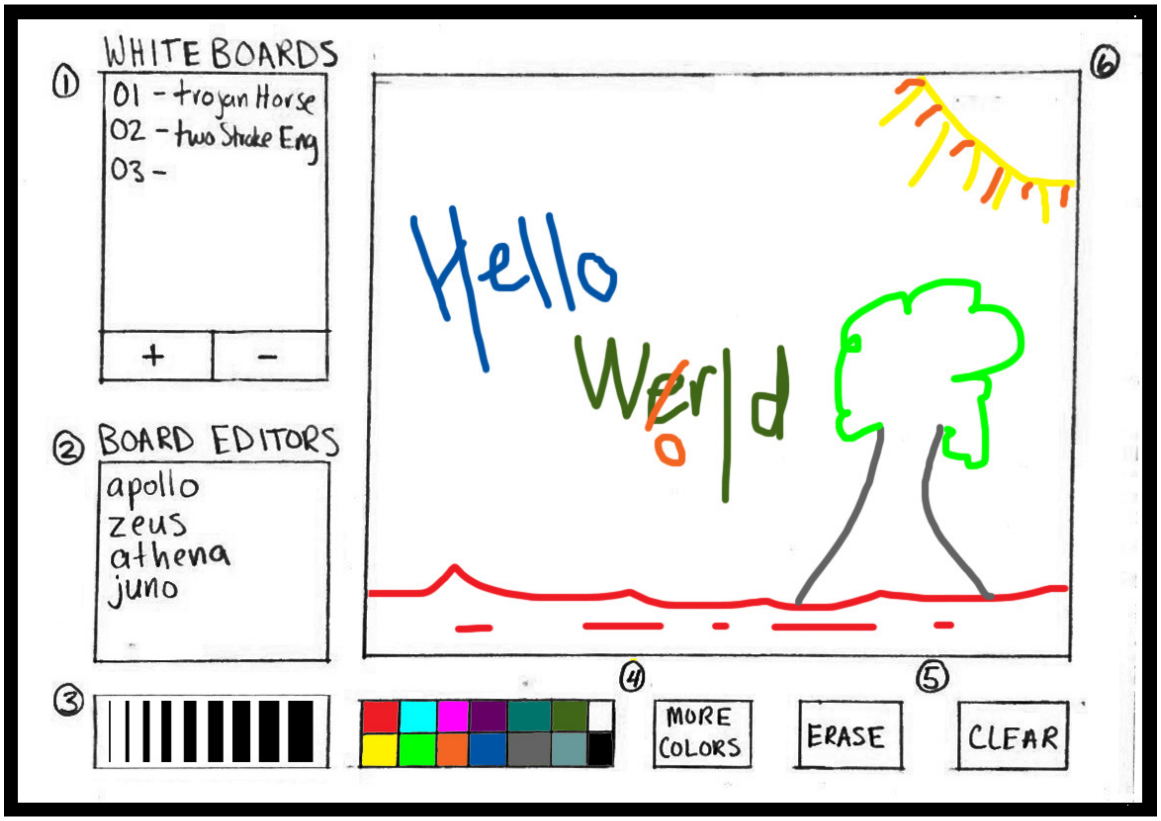
# Design Milestone

Andre Aboulian, Cathleen Gendron, & Jon Beaulieu

6.005 Software Construction - Fall 2013 - Project 2: "Collaborative Whiteboard"

## I. User Functionality Overview

## I. Components



### I.1 Selector

The board selector in the left pane includes a list of all current whiteboards and appears the same for all users. Each line represents an individual board, which are numbered sequentially and named by the user. Upon clicking the "+" button, the user will be prompted to name the new board. When the board has been created on the server, it will be appended to the list for each user. Selecting a board in the list will download the board from the server, overwrite the local copy if one exists, and display the board in the canvas window.

### I.2 Board Editors

Displays a list of users, including the viewer, who are currently modifying the selected board. This list will be updated as users enter and exit the board.

### I.3 Thickness Selector

This tool allows the user to select a brush/eraser thickness for drawing on the whiteboard.

### I.4 Color Selector

The main color palette displays a grid of colors from which the user can choose to paint with. The color currently in use will be highlighted. Clicking the "more colors" button will open Swing's built-in color chooser, which will offer a larger selection of colors.

### I.5 Erasing Tools

The erase button will allow the user to toggle between erasing and painting. "Erasing" will be defined as drawing with a white selection. Erasing will happen in the same order as drawing, so whichever request reaches the server first will erase all that has been drawn under it. Toggling back to painting will restore the user's previous color choice.

### I.6 Whiteboard Window

Displays the currently selected whiteboard, including all of its drawn strokes and erasures. The whiteboard be real-time interactive to allow users to collaborate simultaneously. Edits will be made in the order that modifications reach the server. In other words, a stroke logged on the server at a specific instant will be drawn over any strokes drawn before that instant.

## II. Behavior

**Erasing** TEXT

**Editing a Deleted Board** TEXT

## II. Server-Client Communication

## I. Protocol

### I.1 Grammar

The following grammar will facilitate the text-based communication between the clients and the server. The server will send `StoC_MSG` messages to the client, which will be able to send `CtoS_MSG` messages back to the server.

```
StoC_MSG :== (STROKE | BRD_INFO | BRD_DEL | USER_INIT | BRD_USERS) N

CtoS_MSG :== (STROKE | SEL | BRD_REQ | BRD_DEL | BRD_ALL | USER_REQ) N

STROKE :== "stroke" S BOARD_ID S THICK S COORDS S COLOR
COORDS :== X1 S Y1 S X2 S Y2
```

```
X1, Y1, X2, Y2 :== INT
COLOR :== [0-255] S [0-255] S [0-255]
THICK :== [1-10]

SEL :== "select" S BOARD_ID

BRD_REQ :== "board_req" S NAME
BRD_ALL :== "board_all"
BRD_INFO :== "board" S BOARD_ID S NAME
BRD_DEL :== "del" S BOARD_ID
BRD_USERS :== "board_users" S BOARD_ID (S USER_NAME)+

USER_REQ :== "user_req" S USER_NAME
USER_INIT :== "you_are" S USER_NAME

NAME :== [^N]
USER_NAME :== [A-Za-z]([A-Za-z0-9]?)+
BOARD_ID :== INT

INT :== [0-9]+
N :== "\r?\n"
S :== " "
```

### I.2 Usage

**Adding Users** Upon entering a username in the client application, a `USER_REQ` message will be sent to the server to request the desired username. (Note that regex checking for `USER_NAME` occurs on the client side before this request is made.) The server responds with a `USER_INIT` message, signifying the acquired username for the client. If there is a username conflict, one is chosen for the client. When the client is ready to accept information about existing boards, it calls `BRD_ALL` to begin receiving `BRD_INFO` messages for all previously created boards.

**Adding Boards** When a client wants to create a new whiteboard, it sends a `BRD_REQ` request to the server with a desired `NAME` (duplicate names allowed). Once the server has initialized a new internal board object, it sends a `BRD_INFO` message to all connected users to inform them of the newly available board. Note that the `BOARD_ID` used is a number unique to each whiteboard and is never reused. This is a different number than the sequential board numbering in the GUI, although the order is preserved.

**Removing Boards** Deleting boards entails a process similar to adding them. A client sends a `BRD_DEL` request to the server, which forwards this requests to all other users. The server internally disassociates all connected users and removes the board, taking care to ignore drawing requests and selection requests for this deleted board.

**Selecting Boards** Upon selecting a different board, the client sends a `SEL` request to the server. The server clears all stroke messages queued to update the client's whiteboard before associating the requested whiteboard object to the user, if available. The `SEL` command also requests all previously drawn strokes to be sent to the client. A `BRD_USERS` messages is sent to all users of the previous and current whiteboard to inform them of this change in editors.

**Disconnecting Users** When a client disconnects – by either severing the connection or closing the client application – the server closes the associated socket and streams, disassociates the user from its whiteboard, and removes the user from its main users list. A `BRD_USERS` message is sent to all users of the board that was being edited.

**Drawing Strokes** When a client draws a stroke in the selected whiteboard, a sequence of `STROKE` messages are sent to the server. The corresponding lines are logged as drawn in the order they are received by the server. The server proceeds to forward the `STROKE` messages to all users editing the same whiteboard, including the user who made the edit. Since the server sends the `STROKE` updates in the order they were made, the user's stroke will be covered by the echoed version from the server. This will not be visually apparent and will ensure that concurrently drawn lines appear constant across clients.

## II. Data Transport

### II.1 New Connections

The Whiteboard server will maintain a background thread that listens for new connections. Upon accepting a new socket, a new thread prompts the client for a username, and then instantiates a new User, which is then added to the ArrayList of all Users.

### II.2 Request Streams

GUI: The GUI maintains two threads, one which listens for requests from the associated User, and another which contains a BlockingQueue containing requests to be sent to the User. Possible requests include creating a new Whiteboard, deleting a Whiteboard, selecting a different Whiteboard, or sending a Stroke.

User: Each User object holds the client-side socket. The User maintains two threads, one which listens for requests from either its MasterBoard or its GUI, and another which contains a BlockingQueue containing requests to be sent to the GUI. The request() method parses GUI requests and sends the appropriately formatted text protocol message to either the MasterBoard (when a Stroke is made) or the WhiteboardServer (for other User/Whiteboard requests).

MasterBoard: The MasterBoard maintains two threads, one which listens for requests from any of its associated Users, and another which contains a BlockingQueue containing requests to be sent to all of its associated Users. For example, the listening thread will accept a request to create a new Stroke to be added to its strokes ArrayList, and then send that updated ArrayList to all its Users.

WhiteboardServer: The WhiteboardServer is responsible for instantiating new Users with each new connection, and creating new MasterBoard objects in response to requests from Users. It maintains an InputStream to receive requests from the Users, but does not need an OutputStream.

## III. Thread Safety

### I. Processes

#### I.1 Adding New Users

#### I.2 Adding and Removing Boards

#### I.3 Drawing Strokes

#### I.4 Selecting Boards

### II. Averted Race Conditions

#### II.1 New Board and New User

#### II.2 Concurrent Strokes

#### II.3 Serve-Client Races

#### II.4 Atomic ID Generation

## IV. Testing

Overview Text

### I. SS1

Section 1

### II. SS2

Section 2

#### II.1 SSS 2-1

Subtopic of Section 2

#### II.2 SSS 2-2

Another Subtopic of Section 2

### III. SS3

Section 3