

# Whiteboard Computing: Towards A Sketch-Centric Operating Environment

Ryan Dixon and Timothy Sherwood  
Department of Computer Science  
University of California, Santa Barbara  
Santa Barbara, CA 93106-9560  
{rsd,sherwood}@cs.ucsb.edu

## ABSTRACT

The simplicity and accessibility of whiteboards provides an appealing avenue for sharing ideas and solving problems. Yet even as these surfaces sit at the center of computer-intensive work environments, whiteboards are largely relegated to serving as note-taking devices. We present our concept of whiteboard-based computing, highlight current hardware and software trends that have laid the groundwork for whiteboard-based systems, and describe our efforts in creating a whiteboard computing framework.

## 1. MOTIVATION

Whiteboards offer a simple and powerful means of describing and communicating complex ideas. Yet, while their utility as an integral problem solving tool is recognized across many disciplines, the whiteboard largely remains a static device with only limited integration into our current computer-based workflows. Our goal is to create the next generation whiteboard system whereby free-form input is used to not only draw and share static images, but also perform dynamic computations. It is our belief that current hardware and software trends support our vision.

With recent advances in multi-touch technology, a new market for devices larger than traditional computer displays appears to be emerging [3, 6]. As electronic displays continue to double in size approximately every year and a half, we are nearing a convergence in the landscape of desktop computing and wall-sized human-computer interaction [2]. While existing windows, mouse, and pointer interfaces are being adapted to better suit these changing demands, there is significant potential for new methods of interaction. Given the current trends, it is evident that both hardware and software will play a critical role in the evolution of this technology.

Over the past decade, numerous handcrafted, domain-specific software systems have been designed to handle free-form input [1, 4]. The first generation of prototypes is just now reaching a product-ready level of maturity; however, there is little to no coherence between any of the individual projects.

Ultimately, we desire to harness and combine these trends into a system that can compute upon free-form input from any number of domains without sacrificing the freedom provided by a conventional whiteboard. We believe this presents a number of new challenges to architect, system, and language designers alike, and we have started to take the necessary steps towards making this a reality.

## 2. WHITEBOARD TRAFFIC

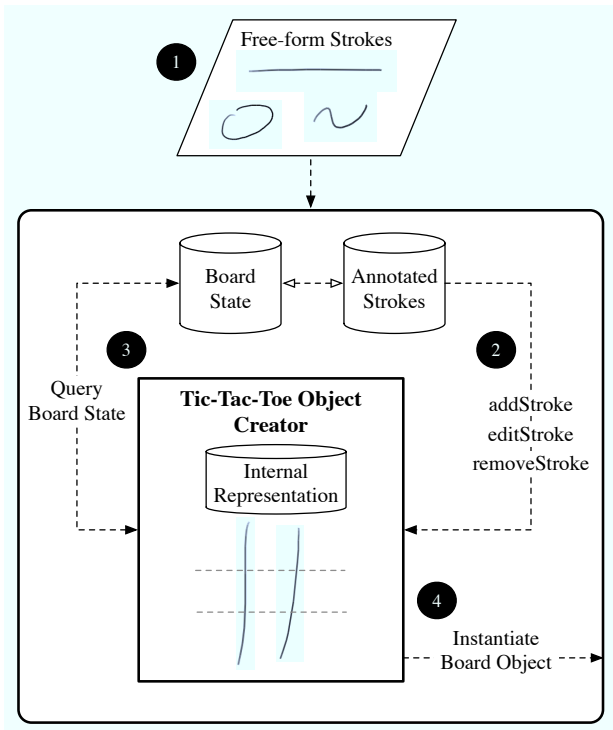
To better understand the data requirements of whiteboard systems, using an eBeam capture device [5], we have collected roughly four months worth of input on our laboratory whiteboard. Our experiment does no recognition, instead it passively records statistics from the data points that are naturally written by our lab for computer architecture research so that we can understand the distribution of inputs that might be given to recognition systems. Beyond spatial coordinates, the raw data captured includes timing information, marker color, and marker size.

Over the survey period, approximately 10-15 lab members had access to the whiteboard. The sample period, spanning from November 19, 2007 to March 17, 2008, includes portions of Winter and Spring quarter, including a significant holiday break in between. Of the 120 days of the survey, 49 days (approximately 41%) contained board activity. The combined days of board activity produced over 300,000 input points and 11,420 input strokes. An input stroke is generated every time an input pen touches the whiteboard surface. A stroke accumulates all point data captured until the pen is lifted from the surface of the whiteboard. Our entire data collection consists of strokes composed of up to hundreds of points each. All of the observed traffic was consistently bursty, often occurring within the span of an hour, followed by hours of inactivity.

Throughout the duration of the survey, we also designed tools to help parse, interpret, and understand the whiteboard data. From this work, we have created the beginning of a whiteboard computing environment, where multiple concurrent applications are allowed to run and interact with free-form sketches and perform live computations.

## 3. WHITEBOARD FRAMEWORK

Our current work is focused on developing a whiteboard environment where users can freely sketch and view computations on-the-fly. At a high level, the board operating environment is a lightweight event-based system that is responsible for managing board real-estate, and access to annotated stroke data. Figure 1 illustrates the current board architecture and four of the distinct phases outlined below. In its initial state, a board hosts a set of board objects that will be responsible for parsing and interpreting all incoming stroke data. These objects largely define the functionality and behavior of the board.

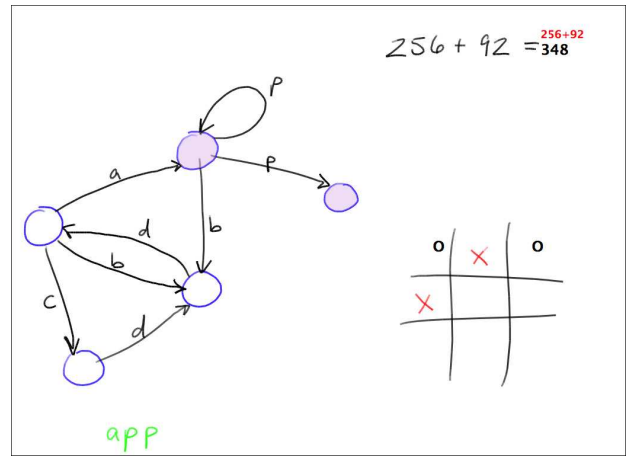


**Figure 1: The board object event loop: (1) Stroke input, (2) Event processing, (3) Querying board state, and (4) Board object instantiation.**

As a user draws, the board is responsible for collecting and storing representative stroke data. An initial pass is made over the data to annotate specific features that will likely be relevant to a number of board applications. These annotations might include features such as: fragmentation, where complex strokes are separated into simpler component pieces; beautification, where overlapping strokes are refined into a single representative stroke; and even character recognition. The initial board annotation phase enables the board to support gestures. Currently, only the scribble gesture is supported to enable full-stroke erasure.

After the initial annotation phase is complete, each of the board objects is presented an event, indicating that a new stroke has been added to the board. It is then up to each individual board object to determine what significance that new stroke has with respect to its own state of computation. A board object can also query the board for an up-to-date view of board strokes. Lastly, a board object has the potential to create and add new board objects to the board. In the case of Tic-Tac-Toe, a board object constantly monitors the board for the appearance of a Tic-Tac-Toe game. Once the necessary hash marks have been drawn, the board object instantiates a new Tic-Tac-Toe game.

Our design approach can, in many ways, be viewed as a generalization of the spreadsheet; portions of the board may interact and affect other regions of computation. As soon as a stroke is drawn, changes are immediately presented to the user. Figure 2 provides a sample of the applications we



**Figure 2: An example of three concurrent whiteboard applications: an equation solver, Tic-Tac-Toe, and a finite state machine.**

have built thus far. These three applications exemplify the intended interaction with the whiteboard. For example, as input is written for the the finite state machine (the bottom text in Figure 2), the active states are automatically updated and highlighted. Likewise, when edges and nodes are added or removed from the state machine, changes are reflected immediately in the new graph.

Our work is still very much in the preliminary stages. Although we have not yet performed any definitive tests on the system, early user feedback has been positive and we believe our initial efforts are a step in the right direction.

## 4. CONCLUSIONS

This work presents challenges across many domains. User interface decisions affect the stroke recognition process which, in turn, affects recognition accuracy and system performance. Many questions pertaining to system design remain open. It is our hope to not only provide a viable solution to the whiteboard computing problem, but also provide a starting point for future development in this area. We hope to inspire a combined effort between architecture, system, user interface, and language design experts to help further develop this idea.

## 5. REFERENCES

- [1] C. Alvarado and R. Davis. Dynamically Constructed Bayes Nets for Multi-Domain Sketch Understanding. In *IJCAI '05*, August 1 2005.
- [2] R. Dixon and T. Sherwood. Whiteboards that Compute: A Workload Analysis. In *IISWC '08*, 2008.
- [3] J. Y. Han. Low-Cost Multi-Touch Sensing Through Frustrated Total Internal Reflection. In *UIST '05*. ACM, 2005.
- [4] J. J. Laviola and R. C. Zeleznik. MathPad2: A System for the Creation and Exploration of Mathematical Sketches. *ACM Trans. Graph.*, August 2004.
- [5] Lucidia. eBeam - Interactive Whiteboard Technology, 2008.
- [6] Microsoft. Microsoft Surface - Surface Computing, 2008.