6.005 Project 2: Collaboard

Original Design Document

Date: December 3, 2013

Team Members: Kate Yu, Cathie Yun, Eric Ruleman

Assigned TA: Brian Rosario

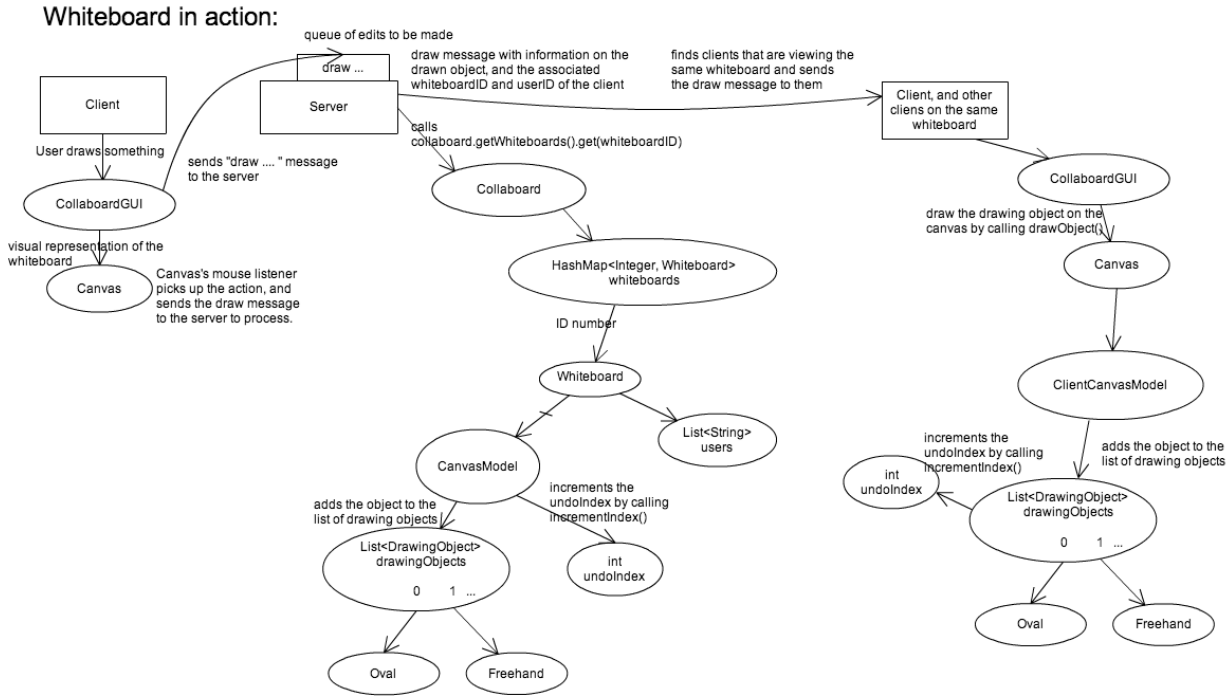
Product: Collaboard, a collaborative whiteboard program that supports freehand drawing and undoing/redoing which multiple users can edit concurrently.

Table of Contents

Q&A

- What set of editing actions are provided?
    - Freehand drawing, inserting shapes in line thickness and color of user's choosing
    - Undoing and redoing. There is a shared undo/redo queue for all users on the canvas (they can undo/redo each other's changes)
- How is the whiteboard structured?
    - From the user's point of view, each whiteboard has a toolbar and a canvas.
- How are whiteboards named and accessed by users?
    - Each whiteboard has a unique numerical ID assigned by the user during the creation of the whiteboard.
    - Whiteboards are accessed from the whiteboard selection view.
- How are whiteboards stored or cached? (e.g. at a central server or on clients)
    - It is stored on both the central server and on the clients.
    - The state of each whiteboard is stored on the clients in the List<DrawingObject> drawingObjects in the CanvasModel object. The list stores the editing actions in the order in which they occurred.
    - The state of the whiteboard is also stored on the central server so that we can pass the state of the whiteboard to a user who joins a whiteboard already in progress.
- What guarantees are made about the effects of concurrent edits?
    - All changes are processed in order (they are put on a requests BlockingQueue in the server).

Snapshot Diagram:



Datatype Design:

CollaboardServer: handles connections + Collaboard

- Testing: connect to the server, disconnect from the server, have multiple people connect to the server
- Stores the set of all Whiteboards and their current states, as well as current active threads
- has a BlockingQueue for requests to be handled (and one dedicated Thread to handle requests)

Collaboard: Whiteboard*

- Testing: hold several whiteboards, enter a whiteboard (user can only be in one whiteboard at a time), create a whiteboard (ensure that you cannot create a whiteboard with a numerical ID already taken), input a username (ensure that duplicate usernames or usernames with unsupported characters are rejected)
- Stores a list of already taken-usernames
- Stores a hashmap of UserIDs to Users
- Stores a hashmap of WhiteboardIDs to whiteboards

Whiteboard: CanvasModel + WhiteboardID + UsersList(list of usernames currently editing the canvas)

- Testing: no concurrency issues with multiple edits on one canvas, “Undo” undoes the last stroke, “Redo” redoes the next stroke (not just a line segment, and not out of order), a username does not appear in the usersList of two different Whiteboards in Collaboard (users are not allowed to be in more than one whiteboard), the state of the whiteboard is

shown correctly when a user joins a whiteboard already in progress

DrawingObject interface: implemented by Freehand, Oval so far

- Testing: Freehand stores a list of interconnected line segments as one object. Can undo and redo an entire freehand. Can undo and redo an oval.

Freehand (implemented DrawingObject): Contains a list of Line objects.
Line: Contains a startX, startY, endX, endY, color, and thickness.
Oval (implemented DrawingObject): Contains a shapeStartX, shapeStartY, x, y, color, and thickness

- Testing: (GUI testing) oval is correctly drawn on the screen even if you began (clicked) the oval at the lower right hand portion of the oval
- Contains methods to calculate the correct width, height, topLeftX, and topLeftY used to draw an oval correctly on the GUI.

ToolbarModel: Stores the Color and Thickness settings of the user associated with it.
Each user has one ToolbarModel.

Client: Class representing a connection to the server. Handles inputs from the server.
ClientCanvasModel: each client individually stores a model of the state of the canvas they are currently on.

Integrated Tests:

- The state of each whiteboard is preserved even if a Whiteboard has no current active users.
- Create a valid username, join a whiteboard, and collaborate.
- A user joining a whiteboard in the middle of a session should see the canvas in its correct state.
- If a user is on the Whiteboard Selection view, they should the whiteboard recently created by another user
- If after a series of undo operations, the user draws again, all drawingObjects past that action cannot be redone.

Rep Invariant:

- A user can only be on one whiteboard at a time.
- The list of users corresponds to users currently on the whiteboard.
- Usernames are unique (no duplicates allowed).
- Whiteboard IDs are unique (no duplicates allowed).
- The undoIndex >= 0.
- Strokes are preserved unless drawn/undone.

Client-Server Protocol:

A client will send a message to the server. If the message to the server pertains to the state of a

whiteboard, the server will then rebroadcast this message to all users currently editing the same canvas in order for the client GUIs to repaint themselves. If the message to the server pertains to a user making a board, the server will create a board, and then broadcast the information to all users. If the message to the server pertains to a user joining or leaving a board, the server will update the board, then broadcast the information to all users on that board.

Integrated Tests:

- pass CollabordServer a message pertaining to state of the whiteboard, and expect it to output that message to all the other threads in the same whiteboard. (undo, redo, draw freehand, draw oval.) (Create multiple threads connected to the same whiteboard.)
- pass CollaboardServer a message that a user is making a board, and expect it to output that message to all threads.
- pass CollaboardServer a message that a user is joining or leaving a board, and expect it to output that message to all threads in the same whiteboard.

Protocol for messages from the client to the server:

* USERID: [0-9]+
* VALIDUSER: validuser
* VALIDWHITEBOARD: validwhiteboard
* READY: ready
* COLOR: bl|y|r|g|o|m|blk|w
* THICKNESS: s|m|l
* INITDRAW: initdraw ((freehand ([0-9]+ [0-9]+)([0-9]+ [0-9]+)+)|
(oval [0-9]+ [0-9]+ [0-9]+ [0-9]+) COLOR THICKNESS
* INITDONE: initdone
* DRAWFREEHAND: draw freehand ([0-9]+ [0-9]+)([0-9]+ [0-9]+)+ COLOR THICKNESS USERID WHITEBOARDID
* DRAWOVAL: draw oval [0-9]+ [0-9]+ [0-9]+ [0-9]+ COLOR THICKNESS USERID WHITEBOARDID
* UNDOINDEX: undoindex [0-9]+
* USERTAKEN: usertaken
* WHITEBOARDTAKEN: whiteboardtaken
* ENTER: enter USERNAME
* EXIT: exit USERNAME
* UNDO: undo
* REDO: redo

Protocol for messages from the server to the client:

* WHITEBOARDID: [1-9][0-9]*
* USERID: [0]|[1-9][0-9]*
* USERNAME: [0-9a-zA-Z]+
* MAKEUSER: makeuser USERNAME WHITEBOARDID
* MAKEBOARD: makeboard WHITEBOARDID

* UNDO: undo USERID WHITEBOARDID
* REDO: redo USERID WHITEBOARDID
* COLOR: bl|y|r|g|o|m|blk|w
* THICKNESS: s|m|l
* DRAWFREEHAND: draw freehand ([0-9]+ [0-9]+)([0-9]+ [0-9]+)+ COLOR THICKNESS USERID WHITEBOARDID
* DRAWOVAL: draw oval [0-9]+ [0-9]+ [0-9]+ [0-9]+ COLOR THICKNESS USERID WHITEBOARDID
* ENTER: enter USERNAME WHITEBOARDID
* EXIT: exit USERNAME
* BYE: bye

Concurrency Strategy:
We will have a shared message blocking queue for the server. All edits will be stored in the queue to be handled by a dedicated edits-handling thread in order. Therefore “clashing” edits cannot occur. “draw”, “undo”, and “redo” messages are put in this queue, since they have the potential to clash. All other messages are handled in each client’s UserThread.

CanvasModel, Whiteboard, and Collaboard, whose information is stored on the server, are threadsafe through use of the monitor pattern, to ensure that multiple threads won’t mutate them concurrently, and cause edits to be lost, and to ensure that they won’t view any of these objects while they are being mutated. Accesses to these objects occur on the server when the user first connects (the server will access collaboard’s list of users and active whiteboards and send the appropriate messages to the client), and when the user connects to a specific whiteboard (the server will send the user a list of the Whiteboard’s list of active users, its CanvasModel’s undoIndex, and its list of DrawingObjects). Since these methods are all observing the state of these objects and not mutating them, it is sufficient to just use locking to achieve thread safety.

On the client side, all edits to the GUI are made in the event handling thread through the use of SwingUtilities.invokeLater(). All client threads have their own instance of ClientCanvasModel, which ensures thread safety through confinement.

Testing Strategy:
See notes for each design subcomponent.

GUI testing:
- State #1: Only 1 client running: Run the client. Enter “testing” for username, press Go. Expect to be taken to a landing page with no existing whiteboards and the option to create a new whiteboard.
- From State #1: try to create whiteboard by entering -1. Expect an error message.
- From State #1: create whiteboard by entering 15. Expect to be taken to a whiteboard

labelled “Collaboard 15”. Draw on the board. This is now Board #1.

- While running Board #1: Run another client. Enter “testing2” for username, press Go. Expect to be taken to a landing page with one existing whiteboard (#15) and the option to create a new whiteboard. Attempt to create whiteboard 15 (duplicate). Expect an error message. Then, select and enter whiteboard #15. This is now Board #2.
- Board #2: Expect to see the drawing from “testing” on the board. Draw on the board. Expect to see your drawings appear on the board. Expect to see “testing” and “testing2” as users connected to the board.
- Board #1: Expect to see the drawings from “testing2” on the board. Hit “undo”. Expect to see the most recent stroke from “testing2” disappear.
- Board #2: Expect the most recent stroke to disappear. Hit “redo”. Expect to see most recent stroke reappear. Press the Red color option. Draw a line. Expect to see a red line.
- Board #1: Expect to see a red line. Expect the color option on the toolbar to remain at black (should not have changed.)
- While running Boards #1 and #2: Run another client. Enter “testing3” for username, press Go. Expect to be taken to a landing page with one existing whiteboard (#15) and the option to create a new whiteboard. Enter Whiteboard #9. This is now Board #2.
- Board #3: Expect to see a blank whiteboard with default black color selected, and only “testing3” as a user connected to the board. Select Oval, Blue, Thick from the toolbar, draw an oval. Expect to see a blue, thick oval on the screen.
- Board #1 and Board #2: Expect no changes as a result of the actions of Board #3.
- Board #3: Hit Undo. Expect the screen to be empty.
- Board #2: close the window (exit entirely, stops connection.)
- Board #1: expect to see the users connected list decrease to only “testing.”