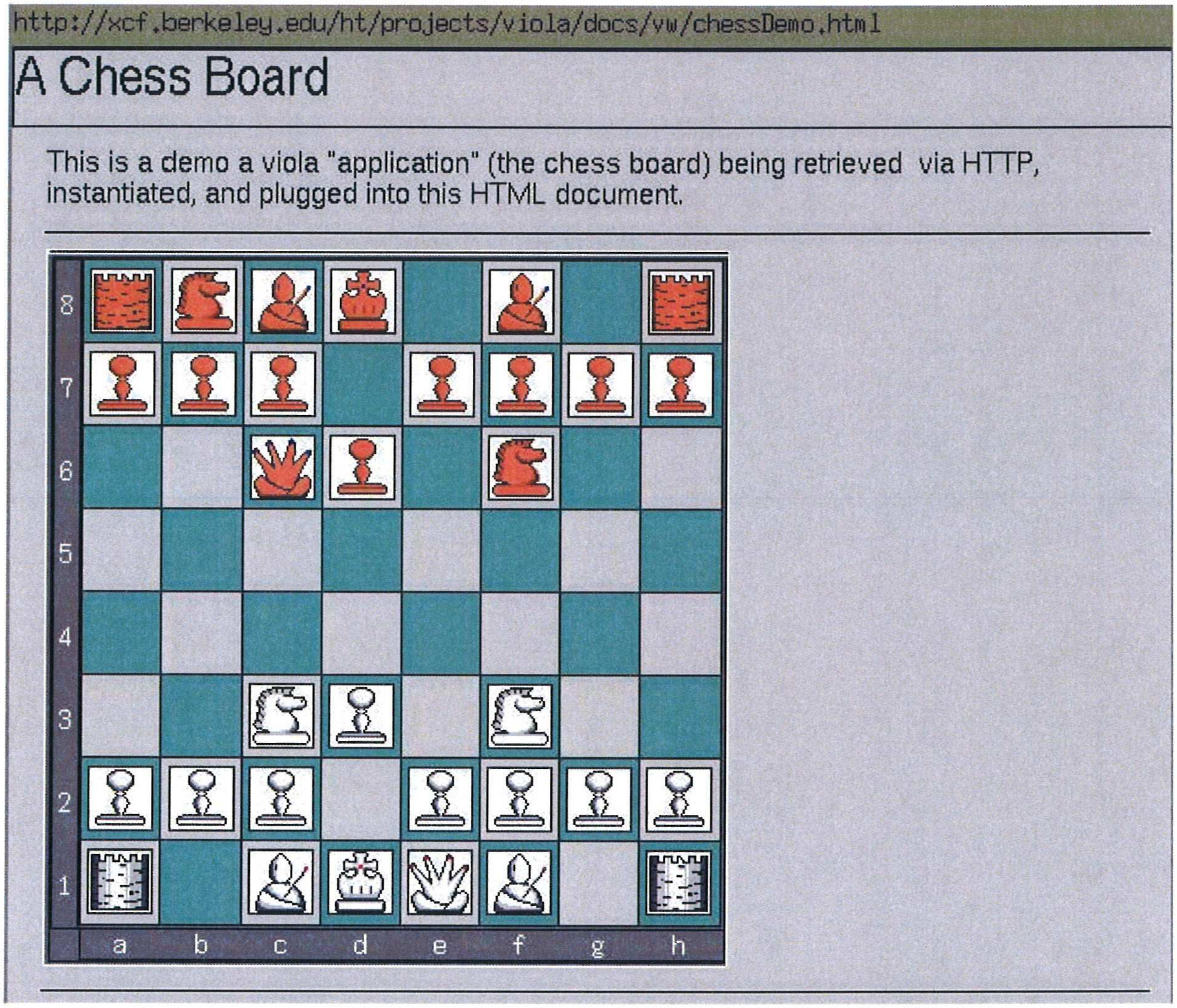
intelligence of the scripts in the application. Given more work, this chess board application can front-end a chess server, connected to it using the socket facility in viola.



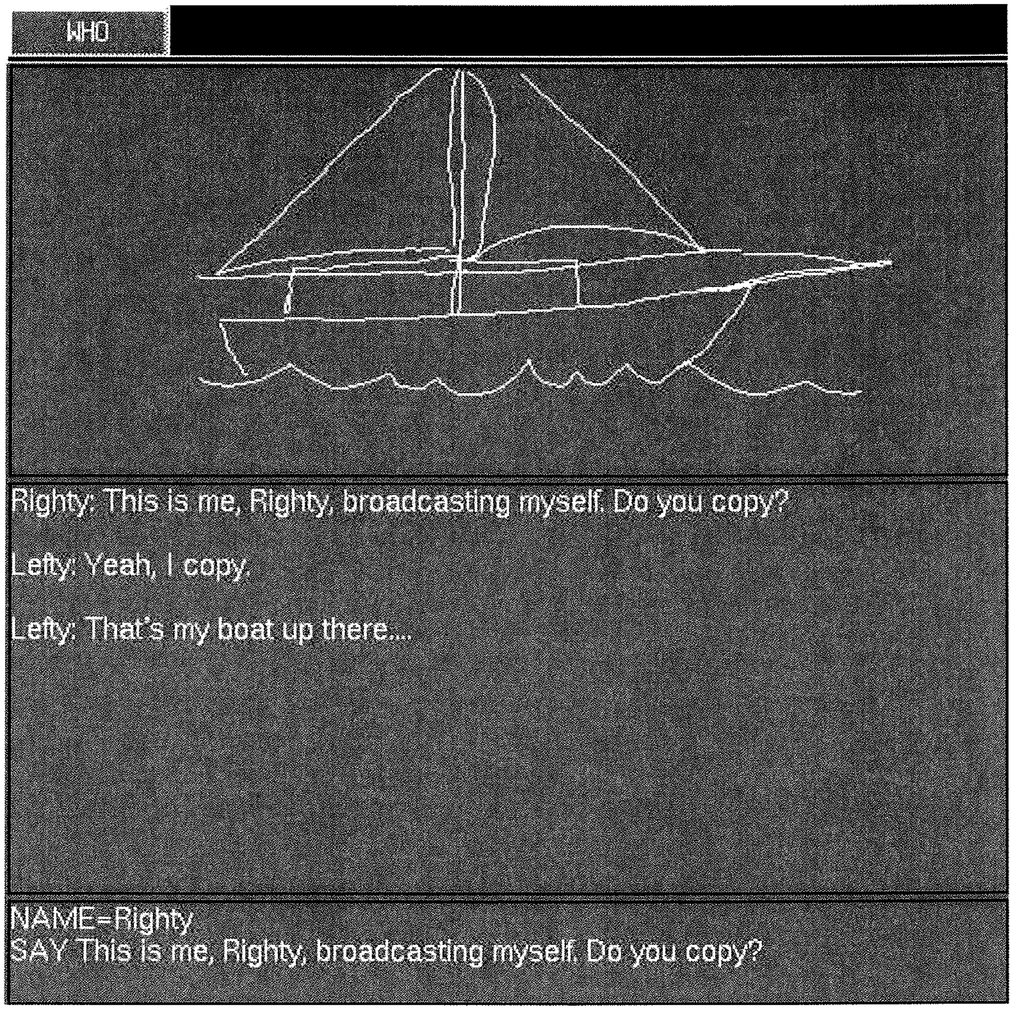
http://xcf.berkeley.edu/ht/projects/viola/docs/vw/chessDemo.html

A Chess Board

This is a demo a viola "application" (the chess board) being retrieved via HTTP, instantiated, and plugged into this HTML document.

What follows is a screendump of a demo of an embedded viola application that lets readers of this HTML page communicate by typing or drawing. Like the chess board application above, this chat application can stand-alone (and have nothing to do with the World Wide Web), or be embedded into a HTML document.

By the way, to make this possible, a multi-threaded/persistent server was written to act as a message relay (and to handle HTTP as well).



Righty: This is me, Righty, broadcasting myself. Do you copy?

Lefty: Yeah, I copy.

Lefty: That's my boat up there....

NAME=Righty
SAY This is me, Righty, broadcasting myself. Do you copy?

This next mini application front-ends a graphing process (on the same machine as the viola process). An important thing to note is that, like all the other document-embeddable mini applications shown, no special modification to the viola engine is required for ViolaWWW to support them. All the bindings are done via the viola language, provided that the necessary primitives are available in the interpreter, of course.

Put it another way, because of the scripting capability, the ViolaWWW browser has become very flexible, and can take on many new features dynamically. C-code patches and re-compilation of the browser can frequently be avoided.

This attribute can be very important for several reasons. It keeps the size of the core software small, yet can grow dynamically as less frequently used features are occasionally used, or as new accessories/components are added.