234-241.

[Fin] FingerWorks. *http://www.fingerworks.com.*

[FJHW00] Fox, A., Johanson, B., Hanrahan, P., and Winograd, T. 2000. Integrating Information Appliances into an Interactive Workspace, *IEEE Computer Graphics & Applications*, 20(3), (May/June 2000). pp 54-65.

[FL00] Funkhouser, T. and Li, K. (eds.) 2000. Onto the Wall: Large Displays. Special issue of *IEEE Computer Graphics and Applications*, 20:4 (July/August 2000).

[Fos98] Fossum, E. 1998. Digital camera system on a chip. *IEEE Micro*, 18(3) (May-June 1998), pp. 8-15.

[Ged98] Gedentyd, H. 1998. How designers work. *Ph.D. Thesis*, Lund University, Sweden, 1998.

[GOP+97] Gunn, C., Ortmann, A., Pinkall, U., Polthier, K., and Schwarz, U. 1997. Oorange: A virtual laboratory for experimental mathematics. Technical report, Sonderforschungsbereich 288, Differential Geometry and Quantum Physics, TU-Berlin, 1997.

[GR93] Goldberg, D. and Richardson, C. 1993. Touch-typing with a stylus. *In Proc. CHI'93*, pp. 80-87.

[GT00] Guimbretière, F. and Winograd, T. 2000. FlowMenu: Combining Command, Text, and Parameter Entry, *In Proc. UIST2000,* pp. 213-216.

[GW92] Gonzalez, R. and Woods, R. 1992. Digital Image Processing. Addison-Wesley, 1992.

[HBEH00] Humphreys, G., Buck, I., Eldridge, M., and Hanrahan, P. 2000. Distributed Rendering for Scalable Displays, *IEEE Supercomputing 2000*.

[HEB+01] Humphreys, G., Eldridge, M., Buck, I., Stoll, G., Everett, M. and Hanrahan, P. 2001. WireGL: a scalable graphics system for clusters. *In Proc. SIGGRAPH 2001*, pp. 129-140.

[HH99] Humphreys, G. and Hanrahan, P. 1999. A Distributed Graphics System for Large Tiled Displays, *In Proc IEEE Visualization* 1999, pp. 215–227.

[HJPS00] Hereld, M., Judson, I., Paris, J., and Stevens, R. 2000. Developing Tiled Projection Display Systems, *In Proc. of the Fourth International Immersive Projection Technology Workshop*, (June 2000).

[Hof96] Hoffman, J. 1996. VPS: a visual programming System and Image Synthesis Application. Technical report, Mathematical Sciences Research Institute, *http://www.msri.org/*, 1996.

[Hop91] Hopkins, D. 1991. The Design and Implementation of Pie Menus. *Dr. Dobb's Journa*l, 16(12) (Dec. 1991), pp. 16–26.

[IH00] Igarashi, T. and Hinckley, K. 2000. Speed-dependent automatic zooming for browsing large documents. *In Proc UIST'00*, pp. 139-148.

[ISH98] Igehy, H., Stoll, G. and Hanrahan, P. 1998. The Design of a Parallel Graphics Interface, *In Proc. SIGGRAPH 98*, pp. 141-150.

[IU97] Ishii, H. and Ullmer, B. 1997. Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms, *In Proc. CHI '97*, pp. 234-241.

[Jen] Jenmar Visual Systems. *http://www.jenmarvs.com.*

[Jen84] Jenks, R. 1984. The new SCRATCHPAD language and system for computer algebra. *In Proc. 1984 MACSYMA Users' Conference*, pp. 409–416.

[JF93] Jackiw, N. and Finzer, W. 1993. The geometer's sketchpad: Programming by geometry. In *Watch What I Do: Programming by Demonstratio*n, pages 293–308. The MIT Press, 1993.

[Joh00] Johnson, J. 2000. GUI Bloopers: Don'ts and Do's for Software Developers and Web Designers. Morgan Kaufmann, 2000.

[JRV+89] Johnson, J., Roberts, T., Verplank, W., Smith, D., Irby, C., Beard, M. and Mackey, K. 1989. The Xerox Star: a retrospective. *IEEE Computer*, 22(9), pp. 11-26, 28-29. Sept. 1989.

[JVC] JVC. *http://www.jvc.com*.

[KB91] Kurtenbach, G. and Buxton, W. 1991. Integrating mark-up and direct manipulation techniques. *In Proc UIST'91*, pp. 137-144.

[KB93] Kurtenbach, G. and Buxton, W. 1993. The Limits of Expert Performance Using Hierarchic Marking Menus. *In Proc. INTERCHI '93*, pp. 482-487.

[KFBB97] Kurtenbach, G., Fitzmaurice, G., Baudel, T. and Buxton, W. 1997. The design of a GUI paradigm based on tablets, two-hands, and transparency. *In Proc. CHI'97*, pp. 35-42.

[KL01] Kelley, T. with Littman, J. 2001. The Art of Innovation : Lessons in Creativity from Ideo, America's Leading Design Firm. Doubleday, 2001.

[KNF+01] Klemmer, S., Newman, M., Farrell, R., Bilezikjian, M., and Landay, J. 2001. The Designers' Outpost: A Tangible Interface for Collaborative Web Site Design. *In Proc. UIST '*01, pp. 1-10.

[KP88] Krasner, G. and Pope, S. 1988. A Cookbook for Using the Model-View-Controller User Interface Paradigm in Smalltalk-80. 1(3) (Aug/Sep 1988), pp. 26-41,48-49.

[Kur93] Kurtenbach, G. 1993. The Design and Evaluation of Marking Menus. *Ph.D. thesis*, University of Toronto, 1993.

[Kus] Kuska, J. MathView3D. *http://www.mathsource.com/Content22/Enhancements/Interfacing/Graphics/0209-056*.

[LC99] Li, K. and Chen, Y. 1999. Optical Blending for Multi-Projector Display Wall System. *Proc. 12th Lasers and Electro-Optics Society 1999.*

[LCC+00] Li, K., Chen, H., Chen, Y., Clark, D., Cook, P., Damianakis, S., Essl, G., Finkelstein, A., Funkhouser, T., Housel, T., Klein, A., Liu, Z., Praun, E., Singh, J., Shedd, B., Pal, J., Tzanetakis, G. and Zheng, J. 2000. Building and using a scalable display wall system, *IEEE Computer Graphics and Applications*, 20(4), pp. 29-37.

[Lev93] Levien, R. 1993. Highly sensitive register mark based on moiré patterns. *In Proc. SPIE Color Hard Copy and Graphic Arts II.* (1993).

[LJM98] Lange, B., Jones, M. and Meyers, J. 1998. Insight Lab: An Immersive Team Environment Linking Paper, Displays, and Data, *In Proc. CHI'98,* pp. 550-557.

[LS97] Lengyel, J. and Snyder, J. 1997. Rendering with coherent layers. *In Proc. SIGGRAPH'97*, pp. 233-242.

[MA98] Mankoff, J. and Abowd, G. 1998. Cirrin: A World-Level Unistroke Keyboard for Pen Input. *In Proc. UIST'9*8, pp. 213–214.

[Mat] MathSoft. *http://www.mathsoft.com.*

[MCM97] Moran, T., Chiu, P. and Van Melle, W. 1997. Pen-based interaction techniques for organizing material on an electronic whiteboard. *In Proc. UIST'9*7, pp. 45–54.

[MGH+97] Monagan, M., Geddes, K., Heal, K., Labahn, G. and Vorkoetter, S. 1997. *Maple V Programming Guide for Release* 5, 1997.

[MIEL00] Mynatt, E., Igarashi, T., Edwards, K. and LaMarca, A. 2000. Designing an Augmented Writing Surface, *IEEE Computer Graphics and Applications*, 20:4 (July/August 2000) 55-61.

[MIEL99] Mynatt, E., Igarashi, T., Edwards, K. and LaMarca, A. 1999. Flatland: New dimensions in office whiteboards. *In Proc. CHI '99,* pp. 346-353.

[Mim] Virtual Ink. *http://www.mimio.com/.*

[Mom91] Momenta: User's Reference Manual. Momenta, 295 North Bernardo Avenue, Mountain View, California.

[MRC91] Mackinlay, J., Robertson, G., and Card, S. 1991. The Perspective Wall: Detail and Context Smoothly Integrated. *In Proceedings of CHI '91*, pp. 173-179.

[MSM+99] Moran, T., Saund, E., Van Melle, W., Gujar, A., Fishkin, K. and Harrison, B. 1999. Design and Technology for Collaborage: Collaborative Collages of information on Physical Walls, *In Proc. UIST99*, pp. 197-206.

[NDV+91] Nunamaker, J., Dennis, A., Valacich, J., Vogel, D. and George, J. 1991. Electronic Meeting Systems To Support Group Work. *CACM*, 34(7), (July 1991), pp. 40-61.

[New] Newton. Apple MessagePad Handbook. Apple Computer 1995.

[NVI] NVIDIA. *http://www.nvidia.com.*

[Nye95] Nye, A. 1995. Xlib Programming Manual. O'Reilly & Associates, Inc. 1995.

[ON63] Oster, G. and Nishijimo, Y. 1963. Moiré Patterns. *Scientific American* (May 1963), pp. 54-63.

[Ope] OpenGL Specifications. *http://www.opengl.org/Documentation/Specs.html.*

[Pac] Pacific Tech. *http://www.nucalc.com.*

[Pal] Palm OS. *http://www.palmos.com.*

[Pal99] Palmer, S. 1999. Vision science - Photons to phenomenology. MIT Press 1999.

[Par] Paragraph Corp. Calligrapher. *http://www.paragraph.com.*

[Per93] Perlin, K. and Fox, D. 1993. Pad: an alternative approach to the computer interface. *In Proc. SIGGRAPH'93*, pp. 57-64.

[Per98] Perlin, K. 1998. Quikwriting: Continuous Stylus-Based Text Entry. *In Proc. UIST'9*8, pp. 215–216.

[PLD01] Paulus, P., Larey, T. and Dzindolet, M. 2001. Creativity in Groups and Teams. *In Groups at Work: Theory and Research,* Turner, M. Editor. Lawrence Erlbaum Associates 2001.

[PLM93] Phillips, M., Levy, S. and Munzner, T. 1993. Geomview: An interactive geometry viewer. *Notices of the American Mathematical Societ*y, 40(8) (October 1993), pp. 985–988.

[PLVB00] Pook, S., Lecolinet, E., Vaysseix, G. and Barillot E. 2000. Control Menu: Execution and Control in a Single Interactor. *In Extended Abstracts of CHI2000*, pp. 263–264.

[PMMH93] Pedersen, E., McCall, K., Moran, T. and Halasz, F. 1993. Tivoli: An electronic whiteboard for informal workgroup meetings. *In Proc. INTER-CHI'9*3, pp. 391-398.

[Rai] Rainbow Displays. *http://www.rainbowdisplays.com.*

[Ras00] Raskin J. 2000. The humane interface: new direction for designing interactive system. ACM Press 2000.

[RBJW01] Ringel, M., Berg, H., Jin, Y., and Winograd T. 2001. Barehands: Implement-Free Interaction with a Wall-Mounted Display, *In Extended Abstracts of CHI2001*, pp. 367-368.

[RBY+99] Raskar, R., Brown, M., Ruigang Y., Wei-Chao C., Welch, G., Towles, H., Scales, B. and Fuchs, H. 1999. Multi-projector displays using camera-based registration. *In Proc IEEE Visualization'99*, pp. 161-168.

[RCM89] Robertson, G., Card, S., and Mackinlay, J. 1989. The cognitive coprocessor architecture for interactive user interfaces. *In Proc. UIST'89*, pp. 10-18.

[RCM93] Robertson, G., Card, S. and Mackinlay, J. 1993. Information visualization using 3D interactive animation. *Communications of the ACM*, 36(4) (April 1993), pp. 57-71.

[RL00] Rusinkiewicz, S. and Levoy, M. 2000. QSplat: A Multiresolution Point Rendering System for Large Meshes, *In Proc. SIGGRAPH 2000*, pp. 343-352.

[Rog83] Rogowitz, B. 1983. The Human visual System: A guide for the Display Technologist. *In Proc. of the SID* 24(3), (1983), pp. 235-252.

[RS99] Rekimoto, J. and Saitoh, M. 1999. Augmented surfaces: a spatially continuous work space for hybrid computing environments. *In Proc CHI'99*, pp. 378-385.

[RSWH98] Richardson, T., Stafford-Fraser, Q., Wood, K. and Hopper, A. 1998. Virtual Network Computing, *IEEE Internet Computing*, 2(1), (Jan/Feb 1998), pp. 33-38.

[Sau99] Saund, E. 1999. Image Mosaicing and a Diagrammatic User Interface for an Office Whiteboard Scanner, Technical Report, Xerox Palo Alto Research Center, 1999.

[SB99] Stupp E. and Brennesholtz, M. 1999. Projection displays. John Wiley & Sons 1999.

[SBF+86] Stefik, M., Bobrow, D., Foster, G., Lanning, S. and Tatar, D. 1986. WYSIWIS Revised: Early experiences with multi-user interfaces. *In proc. CSCW'86*, pp. 276-290.

[Sch83] Schön, D. 1983. The Reflective Practitioner. Basic Books, Inc. 1983.

[SFB+87] Stefik, M., Foster, G., Bobrow, D., Kahn, K., Lanning, S. and Suchman, L. 1987. Beyond the chalkboard: Computer Support for Collaboration and Problem Solving in Meetings. *Communications of the ACM,* 30(1), (Jan 1987), pp. 32-47.

[Shn83] Shneiderman, B. 1983. Direct manipulation: A step beyond programming languages, *IEEE Computer* 16(8), (August 1983), pp. 57-69.

[SKB92] Sellen, A., Kurtenbach, G. and Buxton, W. 1992. The prevention of mode errors through sensory feedback. *Human-Computer Interactions*, 7(2), (1992), pp. 141-64.

[Sma] Smart Technologies, *http://www.smarttech.com*.

[SRS+93] Sears, A., Revis, D., Swatski, J., Crittenden, R. and Shneiderman, B. 1993. Inverstigating tochscreen typing: The effect of keyboard size on typing speed. *Behaviour & Information Technology,* 12(1) ,(1993), pp. 17-22.

[Sto01] Stone, M. 2001. Color and brightness appearance issues in tiled displays. *IEEE Computer Graphics and Applications*, 21(5) (Sept./Oct. 2001), pp. 58-66.

[Sto01a] Stone, M. 2001. Color Balancing Experimental Projection Displays. *In Proc of the 9th Color Imaging Conference: Color Science, Systems and Applications 2001*, pp. 342-347.

[Sur99] Surati, R. 1999. Scalable Self-Calibrating Display Technology for Seamless Large-Scale Displays. *Ph.D. thesis*, MIT 1999.

[Sut63] Sutherland, I. 1963. SKETCHPAD: A Man-Machine Graphical Communication System. *Ph.D. thesis*, MIT, 1963.

[Tes81] Tesler L. 1981. The Smalltalk Environment. BYTE, 6(8), (Aug. 1981), pp. 90-147.

[The] Theorist Interactive (now called LiveMath). *http://www.livemath.com*.

[TI] Texas Instruments, *http://www.dlp.com/dlp.*

[Tuf91] Tufte, E. 1991. Envisioning Information. Graphics Press, 1991.

[VN94] Venolia, D. and Neiberg, F. 1994. T-Cube: A Fast, Self-Disclosing Pen-Based Alphabet. *In Proc. CHI'94*, pp. 265–270.

[Wel93] Wellner, P. 1993. Interacting with paper on the DigitalDesk. *Communications of the ACM,* 36(7), (July 1993), pp. 86-96.

[Win01] Winograd, T, 2001. Architectures for Context. *Human-Computer Interaction*, 16:2-3, (2001). To appear.

[Wol93] Mathlink Reference Guide Version 2. Wolfram Research 1993.

[Wol96] Wolfram, S. 1996. The Mathematica Book (3rd edition). Cambridge University Press, 1996.

[Yen00] Yen, S. 2000. Capturing Multimodal Design Activities in Support of Information Retrival and Process Analysis. *Ph.D. Thesis*, Stanford University, 2000.