Whiteboards

Umesh Maheshwari (umesh@lcs.mit.edu)

M.I.T. Laboratory for Computer Science
Cambridge, MA 02139

1 Introduction

We present a design for electronic whiteboards. Multiple users can open the same board and see the edits made by others in real time. Editing includes printing text and drawing figures. The extent of these boards is virtually infinite. Different users can view different parts of a board at different levels of detail, and they can zoom or pan their views smoothly. Such a system is useful for collaborative architectural drawings, bulletin boards, year-planners, etc.

Boards are managed by a server, which stores them persistently and allows multiple clients to share them. The client program presents an interface to the user to view and edit the boards. An important goal for such a system is to scale in the number of items per board and the number of clients viewing a board.

Whiteboards have been implemented in Modula-3, using Network Objects [BOW93], the Trestle window toolkit, the threads package, and another package for logging and recovery. The implementation largely conforms to the design presented here.

2 The Interface

The interface allows a user to create, open, close, or delete a board. A board is identified by a pathname in the filesystem, which is interpreted by the server as the directory that stores the contents of the board. When a board is opened, the user sees the items drawn on the board through a window. The user can zoom or pan this window smoothly to look at the desired part of the board at the desired level of detail. (A mouse interface similar to the controls in video cameras is provided. Figure 1 shows a board with a year planner drawn on it. The user can zoom onto any month or date by positioning the pointer and holding down a mouse button.)

The user can also edit the board by creating, modifying, or deleting items, as with drawing packages such as Idraw. When a user makes an edit, it appears in real time on other windows opened on the same board. The edit appears at the proper place and size depending on the position of the windows and may even be hidden in some windows. For example, when a user types text, each keystroke is visible on other windows and the font is appropriately scaled to match the level of detail.

Picture to be pasted here.

Figure 1: The user interface of a board client.

Like files, boards persist across open's and close's. The user need not save a board explicitly; with high probability the edits survive client and server crashes.

This research was guided by Dr. Greg Nelson. The work was done in the summer of 1993 while the author was an intern at DEC, Systems Research Center, Palo Alto.

3 The Internals

A board is implicitly associated with a coordinate system with floating-point coordinates. The range and precision of floating-point numbers limits zooming and panning on the board. The client window for viewing boards has a separate coordinate system with the origin at, say, its bottom left corner and a scale of one per pixel. A point on the board can be converted between the board and window coordinates by knowing the window's *offset* — the position of window's origin in board coordinates, and the *scale* — the number of window units per board unit. The offset and the scale of a window together form the window's *focus*.

Items store their size information in board coordinates. Each item has a paint method that uses the window's focus to draw the item in the right place and size. For example, a text item stores its size in board coordinates and then scales the size by the window's scale; if the resultant size is too small, it is drawn as lines. Each item also knows its *bounding box*: a rectangle that bounds the item. The bounding boxes are used to optimize updates as explained later.

When a client opens a board, it registers callback routines with the server so that the server can notify it of edits made by others. At any time, a client caches only items visible through a set of rectangles, called the client's *sight* (see Figure 2). The server knows the sight of each client and keeps the client up-to-date on the items in its sight.

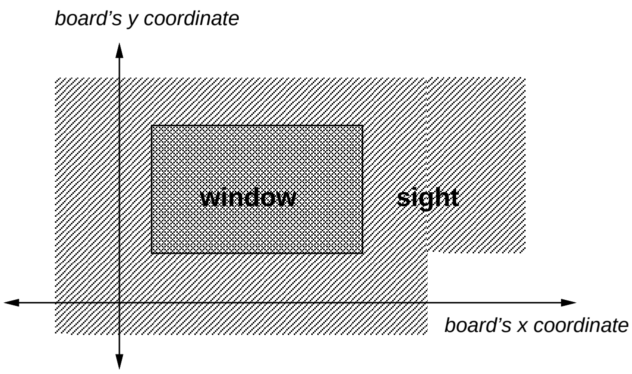

Figure 2: The window is a subset of the sight.

When a window is zoomed or panned, the client need not access the server as long as the window lies within its sight. If the window extends beyond the sight, the client expands its sight to cover the window and informs the server. When the server finds that the sight of a client has expanded, it sends the client the items not included in the old sight.

The client usually expands its sight in big steps so that it does not have to access the server too frequently, *e.g.*, when the window is being panned continuously. This is prefetching. To reduce the load the client can also shrink its sight, say, based on an LRU scheme. This is eviction. (The current client implementation does not prefetch or evict.)

When a client edits an item, it sends the update to the server, which logs the update in main memory. A server thread propagates the update to other clients in the background so that the modifying client is not delayed. Only those clients are notified who have opened the board and whose sights overlap with the old or the new bounding box of the modified item. If the server cannot contact a registered client, it simply discards all state about the client.

The server flushes log updates to the disk in the background: the updates are appended to a file in the directory identified by the board name. Periodically, a snapshot file of the board is stored on the disk and the log file is truncated.

Besides the lazy updates by the server, no cache coherence is provided to the clients. The clients modify items without locking. This implies that if the same item is modified by different users at about the same time, the two may receive the modifications in the reverse order. In the current implementation, modifications are limited to short actions and updates are propagated quickly; so incoherence was not considered a serious practical problem.

4 Limitations

Most limitations of whiteboards stem from the current implementation rather than any basic design flaw. Currently, there is no way to convert the stable snapshot of a board into standard formats such as PostScript. There is no access control: the server uses its own effective uid in accessing board files. As mentioned, there is no concurrency control for near-simultaneous updates to the same item. Fonts do not scale beyond one inch in order not to overload the X server. The range and precision of floating point numbers limits the focus of the windows.

References

[BOW93] A. Birrell, S. Owicki, and E. Wobber Network Objects. In em Proceedings of the Fourteenth ACM Symposium on Operating Systems Principles, 1993.