Honours Year Project Report

WebCollab
A Browser-based
Collaborative Environment

By

Tran Khoa Nguyen

Department of Computer Engineering

School of Computing

National University of Singapore

2007/2008

Honours Year Project Report

WebCollab
A Browser-based
Collaborative Environment

By

Tran Khoa Nguyen

Department of Computer Engineering

School of Computing

National University of Singapore

2007/2008

Project No: H002620

Advisor: A/P Martin Henz

Deliverables:

Report: 1 Volume

Program: 1 CD-ROM

Abstract

WebCollab is a browser-based application that facilitates a collaborative environment for different purposes. Currently, it provides 3 basic collaborative applications i.e. Whiteboard, Messenger and WebCam. However, there is a wide range of other tools that can be easily plugged in the later development stage.

WebCollab leverages on the latest technology of Web 2.0 which includes cutting-edge solutions such as AJAX, Comet and Flash CS. It's a light-weight, portable browser-based application that frees the user from the hassle of software installation, yet, is still able to deliver a comprehensive collaborative toolset.

Subject Descriptors:

D.2.13 Reusable Software	*Reusable libraries*
H.3.5 Online Information Services	*Web-based services*
	Data sharing
H.5.2 User Interfaces	*User-centered design*
H.5.3 Group and Organization Interfaces	*Web-based interaction*
	Computer-supported cooperative work

Keywords:

Whiteboard, Messenger, WebCam, Collaboration.

Implementation Software and Hardware:

Javascript, HTML,Flash

Acknowledgement

I would like to acknowledge and extend my heartfelt gratitude to A/P Martin Henz for his constant support and guidance as the project supervisor.

Besides that, I would like to thanks my teammates : Daryl Seah, Lek Hsiang Hui, Michael Lin, Siva Subramanian, Tang Tung Leh, Wang Shangshang for their assistance and feedbacks.

List of figures

Table of Contents

1. Introduction

Inspired and powered by the evolution of Web 2.0, in this project, we would like to provide a solution for the growing demand of a light-weight and portable all-in-one collaborative application. Currently, the market for collaborative software is still largely dominated by the traditional OS-based applications which require installation before using. There are a number of attempts to transform these OS-based software into browser-based ones by using Javascript and Flash. However, all of them suffer a significant setback due to the access restriction of these web programming languages as compared to other OS-based languages such as C#, Java. Besides that, the large discrepancies in API between different browsers make it hard to produce an one-fits-all solutions to use across browsers. Hence, a lot functions are absent in the browser-based version in comparison to their OS-based counterpart. For examples, local file access is not supported across all browsers; WebCam function is not accessible through Javascript. These difficulties severely limit the functionalities of the collaborative application.

At the present, there is no free all-in-one browser-based package that provides a complete collaborative toolset like the OS-based one. As a result, if the users want to work collaboratively on a virtual Whiteboard, they have to use a separated application. If the users want to conduct a video conference, they need to use another application provider. It means they need to have several accounts for each purpose. Moreover, some of the applications are not free. It is very troublesome and costly without a free all-in-one application.

Aiming to resolve this problem, we have carried out an extensive research on the current available web technology e.g. AJAX, Flash, J2EE, Java Swing to study their capability and limitation. After many discussions and trials, we have decided to use Javascript and Flash as the backbone of our development platform, and several GUI library and Server script as supporting tools.

The end product of our project is a full-fledged browser-based application with 3 basic facilities i.e. Whiteboard, Messenger and WebCam. Other facilities will be developed at later stages and will be easily plugged in the system.

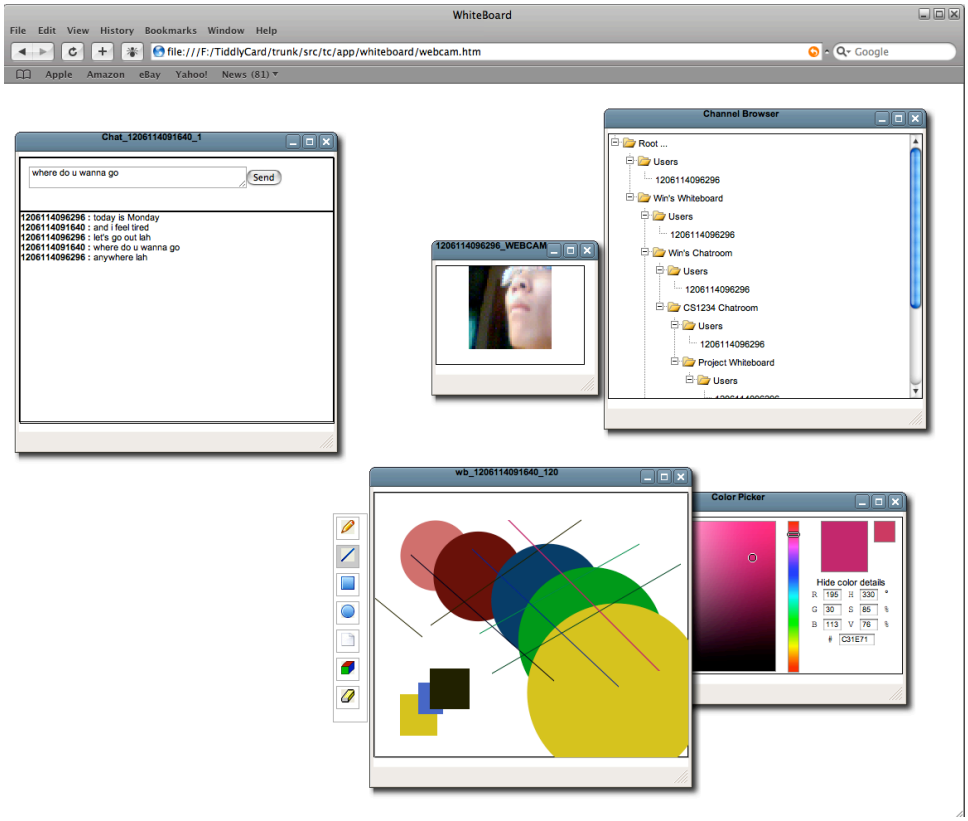

Figure 1 The GUI of WebCollab

System Requirement:

Browser:

- **Firefox 1.5** and above
- **Internet Explorer 6.0** and above
- **Safari 2.0** and above

Flash Player 8 and above

2. Background Information

2.1. TiddlyWiki

TiddlyWiki [1]is a modeled client-side single-page application written by Jeremy Ruston that is designed to be used as a personal notebook. It is a single self-contained HTML file that includes CSS and JavaScript code. When the user downloads it to their PC, TiddlyWiki can save the entered information by overwriting itself on the user's disk, at the user's request.

Unlike traditional browser-based applications which require uninterrupted server connectivity, TiddlyWiki applications are client-centred and remain functional with or without connectivity, enabling users to use the applications anytime without disruption. When a network connection is available, TiddlyWiki applications may make use of the server capabilities to perform tasks such as synchronisation of content repository (automatically or at user's discretion) and collaboration.

One of the most revolutionary feature that TiddlyWiki offer is the capability to develop and install TiddlyWiki-specific application (known as 'plugins') into the TiddlyWiki's core. In this sense, TiddlyWiki's core can function as a tiny operating system.

Built entirely on Javascript and a browser-based framework, TiddlyWiki is platform independent, enabling users to access TiddlyWiki applications through desktop computers, laptops and any browser-enabled devices including Personal Digital Assistants (PDAs) and mobile phones regardless of the underlying operating system.

Following TiddlyWiki conventions, users can make a new entry, called a tidier, in their local copy of the TiddlyWiki file and save it for future reference. Existing tiddlers can also be modified or deleted in the same way. Because it runs under most browsers and requires no installation, it can be easily used as a portable personal wiki.

---

[1] http://www.tiddlywiki.com/

TiddlyWiki is published by Osmosoft under a BSD open source license, which makes it freely available.

A TiddlyWiki is like a blog because it's divided up into neat little chunks (tiddlers), but it encourages you to read it by hyperlinking rather than sequentially: if you like, a non-linear blog analogue that binds the individual microcontent items into a cohesive whole. TiddlyWiki represents a novel medium for writing, and promotes its own distinctive Writing Style.

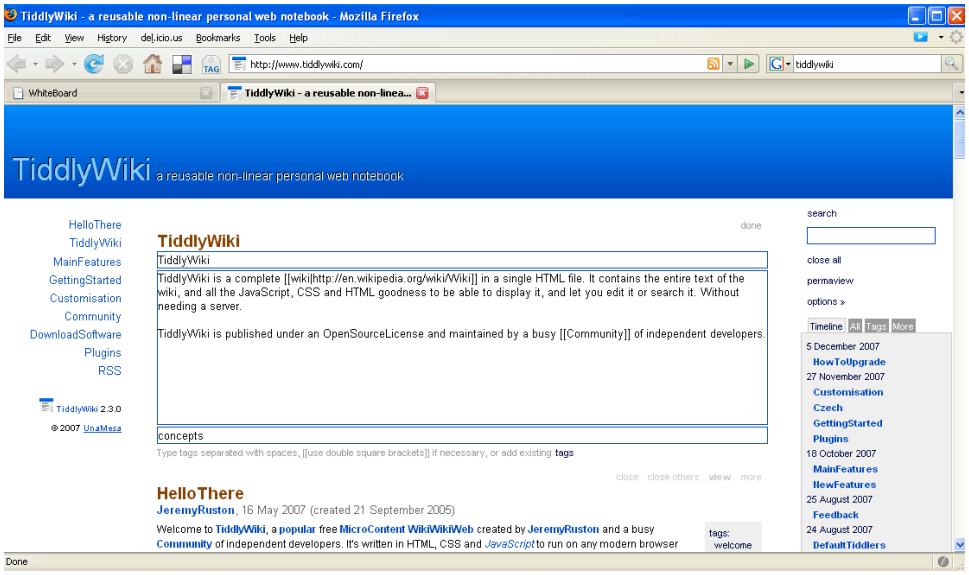

Figure 2 TiddlyWiki

2.2. TiddlyCard

Inspired by the evolution of TiddlyWiki, a group of developers from the National University of Singapore (NUS), lead by Martin Henz, a professor at NUS, have designed a new browser-based platform called TiddlyCard that enables the developers to create and deploy applications that are client-centric, server supported and platform independent.

TiddlyCard has a wide variety of client and server-sided components include:

1. **Asalta** : a WYSIWYG designing tool by **Siva Subramanian**
2. **Asalta for Kids** : a special version of Asalta designed for kids by **Jaysen Pang**
3. **WebCollab** : a collection of collaborative tools by **Tran Khoa Nguyen**
4. **Debugger** : a debugging tool for TiddlyCard by **Zhou Lin**

5. **Deployment System** : a deployment framework for TiddlyCard by **Dilip Rajendran & Chua Rui Wen**
6. **Email in TiddlyCard** : an email plugin for TiddlyCard by **Alvin Ng & Chen Xi**
7. Form Generation and Handling : a tool for automatic form generation by **Tang Tung Leh**
8. **Foundations for a Browser-based Application platform** : a framework to support TiddlyCard plugins by **Daryl Seah**
9. **Multimedia Resource Manager** : a multimedia support for TiddlyCard by **Tran Khoa Nguyen**
10. **Plugin System** : a framework to support TiddlyCard plugins by **Daryl Seah**
11. **TiddlyCalendar** : a collaborative event scheduling calendar tool by **Michael Lin**
12. **TiddlyRSS** : a RSS reader by **Wang Shangshang**
13. **Version Control & Server-Side API** : an API for Version Control & Server-Side communication by **Lek Hsiang Hui**

TiddlyCard is comparable to a browser-based Operating System (OS) where there is a booting process, and is bundled with a Unix-like console, as well as a wide range of applications such as email, calendar, and Messenger.

3. Components

The current version of **WebCollab** consists of 4 key components

3.1. Channel Manager

A channel in **WebCollab** refers to a communication channel in which all the transmitted data can be shared among all the users that joined. There are 2 kinds of channel:

a. **Private channel:** is meant for private communication between only 2 users

b. **Public channel:** is meant for public communication among 2 or more users

Private channel is implicitly created when one user triggers a private communication channel such as Messenger with another user. Private channel is not displayed in the **Channel Manager** because private channel is not meant to be shared among other users.

Public channel is explicitly created by the user. Public channel can be assimilated to a public Chat room in Instant Messenger (IM) application such as Yahoo Instant Messenger [2]. However, public channel in **WebCollab** can represent not only a Messenger application but also other applications such as WebCam or Whiteboard. In addition, a channel can contain many sub channels e.g. a Whiteboard can have several sub-Whiteboard and sub-Messenger, and so on.

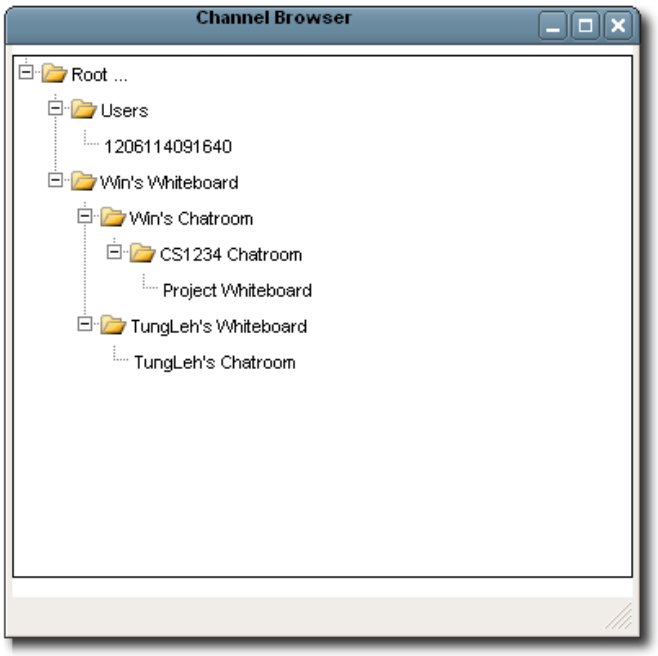

Figure 3 Channel Manager

There is a root channel called *Root* and all subsequently created channels are its sub-channels. A channel can be created by right click on the parent channel and select the type of channel e.g. Messenger or Whiteboard.

[2] http://Messenger.yahoo.com/

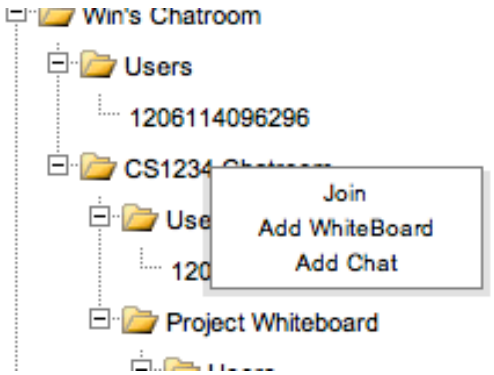

Figure 4 Creation of new Channel

3.2. Messenger Application

Messenger application in **WebCollab** is similar to standard IM offered by **American Instant Messenger (AIM)** [3]or **Microsoft Messenger (MSN)**[4], however, at a smaller scale. There are 2 types of Messenger in **WebCollab**:

- Private Messenger : is a private Messenger between 2 users
- Public Messenger : is a public Messenger for 2 or more users

[3] http://www.aim.com

[4] http://Messenger.msn.com

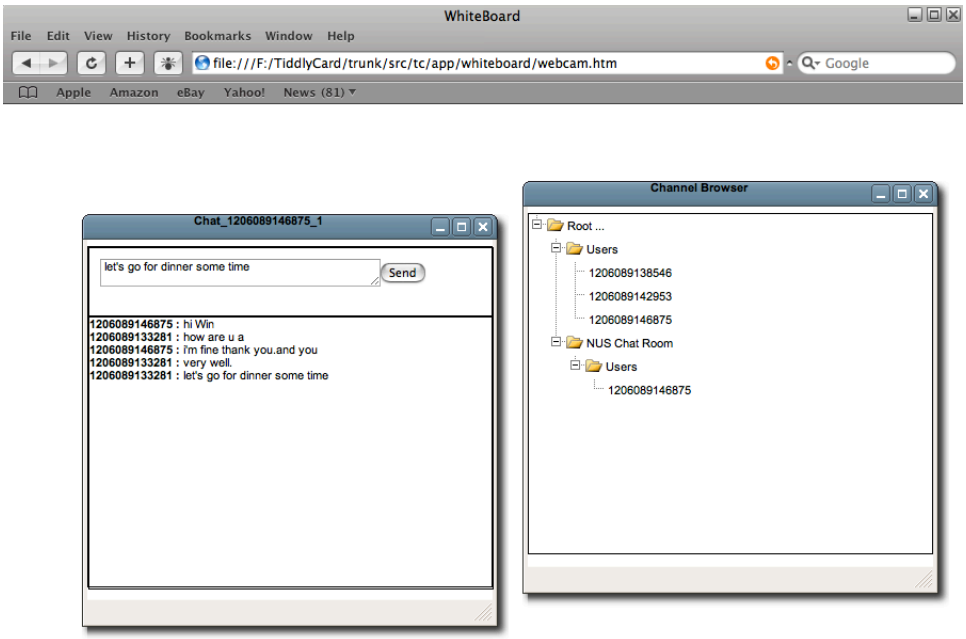

Figure 5 Messenger Application

Private Messenger application can be triggered by right click on the targeted user and select **Start Messenger.** A Messenger window will pop up at both parties and they can start a 2-way text conversation.

Public Messenger application can be added by right click on the parent channel, select **Add Messenger** and enter the name of the new public Messenger. This new public Messenger application will appear in the **Channel Manager** of all users.

Figure 6 Popup window for New Messenger's name

Any users who want to participate in the public Messenger channel can right click on the channel and select **Join**.

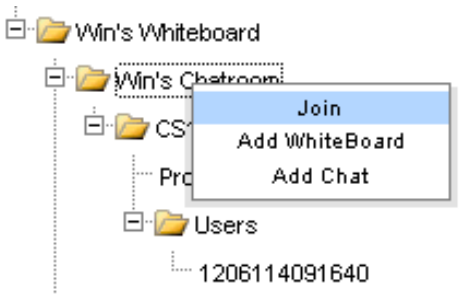

Figure 7 Join a Messenger Application

The list of all the users who joined the Messenger application is displayed in the **Users** list.

3.3. Whiteboard Application

Whiteboard is a drawing application that lets users share their drawing in real-time through the Internet. Similarly to Messenger Application, there are 2 types of Whiteboard Application i.e. private and public one. The procedures to create and join it are also the same.

The latest version of **WebCollab** supports 6 different drawing functionalities:

1. **Pencil**  : draws a path

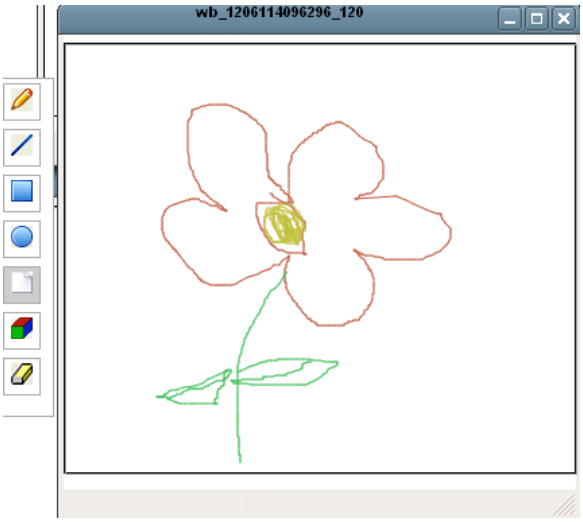

Figure 8 A sketch drawn by Pencil tool

2. **Line**  : draws a line

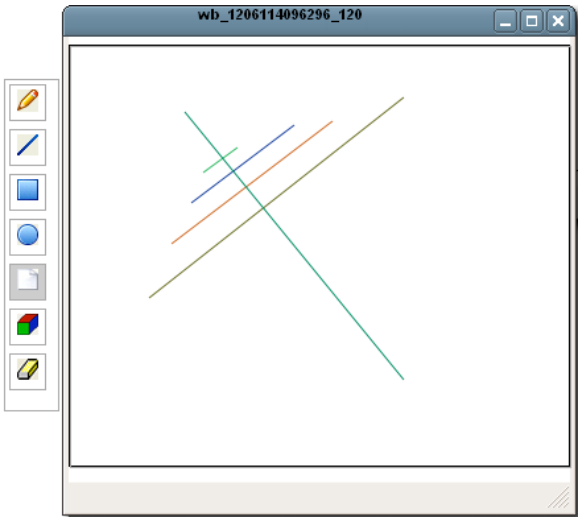

Figure 9 A sketch drawn by Line tool

3. **Rectangle**  : draws a rectangle

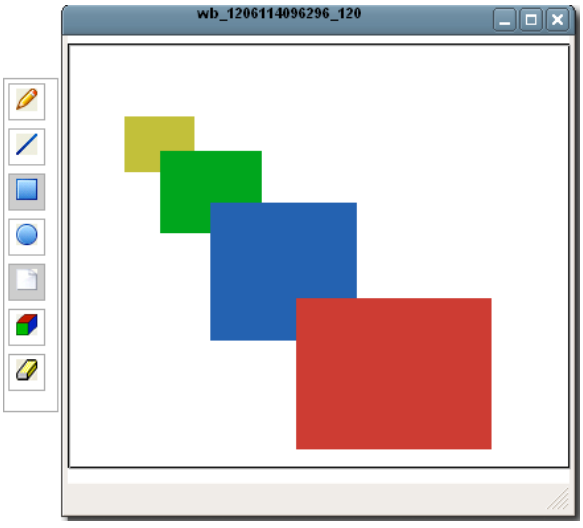

Figure 10 A sketch drawn by Rectangle tool

4. **Circle**  : draws a circle

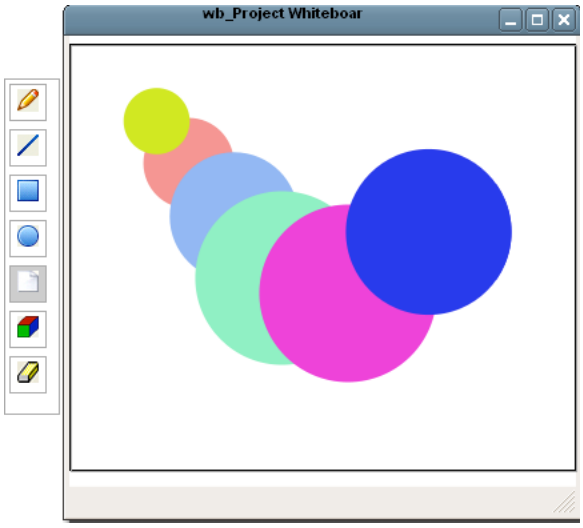

Figure 11A sketch drawn by Circle tool

5. **Clear**  : clear the Whiteboard

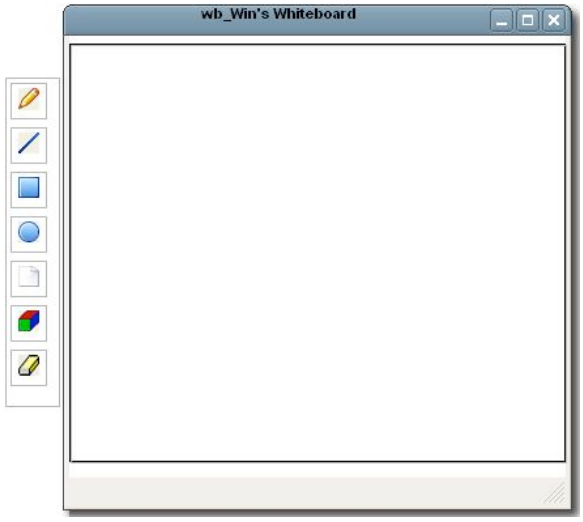

Figure 12 A cleared Whiteboard

6. **Color picker**  : choose the color

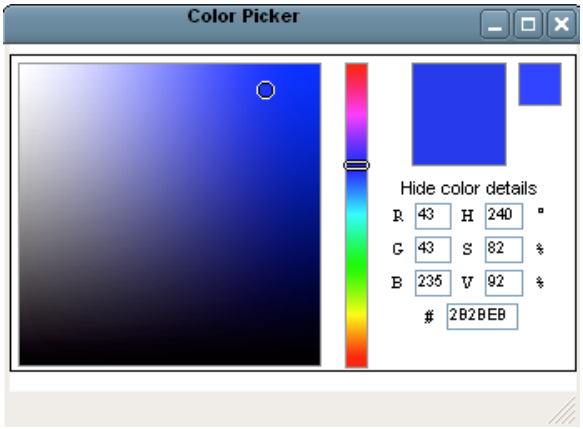

Figure 13 Color Picker

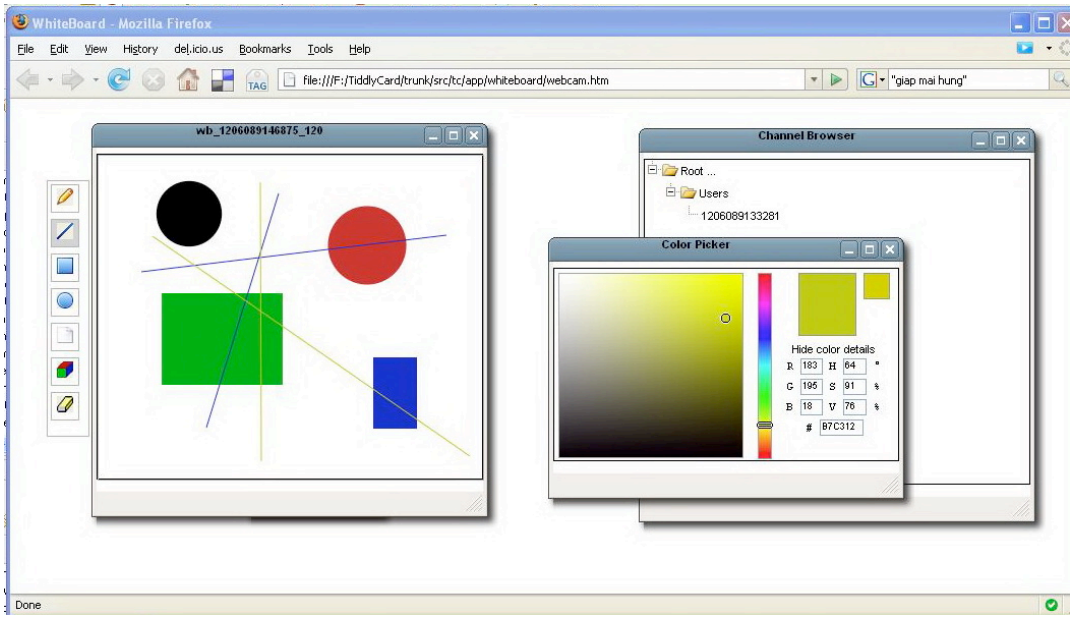

Figure 14 Sample sketch of Whiteboard Application

3.4. WebCam Application

Currently, the WebCam application is only available in private mode i.e. only between 2 users. It means only maximum 2 users can participate in a WebCam application. There is no channel for multiple users to share their WebCam. It's because the WebCam application consumes a very

large bandwidth, which exceeds our server bandwidth. Hence, we have to restrict to only 2 users in the current platform.

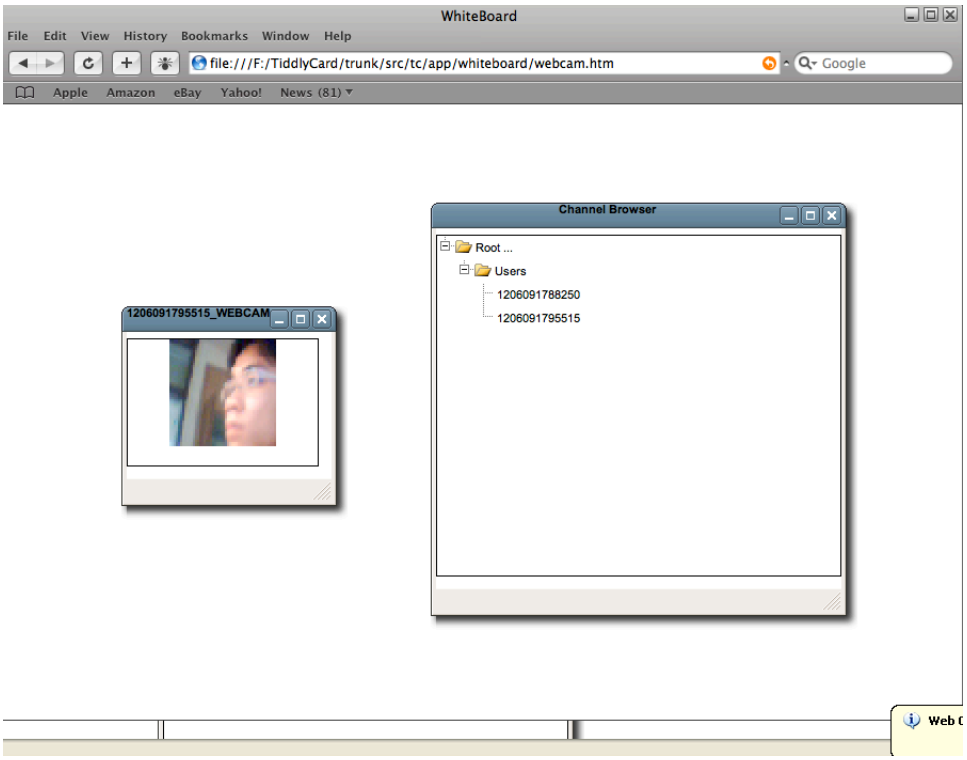

Figure 15 A WebCam application

4. Design

4.1. Window-style GUI[5]

All the components in **WebCollab** are contained in a Window-style <*DIV*> element which can be neatly minimized, maximized, dragged and dropped. The Window-style GUI gives the users an illusion that they are using a Window-style OS which they are more familiar and comfortable with.

The Window-style GUI is provided by a Javascript class called *Window*. The constructor of the *Window* class takes in 2 parameters:

[5] http://en.wikipedia.org/wiki/Window_%28computing%29

- *id (String)* : Unique DIV ID of the created window
- *config (JSON)* : Configuration for the created window. It contains:
  - *xpos, ypos* : initial (x,y) coordinates
  - *w,h* : initial width & height
  - *title* : title bar
  - *body* : <DIV> ID of the body

The *Window* class is developed from a prototype Window-style GUI called **CapOS**[6]. The following additional features have been added to the existing prototype:

- Define the Windows GUI in a class so that it can be initialized more easily
- Add the configuration parameters to make it more flexible
- Add a z-index manager mechanism

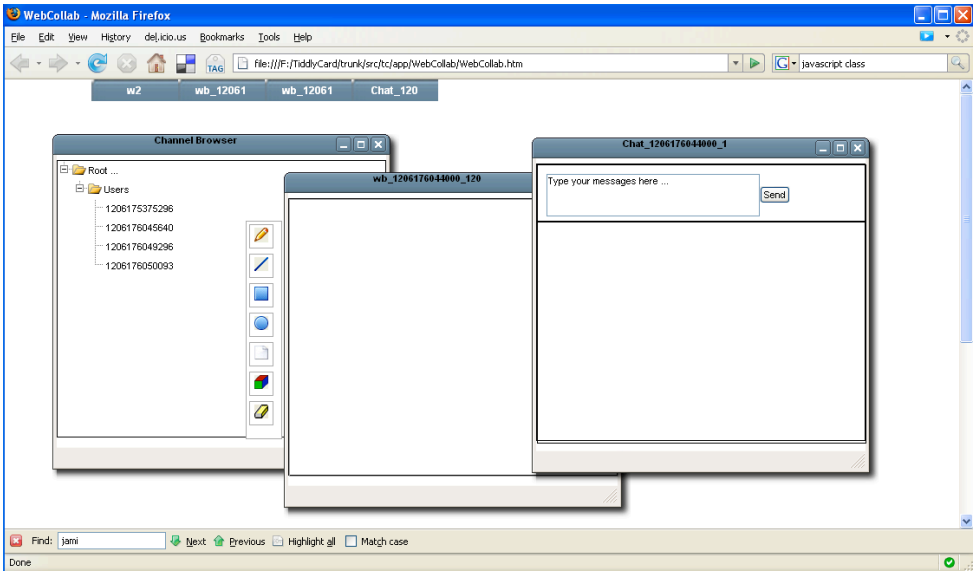

Figure 16 Multiple instances of *Window* class

[6] http://www.captain.at/howto-ajax-browser-operating-system.php

4.2. Hierarchical structure

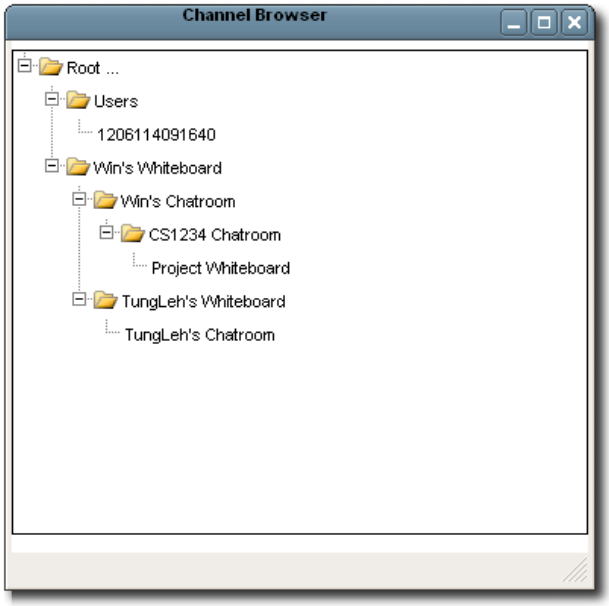

Figure 17 Hierarchical structure of the *Channel*

All the channels are organized in a hierarchical tree structure which is provided by *Tree API* in **Yahoo! User Interface Library (YUI)**. Every channel is represented by a node in a tree and can be manipulated quickly through the API. The hierarchical structure makes it easier to access and organize all the channels.

5. Programming Languages and API

5.1. Javascript

5.1.1. What is Javascript?

JavaScript [7] is a scripting language most often used for client-side web development. It was the originating dialect of the ECMAScript standard. It is a dynamic, weakly typed, prototype-based language with first-class functions. JavaScript was influenced by many languages and was designed to have a similar look to Java, but be easier for non-programmers to work with. The language is best known for its use in websites (as client-side JavaScript), but is also used to

[7] http://en.wikipedia.org/wiki/JavaScript

enable scripting access to objects embedded in other applications (for example Microsoft Gadgets in Windows Vista Sidebar).

5.1.2. Why Javascript?

- **Platform independence**

  Javascript is supported by most of the modern browsers e.g. Internet Explorer, Firefox, Opera, without the need to install any additional plug-in. Hence, **WebCollab** can be run in any operating system including mobile devices that supports a modern browser.

- **Lightweight**

  Javascript leaves a relatively small footprint as compared to other programming languages. This minimizes the resource occupied by **WebCollab** and enables it to be run in system with low memory and processing power.

- **Familiarity**

  There is a great similarity in the syntax and rules between Javascript and Java, a language that all our developers are familiar with. Hence, the development process can be greatly sped up.

5.2. Javascript Object Notation (JSON)

5.2.1. What is JSON

JSON[8], short for **JavaScript Object Notation**, is a lightweight computer data interchange format. It is a text-based, human-readable format for representing simple data structures and associative arrays (called objects).

The JSON format is often used for transmitting structured data over a network connection in a process called serialization. Its main application is in Ajax web application programming, where it serves as an alternative to the traditional use of the XML format.

---

[8] http://www.json.org

5.2.2. Why JSON?

- **Simplicity**

  As compared to XML, it is much simpler to package data in JSON and push it across the network from the sending party. At the receiving party, it is easier to extract data from JSON object since there is no complex parsing needed. Most of the data structures transmitted in **WebCollab** are very simple; hence, it is very convenient to hardness the simplicity of JSON. Moreover, the adopted network protocol i.e. Comet, only supports transfer of data in JSON format.

5.3. HTML 5

5.3.1. What is HTML 5?

HTML 5 [9]provides a number of new elements and attributes ("tags") that reflect typical usage on modern web sites. Some of them are technically similar to `<div>` and `<span>` tags, but have a meaning, for example `<nav>` (website navigation block) and `<footer>`. Such tags would facilitate indexing by search engines and handling by small-screen devices or voice readers for the visually impaired. Other elements provide new functionality through a standardized interface, such as the `<audio>` and `<video>` elements.

In addition to specifying markup, **HTML 5** specifies scripting application programming interfaces (APIs).Existing Document Object Model (DOM)[10] interfaces are extended and de facto features documented. There are also new APIs, such as:

- Immediate-mode 2D drawing
- Storage
- Offline
- Editing
- Drag &Drop
- Messaging/Networking

---

[9] http://html5.org/

[10] http://www.w3.org/DOM/

5.3.2. Why HTML 5?

- **Functionalities**

  New API Library in **HTML 5** facilitates the development of a full-fledged graphic and multimedia-capable browser-based web application. For examples, the Window-style design in **WebCollab** requires extensive use of Drag & Drop and DOM manipulation feature; the offline and storage functionalities are made use in TiddlyCard's core. In particular, the new <*CANVAS*> HTML element which allows dynamic scriptable rendering of bitmap images is the foundation of the Whiteboard application. Canvas consists of a drawable region defined in HTML code with *height* and *width* attributes. JavaScript code may access the area through a full set of drawing functions similar to other common 2D APIs, thus allowing for dynamically generated graphics. In most of other Whiteboard applications, Flash is used to build the drawing board. This results in extra latency and memory usage due to the loading of Flash clip and the data transfer to Javascript. Hence, by harnessing this new <*CANVAS*> API, we have significantly speeded up our application.

5.4. Dojo Comet

5.4.1. What is Comet?

Comet [11]is a World Wide Web application architecture in which a web server sends data to a client program (normally a web browser) asynchronously without any need for the client to explicitly request it. It allows creation of event-driven web applications, enabling real-time interaction which would otherwise require much more work.

---

[11] http://en.wikipedia.org/wiki/Comet_(programming)

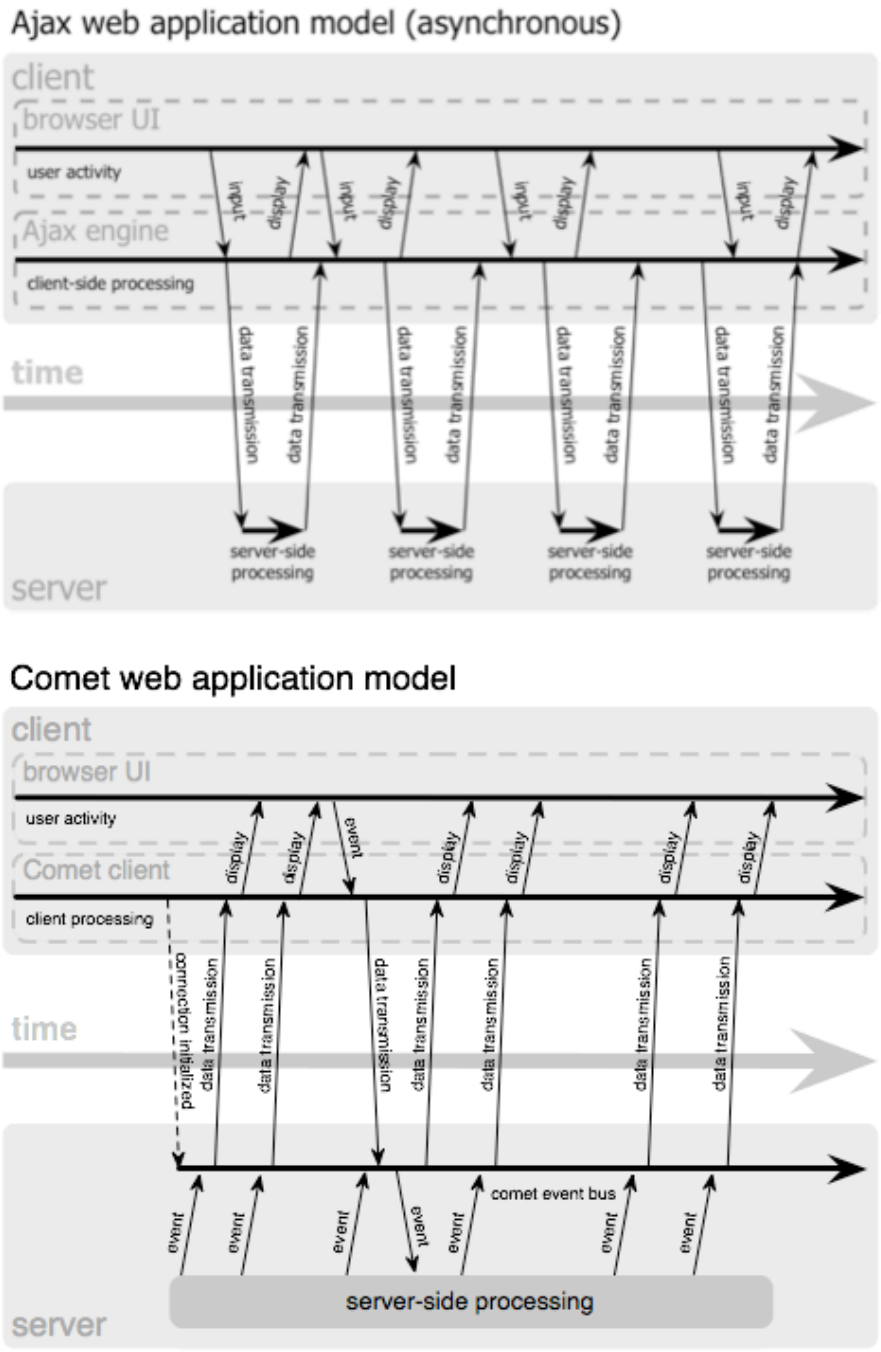

Figure 18 Comet Web Application Model

Comet applications use long-lived HTTP connections between the client and server, which the server can respond to lazily, pushing new data to the client as it becomes available. This differs from the original model of the web, in which a browser receives a complete web page in

response to each request, and also from the Ajax model, in which browsers request chunks of data, used to update the current page. The effect is similar to applications using traditional Ajax with polling to detect new information on the server, but throughput is improved and latency and server load are decreased.

5.4.2. Why Dojo Comet?

- **Speed**

  As compared to the traditional asynchronous Ajax application model which waits for client's request before sending the data, Comet protocol is faster because it sends data without explicit request from the client. Comet is particularly usefully in this scenario because our application is a real-time collaborative environment which involves frequent data exchange among clients. Each data package is quite small; hence, it would be inefficient to use the traditional Ajax communication which sends out frequent requests for updates and waits for the response. Comet reduces not only the data transmissions but also the time latency for data updates.

- **Simplicity**

  Dojo Comet offers a comprehensive Comet API for developers. The API can be extended to cater more complex communication such as Peer-To-Peer (P2P)[12]. In our application, we make use of only 3 methods in the API i.e. *init*, *subscribe* & *publish*. Besides that, there is an additional API developed by Hsiang Hui, one of the TiddlyCard's developers, for advanced usage.

5.5. Flash External Interface

5.5.1. What is Flash External Interface?

The **Flash External Interface** [13]class is the External API, an application programming interface that enables straightforward communication between ActionScript and the Flash Player container; for example, an HTML page with JavaScript, or a desktop application with Flash

---

[12] http://en.wikipedia.org/wiki/Peer-to-peer

[13] http://livedocs.adobe.com/flash/8/main/00002200.html

Player embedded. ExternalInterface requires the user's web browser to support either ActiveX[14] or the Nescape Plugin API [15]that is exposed by some browsers for plugin scripting. Most of the modern browsers support it.

ExternalInterface is similar in functionality to the *fscommand()*, *CallFrame()* and *CallLabel()* methods, but is more flexible and more generally applicable. Use of ExternalInterface is recommended for JavaScript-ActionScript communication. ActionScript can call any JavaScript function on the HTML page, passing any number of arguments of any data type, and receive a return value from the call. Vice versa, JavaScript embedded in the HTML page, can call an ActionScript function in Flash Player. The ActionScript function can return a value, and JavaScript receives it immediately as the return value of the call.

5.5.2. Why Flash External Interface?

- **Multimedia Capability**

  Due to the security restrictions, Javascript can't access any of the computer peripheral such as WebCam or Microphone. However, Flash has higher access right and it is able to access some of the Multimedia peripherals. Therefore, **Flash External Interface** acts as a communication bridge for Javascript to access these peripherals. In our WebCam application, Flash is used to capture and send WebCam data to Javascript which processes and transmits it across the network through Comet protocol.

5.6. Yahoo! User Interface Library (YUI)

5.6.1. What is Yahoo! User Interface (YUI)?

The **Yahoo! User Interface (YUI)** [16]Library is a set of utilities and controls, written in JavaScript, for building richly interactive web applications using techniques such as DOM scripting, DHTML[17] and AJAX[18]. The YUI Library also includes several core CSS [19] resources.

---

[14] http://en.wikipedia.org/wiki/ActiveX

[15] http://developer.mozilla.org/en/docs/Gecko_Plugin_API_Reference:Scripting_plugins

[16] http://developer.yahoo.com/yui/

[17] http://en.wikipedia.org/wiki/Dynamic_HTML

All components in the YUI Library have been released as open source under a BSD license and are free for all uses

5.6.2. Why YUI?

- **Functionalities and Convenience**

  YUI provides a very comprehensive API in which some are widely used in **WebCollab** such as Animation, Menu, Event, and Color Picker. YUI significantly reduces the development time and can help to streamline the GUI across different components in TiddlyCard system.

6. Implementation

6.1. Algorithms

All the data exchanges across the messages are performed through **Dojo Comet API**. The 3 most commonly methods are:

- ***cometd.init****({},serverURL)*

  Establish a connection to a Comet server @ *serverURL*. This is done one, after that all the network transmissions will be handled by this server.
- ***`cometd.subscribe`****`(channelId,handlerFunc)`*

  Subscribe to a channel with ID *channelId*. All the messages received will be handled by the function *handlerFunc*.
- ***cometd.publish*** *(channelId, msg)*

  Publish a message *msg* to channel with ID *channelId*. All the channels subscribe to *channelId* will receive this message *msg*.

6.1.1. Channel Manager

Channel Manger is responsible for the creation, synchronization of the channel list and user list among all the clients. All the information related to the channel list e.g. channel creation, deletion are published to a common pre-defined *CHANNEL_MANAGER_CHANNEL_ID* channel

---

[18] http://developer.mozilla.org/en/docs/AJAX

[19] http://www.w3.org/Style/CSS/

which is subscribed by all clients. The Channel Manager module listens to and uses the information published in this common channel to synchronize the Channel list.

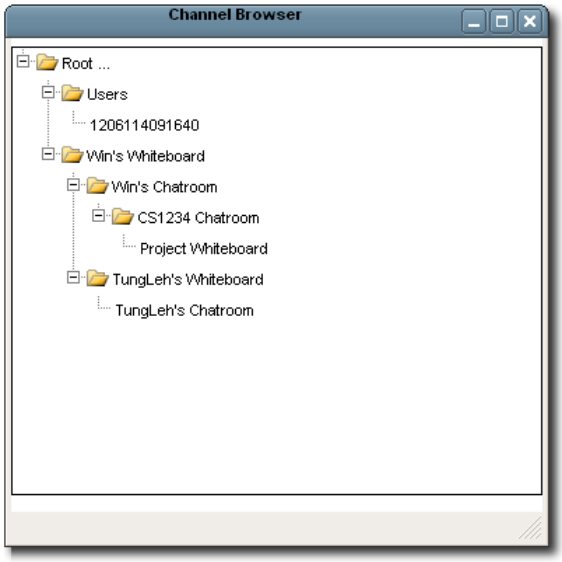

Figure 19 Hierarchical structure of the Channel Manager

When there is a change to the channel list e.g. new channel created or new user, a message will be published to *CHANNEL_MANAGER_CHANNEL_ID* channel. Typically, the message contains the following fields:

- *userId* : ID of the message's owner
- *action* : taken action e.g. new channel created, channel deleted.
- *channelId* : ID of the channel
- *channelPath* : the path of the channel

6.1.1.1. Create a new Channel

A channel in **WebCollab** is represented by a Channel class whose constructor takes in 3 parameters:

- *id* : ID of the channel
- *path* : the absolute path of the channel in the hierarchical tree structure, it is used as the unique channel ID in Comet as well
- *type* : type of the channel e.g. Whiteboard.

There are 2 ways to create a channel:

- explicitly through the creation of new public Whiteboard, Messenger or WebCam
- implicitly through the creation of new private Whiteboard, Messenger or WebCam

When a new channel is created by a client, it is automatically appended to the YUI Tree structure to the correct hierarchical position according to its path. The path of each channel is unique. A Comet "*New Channel*" message is published to *CHANNEL_MANAGER_CHANNEL_ID* channel.

6.1.1.2. Manage and Synchronize Channel list

The synchronization of the Channel list can be done in 2 ways:

- **Event-triggered**: when there is new message published to the *CHANNEL_MANAGER_CHANNEL_ID* channel, the Channel Manager will examine the message and perform corresponding updates. Examples of event-triggered messages are "*New Channel*", "*Remove Channel*".
- **Periodical**: Channel Manager broadcasts message with action: *REFRESH_CHANNEL_LIST_REQ* to request for updates from all the clients. All the clients will send a *REFRESH_CHANNEL_LIST_RES* message which contains a list of the channels they join or create. The interval of auto-refreshment can be adjusted; it is currently set at 1 sec.

6.1.1.3. Manage and Synchronize User list

A user is considered as a private channel in **WebCollab**. When a user joins or creates a Channel**,** a new private Channel which takes *userID* as *channelID* is created under the sub-channel Users of that Channel. For example, user A joins channel C (*channelId* : *C*, *type* : *WHITEBOARD*, *path* : */Root/A/B/C*) , a new Channel A (*channelId* : *A*, *type* : *USER*, *path*: */Root/A/B/C/Users/A*) is created. A "*New Channel*" message is broadcasted to the *CHANNEL_MANAGER_CHANNEL_ID* channel.

When a user exits a Channel, a "*Remove Channel*" message is broadcasted to the *CHANNEL_MANAGER_CHANNEL_ID* channel. Upon receiving this message, the client will delete the user from the channel, by referring to its *path* and *ID*.

6.1.2. Messenger Application

The **Messenger Application** is represented by a Messenger class whose constructor takes in 2 parameters:

- *id*: Id of the Messenger class
- *channelId:* unique Channel Id of the Messenger class

Each Messenger is responsible for handling all the data transmission within a unique channel through 2 main functions:

- *send* : send out a message by publish it to the channel
- *msgHandler* : receive the message and display it to the Messenger window

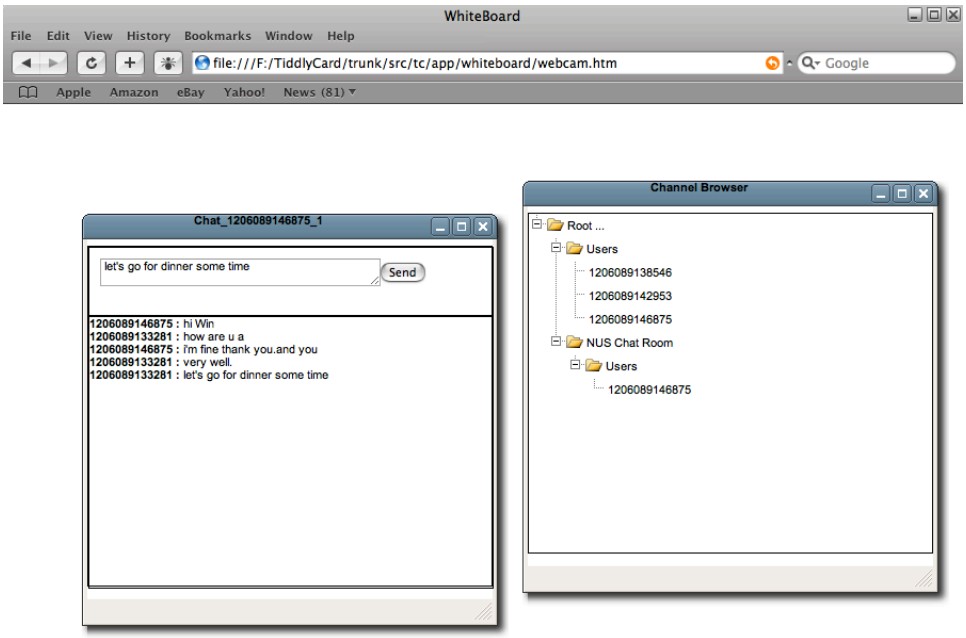

Figure 20 Messenger Application

6.1.2.1. Creation of new Messenger Application

The instantiation of a private or public Messenger application is similar. It involves in the creation of a new Messenger class. This will create a new channel and subscribe to it using the Dojo Comet API *cometd.subscribe*(*channelId,handlerFunc*) with *channelId* is the *path* of the channel in the hierarchical tree structure.

6.1.2.2. Send/Receive messages

After the client subscribes to a channel in the instantiation process, he can start sending and receiving messages. Sending message can be performed by the method *cometd.publish* (*channelId, msg*). The message *msg* contains the sender's ID & the Messenger text. At the receiving side, the *msgHandler* of the Messenger class will be event-triggered and the message will be displayed to the Messenger window. A <*DIV*> is embedded in the Messenger class window to display the text conversation.

6.1.3. Whiteboard Application

The **Whiteboard** is encapsulated in the Whiteboard class whose constructor takes in 2 parameters:

- *id*: Id of the Whiteboard class
- *channelId:* unique Channel Id of the Whiteboard class

The Whiteboard class is responsible for all the network transmission in Whiteboard application e.g. update the drawing board; send drawing status occurred in the application through the main function

- *msgHandler*: receive and process all the updates for the Whiteboard.

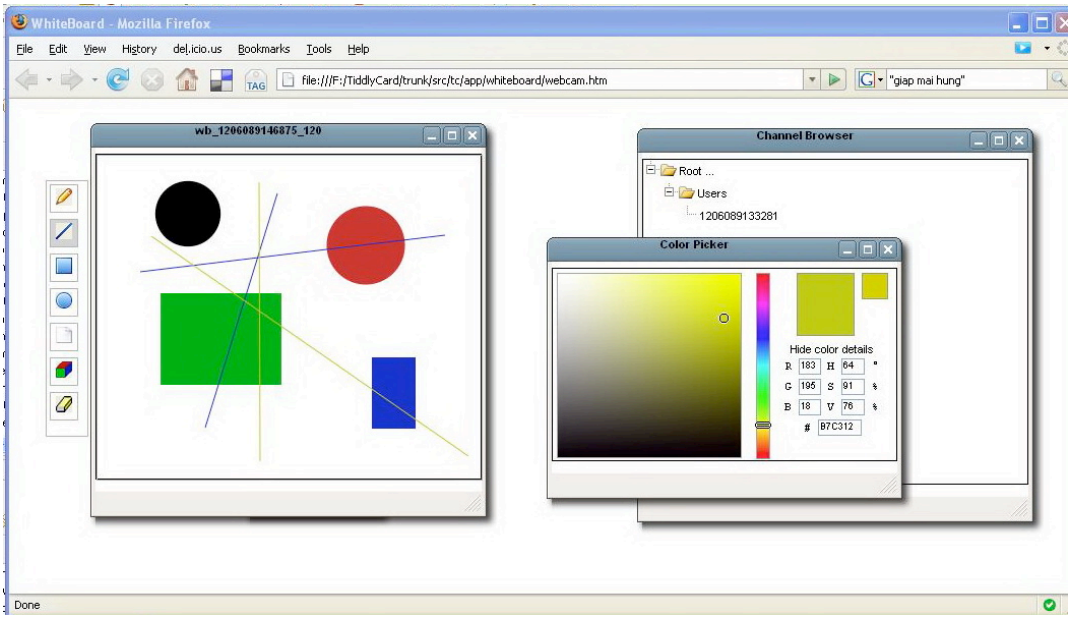

Figure 21 Whiteboard Application

6.1.3.1. Creation of new Whiteboard

The instantiation of a private or public Whiteboard is similar. It involves in the creation of a new Whiteboard class. This will create a new channel and subscribe to it using the Dojo Comet API *cometd.subscribe*(*channelId,handlerFunc*) with *channelId* is the *path* of the channel in the hierarchical tree structure. A <*CANVAS*> *element* is embedded in the Whiteboard window to display the drawing.

6.1.3.2. Draw and update the Whiteboard

The Whiteboard class extends the CanvasPainter class which provides all the drawing functionalities. All the graphics are rendered by using Javascript through a <*CANVAS*> element. The basic drawing functions include:

- *drawLine*: draws a line by calling the <*CANVAS*> *API*

which draws a line from point *x* to point *y.*
A message with action *DRAW_LINE* containing the coordinates of 2 points x & y is published to the channel so that other clients can update their canvas.

- *drawPencil*: draws a path constructed by the mouse movement. This is done by breaking the path into multiple lines and draws them individually by calling the *drawLine* function.
- *drawCircle:* draw a circle by calling the <*CANVAS*> *API* function
  void arc(in float x, in float y, in float radius, in float startAngle, in float endAngle, in boolean anticlockwise)
  A message with action *DRAW_CIRCLE* containing the coordinates of the center and the radius is published to the channel

- *drawRectangle* : draw a rectangle by calling the <*CANVAS*> *API* function
  void fillRect(in float x, in float y, in float w, in float h)
  A message with action *DRAW_RECTANGLE* containing the coordinates of the top left hand corner, the height and the width of the rectangle is published to the channel.

- *clearCanvas :* clear the entire canvas by calling the <*CANVAS*> *API* function

void clearRect(in float x, in float y, in float w, in float h);

- *setColor:* set the drawing color by setting the *<CANVAS> API* parameters *fillStyle, strokeStyle.* The color selection is performed by Yahoo! User Interface (YUI) *ColorPicker*

  A message with action *DRAW_SETCOLOR* containing the value of selected color is published to the channel.

6.1.4. WebCam Application

The **WebCam Application** is represented by a WebCam class whose constructor takes in 2 parameters:

- *id*: Id of the WebCam class
- *channelId:* unique Channel Id of the WebCam class
- *type :* type of the WebCam i.e. Receiver or Sender

Each WebCam class is responsible for handling all the WebCam data transmission within a unique channel through 2 main functions:

- *msgHandler* : receive the images and display it to the WebCam window

In addition to the WebCam class, there is a Flash object WebCam.swf which is used to capture and send the WebCam images to the Javascript WebCam class. The communication between Flash and Javascript is facilitated by the ***Flash External Interface*** class.

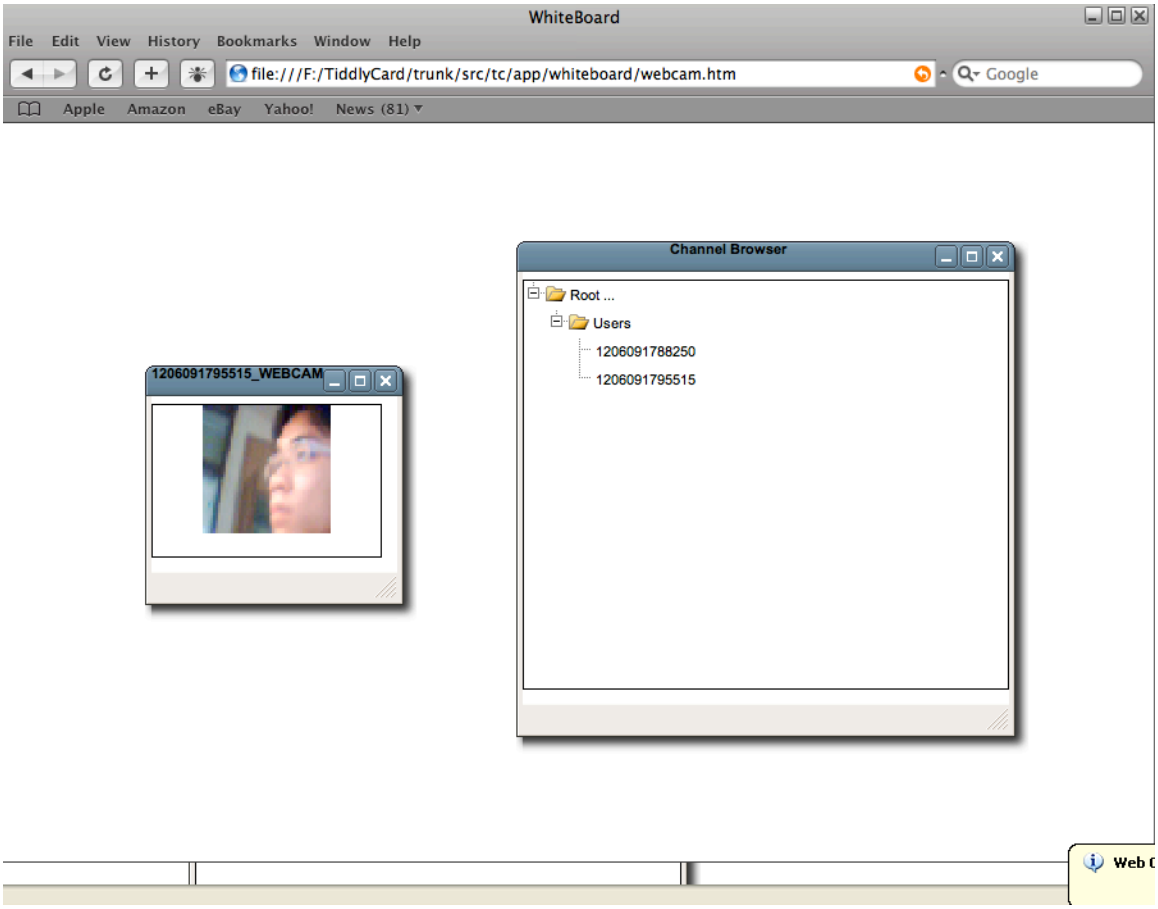

Figure 22 WebCam Application

6.1.4.1. Creation of new WebCam

Only private WebCam channel is available. It involves in the creation of a new WebCam class. This will create a new channel and subscribe to it using the Dojo Comet API *cometd.subscribe*(*channelId,handlerFunc*) with *channelId* is the *path* of the channel in the hierarchical tree structure. A Flash Object WebCam.swf is embedded in the WebCam window to display the images from the WebCam.

6.1.4.2. Capture and Send/Receive WebCam images

At the sender's side, WebCam images are captured by the WebCam Flash object by the Flash function

void Camere.get()

which captures the WebCam image and store all the pixels it in a 2-dimensional array. Each element is a RGB color of each pixel in the captured WebCam image.

The image is display in the WebCam Flash object by calling the Flash function *draw*.The 2-dimensional array is sent to Javascript using Flash External Interface API. Javascript transforms the data into JSON object and send through Comet to the receiving side. At the receiving side, Javascript extracts the 2-dimensional array and sends it to Flash by using **Flash External Interface** API. Flash converts the 2-dimensional array into bitmap object and draw it by calling the Flash function *draw*.

6.2. Design Pattern

7. Features

7.1. Portability

The size of the full **WebCollab** application is only XXX KB with the complete external libraries such as Dojo, YUI included. **WebCollab** can be neatly fit in a thumb drive or hosted on a website. It is written entirely by Javascript and Flash, hence, there is no installation needed. The only software requirements are a compatible browser and a Flash Player plug in which are included in almost all the present computers.

Therefore, the running of **WebCollab** is simply a matter of a click of a button. The user can straight away use **WebCollab** by clicking on the local copy or accessing the website that hosts it. Due to its small size, the user can store **WebCollab** in a thumb drive and bring it to anywhere. Since, there is no installation needed, he can use in public computers where he doest have a administration right to install software.

7.2. Platform independence

Since **WebCollab** is written entirely in Javascript and Flash which are solely handled by browser, and all the APIs that it uses are cross-platform. Hence, **WebCollab** is platform-independent. It means **WebCollab** can be run consistently in any operating system that have the required browser and Flash plug in.

7.3. Reusability

WebCollab is comprised of different modules, e.g. Whiteboard, WebCam and Messenger which are written in the form of classes. Each module is independent and can be separated to use in other applications. For example, the Whiteboard can be use in the all-in-one website design application Asalta as a drawing tool, or WebCam can be integrated in the TiddlyCard Messenger for video chatting.

7.4. Extensibility

WebCollab use Prototype library for its class system, hence, it classes can be extended to provide additional functionalities. Other applications can be easily plug in into **WebCollab** with minimal modification to the **WebCollab** core. For instances, a collaborative Wiki application can be added to WebCollab by simply adding a handler for the Comet messages transmitted.

Moreover, the entire TiddlyCard system, which includes **WebCollab**, is an open source application and it will be available in the SourceForge [20]network soon. Therefore, any developer can easily access **WebCollab** source code to modify and improve it. The potential growth of **WebCollab** is very promising.

8. Future Works

8.1. Log in

Currently, there is no user log in mechanism in **WebCollab**. The user ID is auto generated and assigned to the user. Each time the user starts the **WebCollab**, he will be given a different user ID.

This problem can be solved by having a central database for account registration and account authorization.

---

[20] http://sourceforge.net/

8.2. Compression

The standard Comet protocol only supports JSON format and makes it hard to compress data before pushing into the network. In our Whiteboard and Messenger applications, the data exchanges are relatively small, so the compression is not very necessary. However, in the WebCam application, the data is a WebCam image in raw, uncompressed BMP format [21]whose size can vary from 50KB to a few hundreds KB depending on the resolution (in our application, we set the resolution at 100x100) . If the user engages in more than 2 WebCam connections, the bandwidth consumed will be a few hundreds KB to few MBs. This is a tremendous wastage of bandwidth, and makes WebCam application less attractive to users.

The key to this problem is to compress the data at the sender and decompress at the receiver. There are 2 approaches to deal with the compression:

1. Compress the image in binary format (e.g. JPEG[22], PNG[23]) and use file transfer protocol to send across the network.
2. Compress the JSON data using algorithm such as run-length[24], Huffman coding[25].

The second approach is more feasible because it doesn't require any addition implementation of new protocol in the server side. In order to implement the second approach, we simply need pass the JSON data to a compression function and publish the returned compressed data instead.

8.3. Integrate with other TiddlyCard applications

The capability to integrate one module with another in the TiddlyCard system is what makes TiddlyCard a highly expandable platform. We are looking at the possibility to integrate the Whiteboard application with the all-in-one website design Asalta so that the user can freehand draw and decorate the website. The WikiSpy can be integrated with **WebCollab** to enrich the collaborative environment with a real-time text-editor collaboration.

---

[21] http://en.wikipedia.org/wiki/BMP_file_format

[22] http://www.jpeg.org/

[23] http://www.w3.org/Graphics/PNG/

[24] http://en.wikipedia.org/wiki/Run-length_encoding

[25] http://en.wikipedia.org/wiki/Huffman_coding

9. Related Works and Comparison

9.1. Instant Messenger Application

Instant Messenger applications such as Microsoft Messenger (MSN) , Yahoo! Instant Messenger(YIM) are the most related to **WebCollab**. Recently, MSN and YIM have introduced their browser-based Messenger system.

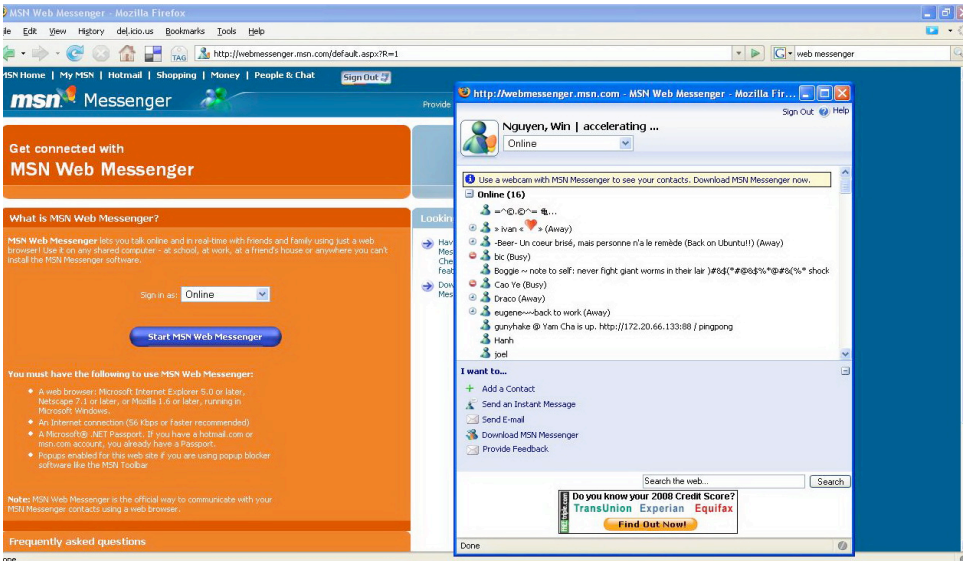

Figure 23 MSN Web Messenger [26]

[26] http://webMessenger.msn.com

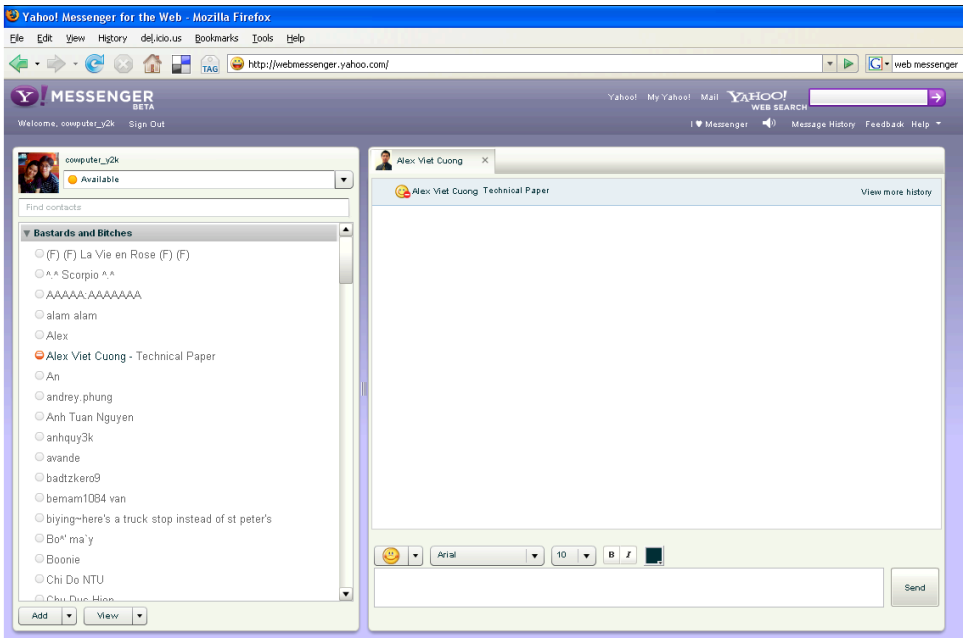

Figure 24 YIM Web Messenger [27]

However the features available in the Browser-base version are very limited as compared to the OS-based one. They are simply Messenger applications that provide chatting. There is no video, no voice-over-IP, and no file transfer support. In this sense, **WebCollab** is superior as it provides a more dynamic collaborative environment with more applications such as WebCam, Whiteboard.

Moreover, the GUI of these MSN & YIM is more intensive, and the bandwidth to the web application is rather slow. The users can notice a significant lag from the moment he sends the message to the time that the message reaches the receiver. This latency can be very irritated and hinders many users from using it. **WebCollab** simplifies the GUI so that it can be loaded faster. Hence, **WebCollab** is more suitable for users with lower bandwidth.

---

[27] http://webMessenger.yahoo.com

9.2. Imagination Cube

Imagination Cube is a Flash application that allows users to sketch freehand with standard basic drawing tools.

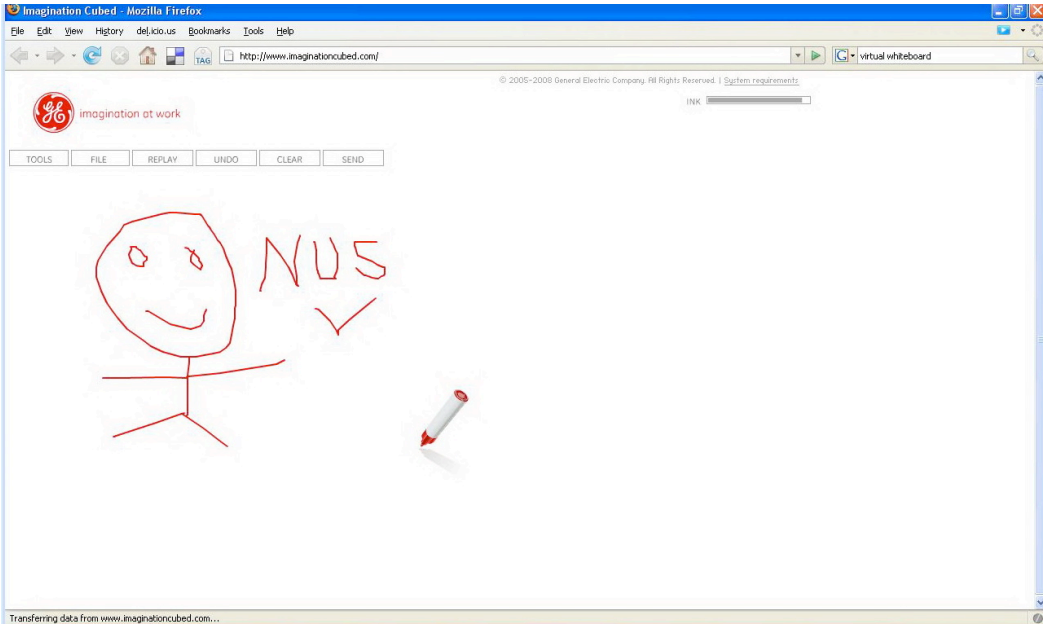

Figure 25 ImaginationCubed Whiteboard Application[28]

This application is a dedicated Whiteboard only without any other supplementary application. It is likely the most vivid and interactive web-based Whiteboard. However, there is no real-time collaboration involved. It's written in Flash so the loading time is slower than **WebCollab** 's Whiteboard application which is purely in Javascript.

9.3. Vyew

Vyew [29]is a collaboration tool suite that lets users share a common online visual workspace and desktop. It offers a comprehensive tool set that includes real-time desktop sharing, white-boarding and drawing tools, text Messenger, Voice over IP, and many more.

---

[28] http://www.imaginationcubed.com

[29] http://vyew.com/site/

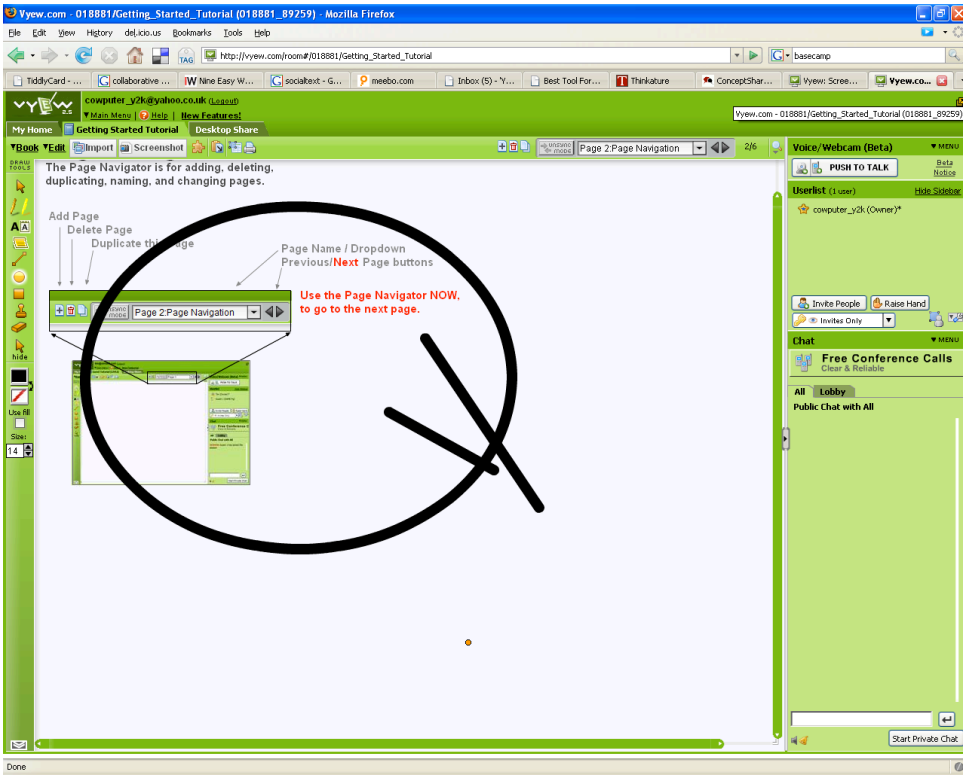

Figure 26 Vyew Workspace

Vyew is an impressive collaboration tool but at a high cost. Hence, it is only suitable for big companies that can afford the price. Moreover, the size of the suite is very big; it consumes large bandwidth and takes quite a long time to load. Faster computers and bandwidth are required in order to run Vyew smoothly. It is overly complicated for those who just want to a simple collaboration tools like **WebCollab** which they can start straight away without going through a tutorial.

10.Conclusion

The key purpose of **WebCollab** is to provide a free and simple, yet powerful enough, collaboration toolkit for non-professional collaboration. Its simplicity, speed and availability make **WebCollab** stand out from other commercial, professional software. There are a lot of rooms for improvement and the potential growth of **WebCollab** is very promising as its source code will be publicized on SourceForge network.