THE UNIVERSITY OF WAIKATO
Te Whare Wānanga o Waikato

Improving Collaborative Drawing using HTML5

Yu-Hsin (Amigo) Huang

This thesis is submitted in partial fulfillment of the requirements for the Degree of Master of Science at the University of Waikato.

April 2014

Abstract

This research looks into improving online web-based collaborative drawing using HTML5. Although many systems have been developed over a number of years, none of the applications released have been satisfactory for many artists; the core drawing experience was too different from a stand-alone drawing applications. Stand-alone drawing applications have better freedom of control with functions like undo and allow artists to work efficiently with hotkeys. The advent of the HTML5 Canvas Element and Websockets in recent browsers has provided new opportunities for collaborative online interaction.

This research used an incremental development approach to build a prototype HTML5 drawing application providing new functionality for online collaborative drawing. The project was supported by two experienced artists throughout investigation, design, implementation and testing. The project artists helped validate design decisions and evaluate the implementation.

As a result, a robust HTML5 collaborative drawing application was built. The prototype contains core drawing functionality that existing applications did not. Features include: undo and redo, free canvas transformation, complex hotkey interaction, custom canvas size support, colour wheel, and layers. All these features work smoothly in a fully synchronized network environment under a client-server model. The collaboration system uses an authoritative server structure with local prediction and re-synchronization to hide latency.

Although the result is only a prototype, the evaluations from the project artists were very positive. Once more functionality targeted towards social interaction is built, the prototype will be ready for mass public testing. Although there are some issues caused by the immaturity of HTML5 technology, this project affirms its capability for collaborative web applications.

Acknowledgement

I would like to thank my supervisor, Bill Rogers, for making this research possible. Thank you Bill for being supportive of this research and fixing up my bad English. Your wisdom and contribution to this research was valuable and deserves my highest respect. I would like to thank the two project artists' for their participation and effort put in to this research. This research would not have been this successful without the knowledge and opinion of the project artists. The two project artists will remain anonymous in this thesis due to the agreement under the Ethics Consent, but they will be named in the prototype application website. Doing this research from an incremental methodology with end-users perspective was fun and interactive; I hope more academic research in the future could have similar approaches.

Contents

Chapter 1: Introduction

Online web-based collaborative drawing has become a popular service in art communities in recent years. A collaborative drawing application is an application where people can draw and chat together in a shared environment. A web-based application, or web application, is an application that runs inside a web browser; as long as the necessary plug-ins are installed, any web application can run from a modern browser without further installation or updating.

Web-based collaborative drawing is popular among artists because it can be used for social interaction, education or exposure between artists over the internet. But more than 5 years have passed since the release of the currently popular applications, and little attempt has been made to improve the core drawing experience of these applications. The difference in drawing experience between stand-alone drawing software and web-based collaborative drawing applications is still too great. Collaborative drawing applications still lack basic functions such as undo or free canvas transformations. This clearly raise a burning curiosity in many end-users; why is it not possible to have a better collaborative drawing experience.

Conveniently HTML5, the new standard for providing web interactivity, have gained strong support by web browser developer since its announcement in 2008. With improved performance of modern computers over time, web browsers also became much powerful at running complex web applications. One of the most notable improvements of HTML5 from HTML4 was the ability to display dynamic graphic content without the need of any plug-ins. With all the advantages of being a web application and no need of plug-ins, HTML5 is a plausible candidate as a base technology solution to this research. Therefore, the purpose of this research is to evaluate and improve the quality of web-based collaborative drawing experience through the use of HTML5 technology.

With help and guidance from experienced artists, an HTML5 prototype was built addressing the requirements to improve the web-based collaborative drawing experience. The development followed a methodology of expert guided incremental development; frequent consulting and testing sessions are held with artists who have good experience in this topic. The artists provided expert knowledge and advice on the requirements and give evaluations of implementations throughout the research.

This thesis explains how identified issues were solved with HTML5 and various other pieces of technology, then evaluated against experienced end-users. Notable technology used is: HTML5 Canvas element, Websockets, Node.js, Javascript, and CSS. Some of the main challenges addressed are: network synchronization, network prediction, undo, canvas transformation, layers, web browser performance, hotkey interactions, and user interface control features.

This thesis is divided into 12 chapters. Chapter 2 describes the background of how this topic was identified and why this is an important area of work. Chapter 3 describes how this research was approached and conducted over time. Chapter 4 explains the technology used in developing a prototype system. Chapter 5 is a study of notable existing web-based collaborative drawing applications. Chapter 6 is where the requirements for the prototype are define with help from the project artists. Chapter 7 looks into previous related academic research relevant to the development of the prototype. Chapter 8 describes the implementation of the main system and back-end of the prototype. Chapter 9 describes the implementation for user controls and front-end interaction. Chapter 10 is about the final evaluations of the prototype. Chapter 11 discusses considerations of additional work to build on the prototype in order to make a more complete drawing system. Chapter 12 presents conclusions drawn from this research.

Chapter 2: Background

With increase in connectivity between people through the internet, the desire for highly expressive means of interaction have arisen. Chat rooms or instant messaging over internet are a very common technology and many people use them every day. It is natural for people to start looking for a more visual interaction rather than just text. Ever since the introduction of plug-ins to web browsers, interactive web applications have grown in study and use. The most notable features of a web applications compared to native software are that they are often cross-platform and have low pre-installation requirements. Single-user drawing tools amongst other web applications were developed and became popular tools for some people. Studies of web-based collaborative drawing applications was released as early as 2004 (AlRamahi & Gramoll, 2004). Most studies of collaborative web applications were focused on usage in education, communication, rapid prototyping, or basic sketching (Sangiogi, Beuvens, & Vanderdonckt, 2012) (AlRamahi & Gramoll, 2004). Rarely did early studies focus on providing a better drawing experience for collaborative artwork. Interestingly that work was embraced by art websites with releases of web-based collaborative drawing applications around 2009 (Pixiv, 2009). In recent years, web-based collaborative drawing applications have risen in popularity. Art communities use collaborative drawing applications as a mean of social, exposure, or mentoring service. Due to the popularity of the early applications, many new collaborative drawing applications were released later.

An artist, who uses collaborative drawing applications regularly, approached the researcher and raised the question; 5 years have passed since the release of ,what he considered, the most popular collaborative drawing tool: Pixiv Chat. The base technologies and interactions still have not changed much over time; many new collaborative drawing applications came out during those 5 years, but important issues related to the basic drawing experiences were still not properly addressed.

Commercial stand-alone drawing software packages all have functions like undo, canvas transformations, and complex hotkey interactions; these functions help to give the user freedom of control, convenience, and efficiency in their drawing experience. In contrast online collaborative drawing tools are still nowhere near this level of interaction. The artists claims it is the lack of these functions that is preventing the technology from becoming even more popular. The artist feel it is because developers were unaware of this problem, so many new applications release only providing more tools than others. There are other minor of issues in existing applications such as the requirement for plug-ins, cluttered user interfaces, small canvases…etc. These minor issues does eventually add up to a poor drawing experience.

With the announcements of HTML5 in 2008 (W3C, 2008), new opportunities for interactive web applications became a popular topic. One of the major opportunities offered by HTML5 was to provide interactive web experience through any modern web browser without the need for plug-ins. However, at that time, most specifications in HTML5 were still in early draft and adoption in web browsers were slow and few. Two of the first introduced specifications of HTML5 were the Canvas element and the Websockets. The Canvas element allows dynamic scriptable graphics content on a webpage and the Websockets allows low latency, bi-direction TCP communication between server and client through port 80. Over the years, both specifications have become fully supported by all modern browsers. With this knowledge, the researcher proposed to address the questions asked by the artist using HTML5 technology.

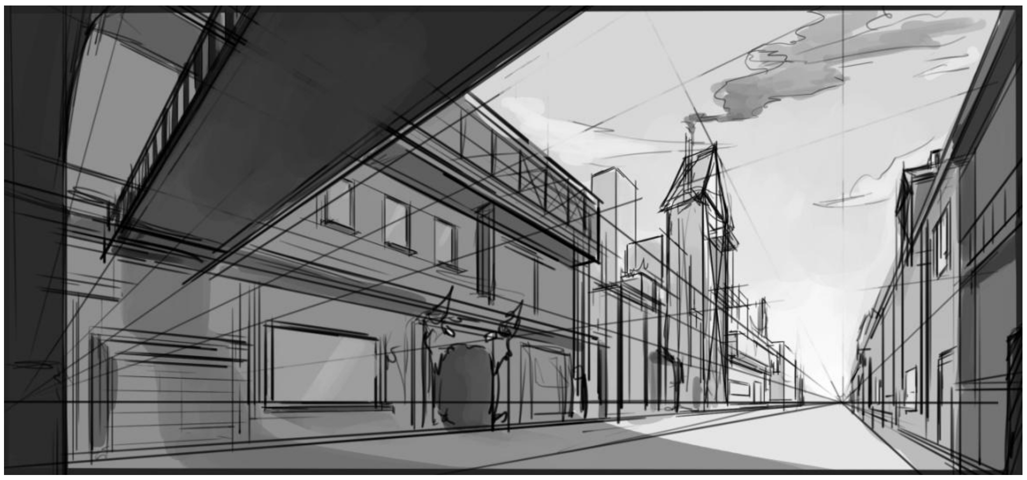
Chapter 3: Project Approach

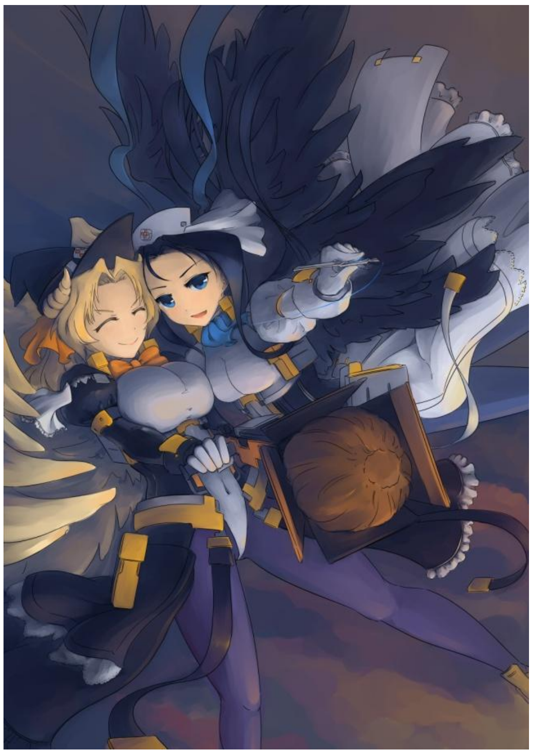
The goal is to build a HTML5 collaborative drawing application prototype with functions considered important to common stand-alone drawing software that current existing collaborative applications lack. The artist who suggested this research idea, along with another artist friend, have agreed help this project as expertise consultants. Both artists are professional digital artists and have experience in using online collaborative drawing applications. Throughout this research they will be referred as “project artists” and are expected to help this research from a end-user point of view. Many small casual consultation sessions will be held between the researcher and the project artists to discuss decisions or comments on this research. The focus is to implement drawing related functionality that the project artists have identified as the most apparent problems with existing applications. Social interaction, or social oriented functions will mostly be a secondary scope of this project. The reason for this focus is because the project artists feels the biggest problem with existing applications is the drawing experience, not the social experience. Collaborative drawing applications already have achieved a good social experience by their nature; people can draw and chat in a shared environment. However there are still some important issues in social experiences that should be considered, in the view of the project artists.

The researcher will study notable web based collaborative drawing applications; some will be identified and commented on by the project artists. A set of requirements for the HTML5 prototype will then be constructed with the knowledge gained from the study and aid of project artists. As parts of the prototype are being developed, small testing sessions will be held to get feedback and comments from the project artists. These testing sessions are mostly very casual; sometimes it is only done with one artist within few minutes. Once the prototype is complete, larger testing sessions will be held. The larger testing sessions consisted of the project artists collaborating on one drawing for roughly 20 minutes. Multiple larger testing sessions will be held to get an accurate evaluation of the final prototype.

This research has the approval of University of Waikato Ethics Committee in conducting testing/consultation session with targeted artist. The project artists will remain anonymous and have the right to disassociate from this research at any time. The project artists own all the rights to their published drawings in this research. Only comments and/or screens will be recorded during any testing session.

Example works of project artists on stand-alone drawing software:





Example works of project artists on online collaborative drawings

[redacted] [redacted] [redacted] [redacted] [redacted] [redacted] [redacted] [redacted] [redacted]

Chapter 4: Technology

This chapter describes the main technologies that was used in this research.

4.1 HTML5

Hyper Text Mark-up Language(HTML) is the base technology of web pages. It is the standard that defines the contents of a web page. HTML5 is the base technology for this project. Ten years had passed since HTML4’s release, HTML5’s release in 2008 was a significant contribution to web technology advancement. HTML5 is often compared to Adobe Flash due to both having their goal of improving web interactivity and multimedia support. However the two technologies have completely different nature in structure and behaviour. Flash is compiled and proprietary, and HTML is interpreted and open. There is no access or interaction available for other web objects in a Flash application, this can be problematic to other web services, preventing the extraction of meta data from Flash content in a website. Flash software is owned by a company. Licenses and plug-ins are required in order to develop or run in Flash. The future of Flash is solely in control of one company. HTML5, on the other hand, is just a new standard, improved from HTML4. HTML specifications are developed by the World Wide Web Consortium(W3C), the process is more community driven with less commercial constraint. Because HTML is just a set of rules and descriptions, it is the web browser developers’ responsibility to comply with the standard. Hardware is never likely to be an issue with HTML while Flash requires both compatible browser and plug-in to work on each kind of computer hardware. One obvious issue with HTML5 is that it is still under development. The specifications are still evolving and not finalised. Browsers are updating and developing different parts of HTML5 in their own pace. Different browsers have different display and code for the new specifications.

The HTML5 Canvas element and the Websockets were in the first specifications introduced. The Canvas element had already existed in Safari and Opera browsers prior to first draft of HTML5, so it was also one of the earliest specifications adopted by other browsers. The Websockets, however, was only later added and fully standardized in all browsers until late 2011. This project will focus on combining the Canvas element and Websockets as its base technology. Both features are available in modern web browsers at time of development and are accessed in Javascript. The Canvas element allows dynamic scriptable graphics in a web browser. It is stored in a bitmap structure, hence it is raster-based. The HTML5 SVG element is an alternate structure to graphics in HTML5. The SVG element supports vector-based graphics. This project focuses more on raster-based drawing experience than vector-base, so the Canvas element was chosen as the base technology. An additional graphics technology built on top of HTML5 Canvas is WebGL(Web Graphics Library) (Khronos Group, 2011). WebGL uses the HTML5 Canvas element as a display portal but has a separate API and structure similar to Open GL ES 2.0. WebGL uses Javascript to run GPU accelerated content. However the biggest limitation of WebGL is its graphics card dependencies. Not all graphics cards, especially mobile graphics cards, support WebGL (Moelker & Wijbrandi, 2012). This project hopes to reduce software and hardware dependencies to only having installing a modern browser, so WebGL was not considered as a base technology. Another reason why WebGL is not suitable for this project is because WebGL is mainly targeted towards rendering 3D graphics. Its back-end structure relies on Javascript feeding vertices, shaders, and textures into the GPU. There may be a work-around to achieve a drawing canvas application in WebGL, but the HTML5 Canvas element providses many drawing and pixel manipulation functions by default, so is much suited for a 2D drawing application.

The Websockets is a new protocol that allows clients to communicate with servers over sockets, much like stand-alone software. The connection is full-duplex and uses TCP over port 80. Because Websockets is a protocol, it requires both server side software and the client side browser to both support the specification. Prior to Websockets, communication between server and client was only available through client side requests. This means the server was always passive and could not actively send information to the client. In early Internet, communication between client and server only contained request of full web pages. After the introduction of iframes in 1996 by Internet Explorer, a new technique was developed with it that allowed clients to asynchronously request partial content from a server without the need to refresh the web page. This allowed more dynamic communication between the client and server. This technique was later popularised by Google in 2004 and now referred to as AJAX. However even with AJAX and its variations, the server is still passive and all requests still contains HTTP header overhead per-request. AJAX is incapable of achieving low-latency and bi-directional communication. An alternative method to achieve socket-based communication was using FlashSockets in Flash. However the availability of services from web hosting services are limited because it require servers to be specially setup to allow FlashSockets and also require a license purchase of Flash Server (AlRamahi & Gramoll, 2004). Another method to achieve to socket-based communication was through Java Applets. But, like Flash, a plug-in is needed to be installed and activated upon usage. A study released in 2011 (Gutwin, Lippold, & Graham, 2011) clearly showed that Websockets have the best performance compared to all other alternatives for achieving real-time web-based networking from web pages. However Websockets, at time of this project, are still slowly being integrated into common server side software and services.

4.2 Node.js

One of the early Websockets adoptions in a server application is Node.js, sometimes referred just as Node. Node.js is an asynchronous Javascript server application released in 2009 (Joynet, 2009). The first major difference to other server applications is that Node.js uses Javascript as a programming language. People may have misconceptions of the network handling power of Node.js due to the slower processing nature of Javascript. While the server runs Javascript, the application itself is written in C; this means it is completely capable of handling heavy networking needs. Javascript also has more supporting resources available compared to C due to the larger number of web developers than server programmers. Node.js therefore lowers the bar-of-entry to server programming by not requiring a knowledge of C language. The front-end of this project will be written in Javascript too, so having both ends developed in same language is a great convenience. The second major difference of Node.js to other server applications is that it uses an event-driven architecture. Node.js is single threaded and handles all requests and connections with events and call-back functions. This means requests can be handled concurrently and only use resources when needed. The asynchronous architecture also allows non-blocking I/O, so the server does not have to worry about preventing deadlocks and thread synchronization. Traditional server applications are commonly multi-threaded; each connection and request would have a larger process overhead and resource allocation than Node.js. The Node.js structure also allows the system to scale well for different number of connections and handles smaller requests well. All above behaviours makes Node.js a well suited server for a collaborative drawing application. The most obvious disadvantage of Node.js is its computational power. Javascript is still an interpreted language, so server computation process will always be slower than those written in traditional compiled languages. For this project, the server machine will not be running a client. This means that no graphics computation will be run server side. The server will only need to act as a message relaying machine and authenticator. This should not impose too heavy of a processing requirement.

4.3 Open Shift

Performance in networking is a major factor to this project. In order to simulate and address realistic network latency, the experimental server of this project should be remote. Network latency is an obvious issue in all network applications. Running experiments on localhost or with a LAN setup will give an inaccurate impression of the user experience. At the time of the project, finding a remote web hosting service with Node.js support was not difficult. But finding a Websockets supporting services with low cost was more difficult. Websockets was still quite new and experimental, many web hosting services only allowed it as a premium service. Fortunately, Open Shift provides both Node.js and Websockets support with their Free Plan. Open Shift (Red Hat Enterprise, 2013) is a PaaS(Platform as a Service) by Red Hat Enterprise. It is a Linux cloud computing service that provides the user with a computing platform that automatically scales to its service needs. This means the user does not need to manage the underlying hardware or software as it is all automatically managed and scaled by the web hosting service. It allows users the convenience of focusing on their own website or service to be run on the platform. All servers are located in US at the time of project. This provides a good latency test from New Zealand. The Free Plan provides up to 1.5GB memory, up to 3GB storage, and no bandwidth limit. These specifications are well sufficient for testing among small groups. Note, mass public test is not within the scope of this project. The Open Shift service provided excellent conditions for this project at no cost.

4.4 Browsers

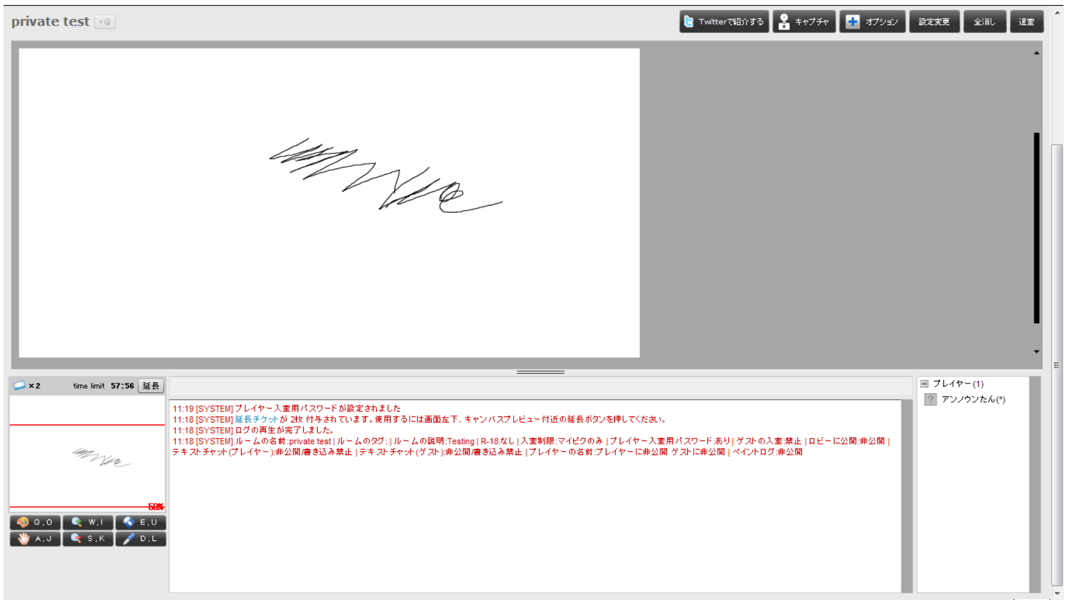
A modern browser is theoretically all that is needed to run HTML5. But as previously mentioned; not all specifications are finalised; not all browsers are up to date; some browsers went ahead and created their own syntax/implementation; and different browsers handle tasks using their own methods in the background. Constant testing on different browsers will be very time consuming, so this project will only mainly focus on testing with Mozilla Firefox as the main platform. Testing on other browsers will only be occasional. There are no particular reasons for the choice of browser, the researcher happens to be most familiar with it for development. At time of this project, Firefox (version: 28) supports all the technologies needed for this project.

Chapter 5: Existing Applications

Web-based collaborative drawing is not a new concept. Studies of such ideas implemented with Flash exists as early as 2004 (AlRamahi & Gramoll, 2004). Most popular web-based collaborative drawing applications to this date are still built with Flash. Many applications existed, but have been taken down from the web for to various reasons. One common reason for those taken down was because the bandwidth usage was too large and expensive to maintain. With availability of Open Shift's services, maintenance is not a concern in this project for now. Below is a analysis of popular existing web-based collaborative drawing applications. Note, online collaborative applications uses the term "room" or "session" to describe one collaboration environment; the two terms may be used interchangeably at times.

5.1 Pixiv Chat (Pixiv, 2009)

Pixiv is the largest online art community website. First launched on 2007, now has over 5 million members and 3.3 billion monthly page views. Pixiv is a website where users can submit and share their art work. In 2009, Pixiv launched a Flash chat room service with drawing collaboration called "Pixiv Chat". Pixiv Chat has become a popular social ground for artists to meet, learn and collaborate. It allows more dynamic interaction than traditional chat rooms. All Pixiv members can create and participate in public or private rooms. Non-members can still join a public room as spectators. Pixiv Chat only has three tools: pen, eraser, and colour sampler. The pen and eraser both have options: binary edge and soft edge option, radius control, opacity control, and colour control. There are three view controls: zoom-in, zoom-out, and pan view. The zoom function has three levels of zoom: 50%, 100%, and 200%. The canvas size is a fixed 1600x1200 with two layers available. The project artist state it is one of the best collaborative drawing tools that exists.



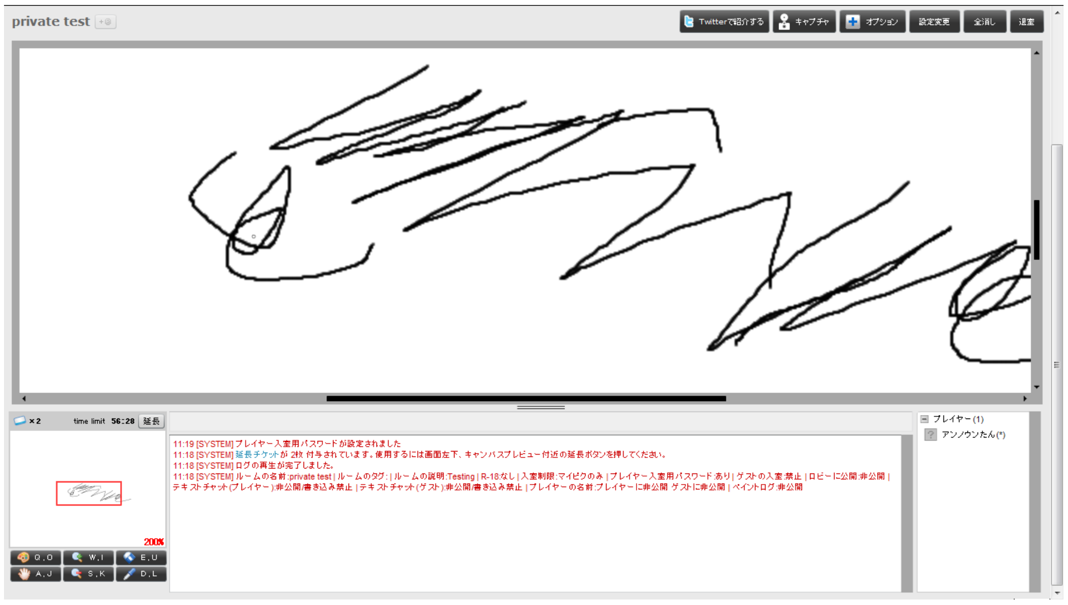
```
11:19 [SYSTEM] プレイヤー入室用パスワードが設定されました
11:18 [SYSTEM] 延長チケットが 2枚 付与されています。使用するには画面左下、キャンバスプレビュー付近の延長ボタンを押してください。
11:18 [SYSTEM] ログの再生が完了しました。
11:18 [SYSTEM] ルームの名前:private test | ルームのタグ: | ルームの説明:Testing | R-18:なし | 入室制限:マイピクのみ | プレイヤー入室用パスワード:あり | ゲストの入室:禁止 | ロビーに公開:非公開 |
テキストチャット(プレイヤー):非公開/書き込み禁止 | テキストチャット(ゲスト):非公開/書き込み禁止 | プレイヤーの名前:プレイヤーに非公開 ゲストに非公開 | ペイントログ:非公開
```

Example view of Pixiv Chat.

5.1.1 Notable features of Pixiv Chat

Canvas Mini-Map and View Transformations

There is a small thumbnail image that displays the current state of the canvas as a whole. This helps the user be aware of the whole canvas while focusing on a smaller section. Most collaborative drawing applications do not have a canvas thumbnail because of smaller canvas size or lack of view control. The user can pan and zoom the view of the canvas. The thumbnail can be clicked to move the pan position.



```
11:19 [SYSTEM] プレイヤー入室用パスワードが設定されました
11:18 [SYSTEM] 延長チケットが 2枚 付与されています。使用するには画面左下、キャンバスプレビュー付近の延長ボタンを押してください。
11:18 [SYSTEM] ログの再生が完了しました。
11:18 [SYSTEM] ルームの名前:private test | ルームのタグ: | ルームの説明:Testing | R-18:なし | 入室制限:マイピクのみ | プレイヤー入室用パスワード:あり | ゲストの入室:禁止 | ロビーに公開:非公開 |
テキストチャット(プレイヤー):非公開/書き込み禁止 | テキストチャット(ゲスト):非公開/書き込み禁止 | プレイヤーの名前:プレイヤーに非公開 ゲストに非公開 | ペイントログ:非公開
```

Canvas mini-map on the bottom left hand side.

Two layers

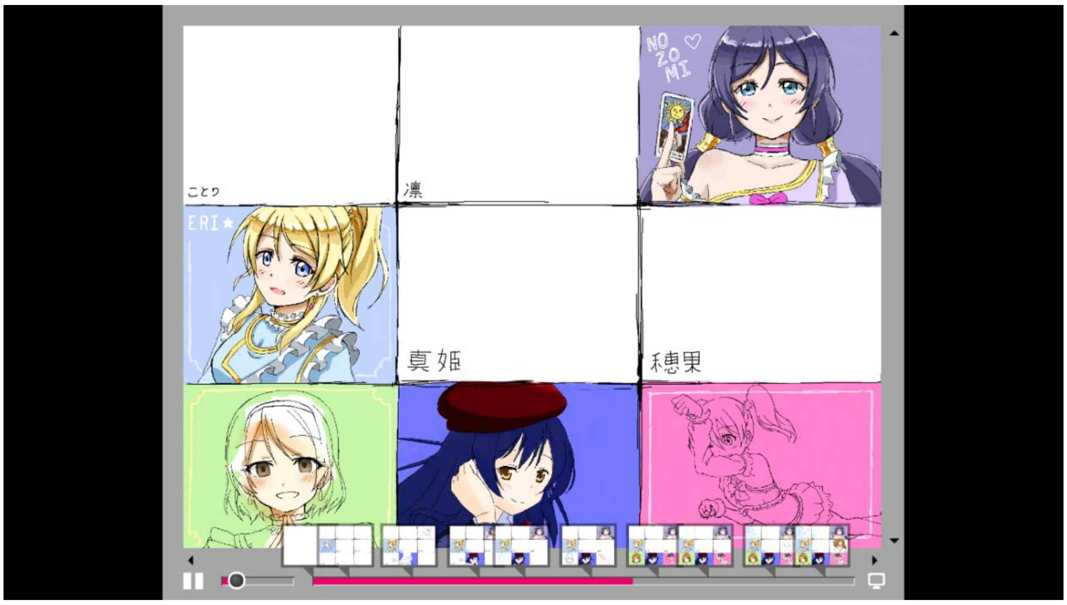
Having two layers allows more complex drawings. Most other collaborative drawing applications do not have layers. If a drawing application only has one layer the method of drawing will only allow a paint-like approach. This means new strokes are merged onto earlier strokes. Having another layer allows the more common digital approach where the user can draft on the bottom layer and then do clean artwork on the top layer.

Ticket System

Pixiv Chat implemented a "Ticket System" to control the length of each room. Because all members can create a room, server load would be difficult to manage. All rooms are given a life time and they will shut down when the time ends. To extend the time of a room, Tickets need to be added. Each member gets two Tickets a day; more tickets can be obtained by a room owner if participation level in that room is high. This system allows the lifetime of each room to be supported by the participants, which is a fair way of controlling to the large number of rooms that can exist. This system also helps control the size of the drawing log so it does not put excessive demands on the memory of the server.

Room playback

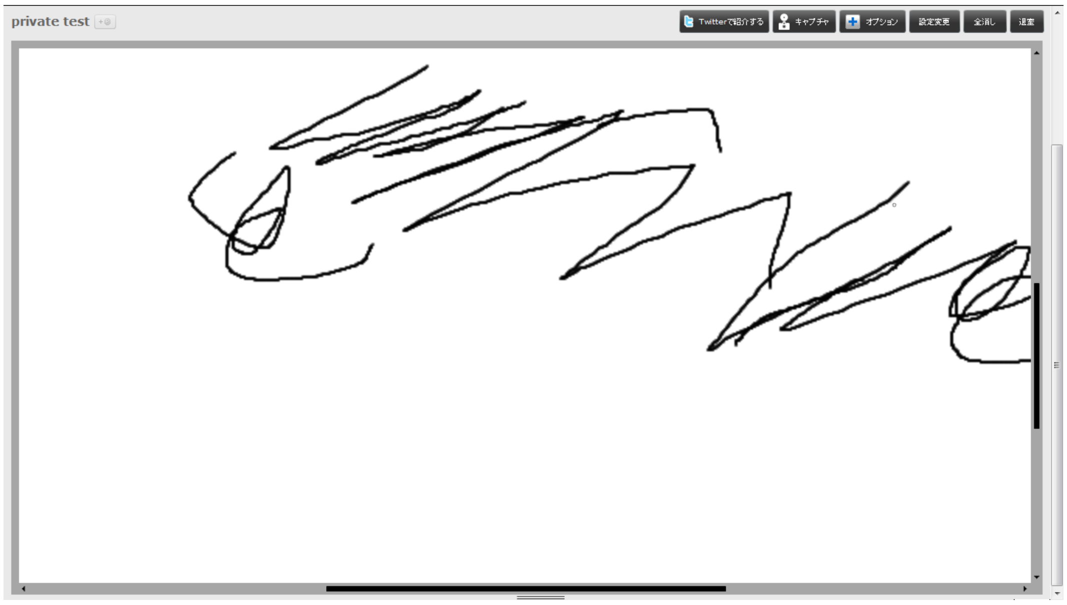
Because a log is remembered for every room, each room can be played back in a player. This is a very good feature allowing users to watch other people draw. Some other drawing applications have this feature too, but Pixiv Chat has more control of speed of execution and timeline.



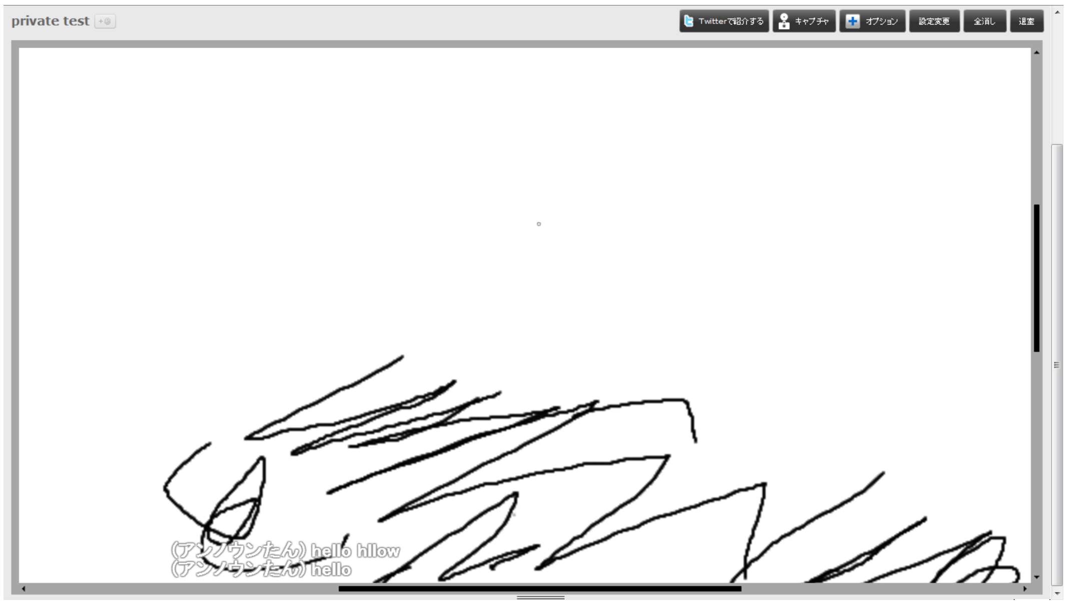
Room playback video built in Pixiv Chat.

User Interface and chat system

Pixiv Chat's approach to its user interface and chat system has the best design in collaborative drawing applications according to project artists. The control interface can be hidden as a whole leaving only the canvas visible. The user can interact with the all tools of the canvas purely with hotkeys. When pen size or colour change is needed, the user just needs to hold the hotkey; a control panel will appear at the cursor and the user can quickly change settings without moving the cursor. This feature is a very important factor for drawing efficiency and user friendliness. To help users keep up to date with chats in Pixiv Chat while the interface is hidden, a separate temporary chat popup will appear when a message is received. This popup will appear for a short period of time before disappearing. This is extremely helpful for people who just want to "listen" to a conversation. If the user wants to reply, a send message option can be popped up with the press of a hotkey, without the need to show the whole control interface. The use of space and hotkeys makes Pixiv Chat very user friendly and efficient to work with.

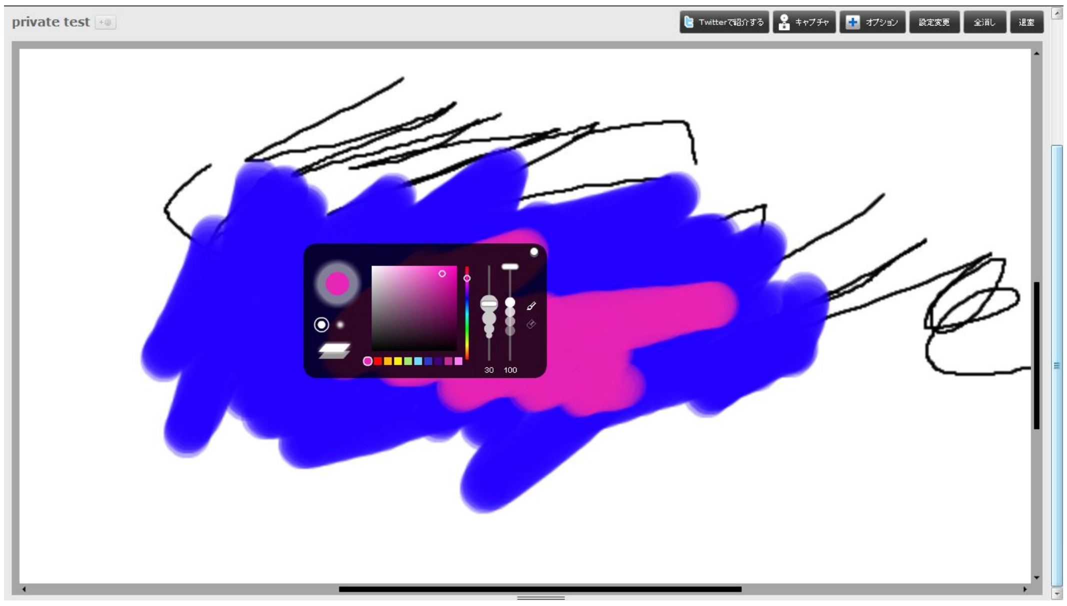


The main control interface can be hidden.



(アンノウンたん) hello hllow
(アンノウンたん) hello

Chat messages can still feed through as little pop-ups.



Control interface pops up where the cursor is from a press of a hotkey.

Large canvas size

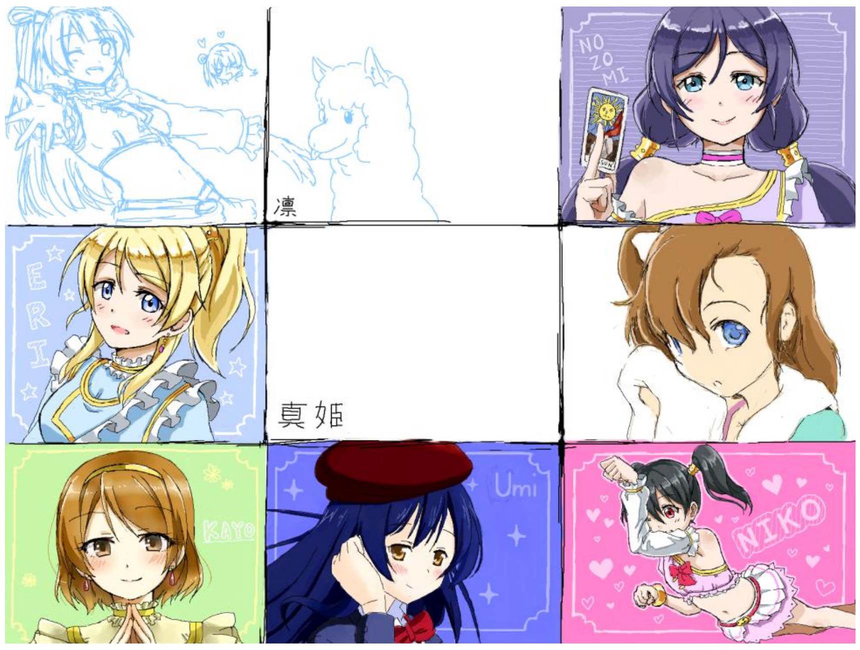
Pixiv Chat have one of the largest canvas sizes of all collaborative drawing applications. This allows more people to participate and also makes people more willing participate because of the extra space.

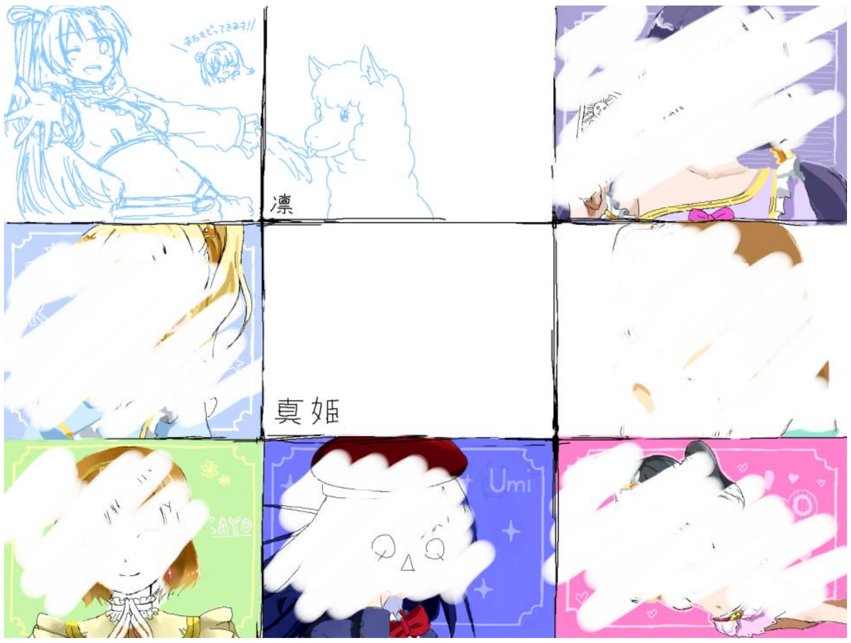
Free Spectate

Pixiv Chat does need registered account to draw, but users can spectate on any public rooms without registering. This is a convenient feature for people who just to want to watch. This feature also introduces the application to interested new-comers without the trouble of signing up.

5.1.2 Notable drawbacks of Pixiv Chat

The project artist feel that Pixiv Chat has provided all the most important features required to draw collaboratively. Most other collaborative applications have more tools, for example: shape tools, stamps...etc. But those additional tools do not contribute much to the core experience of drawing. One minor issue with Pixiv Chat is that the colour selection tool does not provide accurate colour selection, there are no options for numerically entering a pricise colour. This can be a issue as some artists have very specific colour preferences and in Pixiv Chat can only estimate. With all the great features provided by Pixiv Chat, the most desirable addition would be Undo. Mistakes and accidents always happen. If a stroke went through the wrong section on the same layer, the only way to fix it is to re-paint over the mistake. The social impact of these mistakes can be magnified if the user drew over another user's artwork. The lack of Undo also causes people to be less willing to participate. This is because people are too used to commercial drawing software that all has Undo functions, so the drawing style and mind set change can be too great. With any public social platform, there is a risk of negative social behaviours. Even though many online collaborative drawing application use a log in system like Pixiv Chat, there will still be people that intentionally ruin other people's drawings. Hours of work can easily be destroyed by one person intentionally drawing over another person's work.

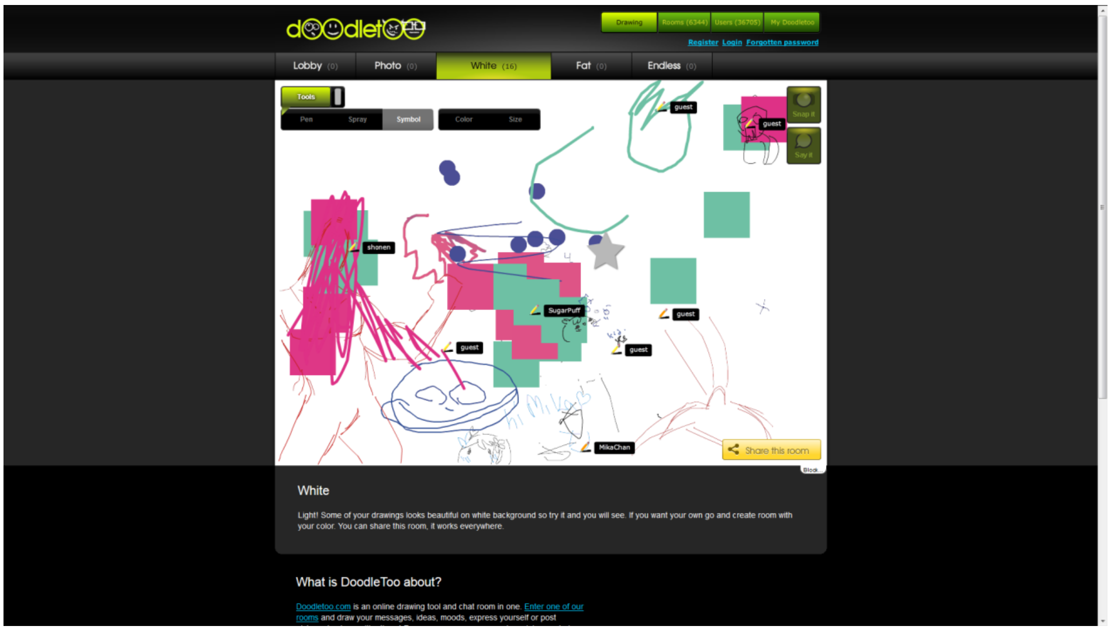




An recorded example of someone who joined, then intentionally cleared the project artist's session.

5.2 DoodleToo (WDF, 2010)

A simple Flash collaborative drawing application. It is one of the few where membership is not needed to participate. The public board never closes, but will only send a fixed number of previous actions back to new connecting users. This means most of time everyone is seeing different things because everyone connects at different times. All drawn lines will slowly fade out over a period of time, which means users cannot really draw anything complicated or detailed. The experience DoodleToo provides is a very limited set of tools and colours. Like many casual collaborative drawing applications DoodleToo has a stamp tool for common solid shapes. DoodleToo rooms have an interesting dynamic compare to other drawing applications. Because no large or complicated drawings can be done, people communicate and draw at a very fast pace. Interaction between users can vary between everyone minding their own business and everyone quickly drawing and responding to someone else's drawing or chat. However, often everyone is just doodling in their own space with no interaction with other. The experience DoodleToo provides is very casual and social centric, it focuses on quick visual communication rather than collaborative drawing. This type of experience is not what this project aims to achieve, but it demonstrates other type of experience available through online collaborative drawing.



White

Light! Some of your drawings looks beautiful on white background so try it and you will see. If you want your own go and create room with your color. You can share this room, it works everywhere.

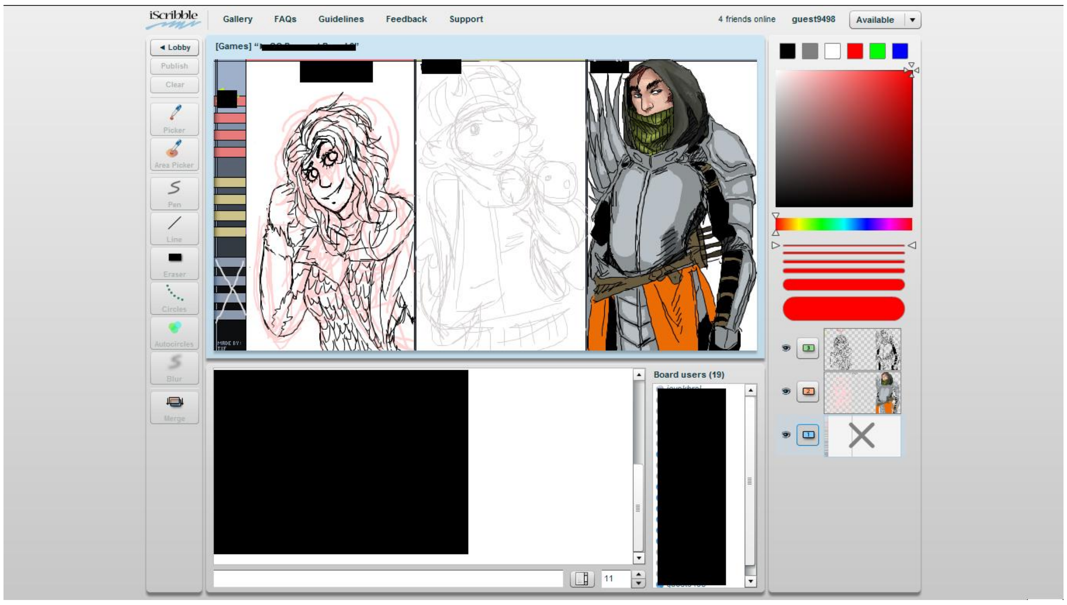
What is DoodleToo about?

Doodletoo.com is an online drawing tool and chat room in one. Enter one of our rooms and draw your messages, ideas, moods, express yourself or post

Example image of DoodleToo application.

5.3 iScribble (iScribble, 2014)

A very popular Flash based collaborative drawing application. There are over 500,000 registered users. The community is highly active, and there are events with prizes at times. iScribble is one of the most popular collaborative drawing application for English language users.



Example image of iScribble application.

5.3.1 Notable features of iScribble

Different Account Types

iScribble has 3 levels of accounts: Guest, Restricted, and Regular. New registered users are under Restricted and not allowed to run certain functions like publishing to their Gallery. To get to the full functionality, a user must participate enough in drawings, have a Regular user to publish their work, and get enough positive votes on that published work.

3 Layers

iScribble, at time of this project, is the only online collaborative drawing application that provides three layers to its users. This feature really boost the type of drawing people can create. The layers can be individually hidden and shown. To prevent users from accidentally drawing on the wrong layer, a coloured border around the canvas is drawn to remind the layer the user is on. This is considered a nice feature by artists. The layer system is overall better than Pixiv Chat’s.

Submitted work gallery and playback

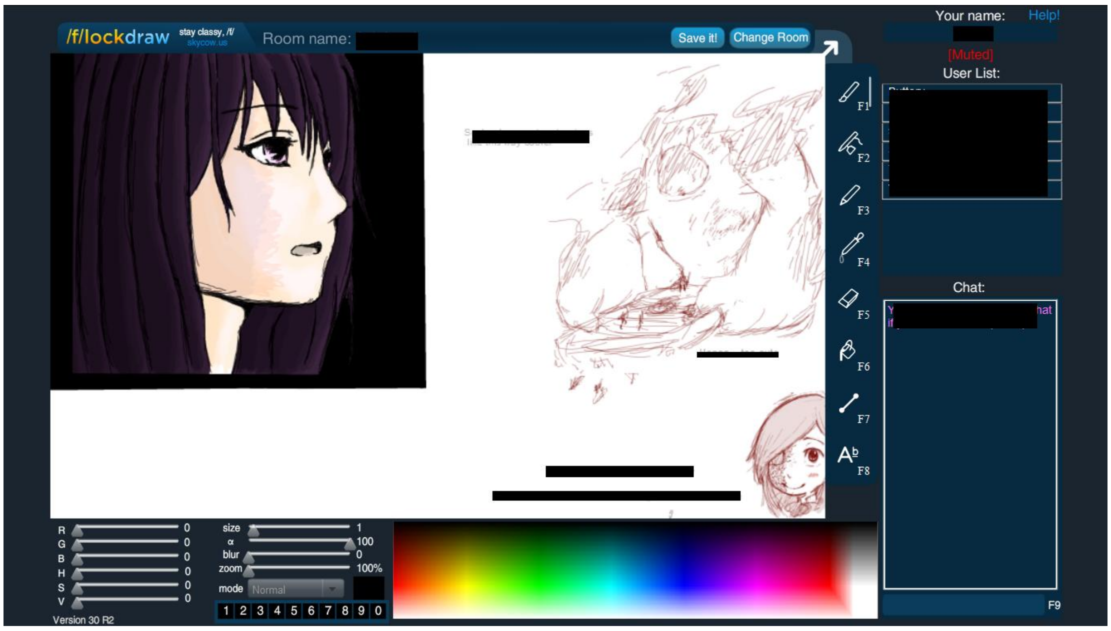
iScribble has a public Gallery where collaborations can be submitted. When a drawing is viewed, a playback of the drawing process to the room can be requested. There are buttons for controlling play speed, but no streaming controls. While Pixiv Chat does have records of rooms, they are more linked to individual profiles and there is no one gallery where all collaborations are shown.

5.3.2 Notable drawbacks in iScribble

The canvas size is still quite small at 700 x 376 pixels. This heavily limits the number of users that can participate. The most common number of collaborators in a room is 3, which is really low. One interesting point about iScribble's tools are that there is no transparency colour control on the colour selection tool. However there is a "Blur" tool that puts blurs pixels together, which helps create some effects that are often created with a transparency brushes. Even though any drawing can be created without transparency control, it still limits the draw style and behaviour of an artist. While iScribble does provide some functions better than Pixiv Chat, the overall experience is still lacking due to the small canvas, basic hotkeys, and basic interface support.

5.4 Sky Cow's /f/lockdraw (Skycow, 2011)

/f/lockdraw is a a modified version of Flockdraw (Flockdraw, 2009). Flockdraw is a Flash based application that provided fairly primitive tools and functions. Spacecow.us managed to create an alternate improved version called /f/lockdraw. /f/lockdraw also does not need an account to participate. There is a good set of tools and options for creating complex digital drawings. All tools are given a hotkey. Colour and other tool settings can only be interacted with at the bottom of the canvas. There are additional hotkeys available but the information is only available through a "Help" wiki website.



Example of image of /f/lockdraw application.

5.4.1 Notable features of /f/lockdraw

Tools

There are total of 9 tools available in /f/lockdraw; 3 type of brushes/pens that use different algorithms to stroke, eraser, colour sampler, paint bucket, line tool, and text tool. The project artists have mixed opinions on the importance of the available tools. The paint bucket, line tool and text tool are all very situational tools. The fear of destroying a drawing with the paint bucket is too high for most artists to really want to use. The line tool and text tool are situational. There are rooms where the line tool is used to segment out user draw spaces. There are rooms where the text tool is used like a banner for informing new users of custom rules or information. Those two tools have been useful for keeping social order at times.

Tool settings

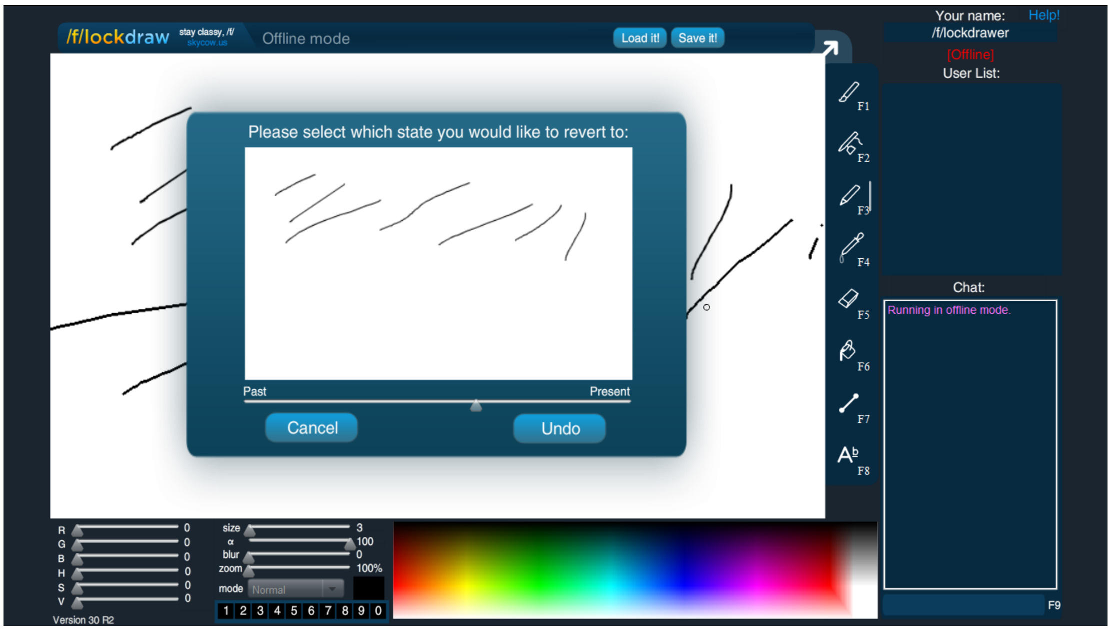
/f/lockdraw provides good number of options for its main brush tool. Two notable options are a brush blurring value and 11 different brush “modes”. The blurring value controls the size of the blur to solid edge in the brush stroke. Most online collaborative drawing applications do not have this much freedom of option. Pixiv only has one set “soft” brush edge that can be used. The various “Modes” give the brush colouring method great flexibility and can achieve many complicated effects that most applications cannot. To accommodate the great freedom of setting brushes, /f/lockdraw provides a “save tool setting” feature where brush colour and settings can be remembered in 10 available slots. Users can use hotkeys to bind and call back from each slot.

Canvas zoom and panning

/f/lock draw is one of the few applications that has free zoom and panning. However the canvas being locked at 800x480 pixel is quite small which reduces the value of this feature.

Moderator Features

To maintain order in a room, a moderator system is used. The room moderator can ban users which will kick them out of the room and send them to a “ban room” where all banned users go. There are no moderators in that room. However because an account is not needed to join rooms, people can still restart the application and give themselves a unbanned user name to rejoin. One additional feature to prevent bad behaviour is that moderators can undo the whole board history. The board undo can go back a number of steps and there is a preview to show what the board was like at that time. This is a great feature to reverse any intentional or unintentional damage to the drawing.



Board undo function in /f/lockdraw.

User re-synchronizing

To help with potential de-synchronization, /f/lockdraw allows a function where the user can “get board” of another user in the room. This function could act as a safety check to make sure the user is synchronized properly with at least one user.

Room Playback

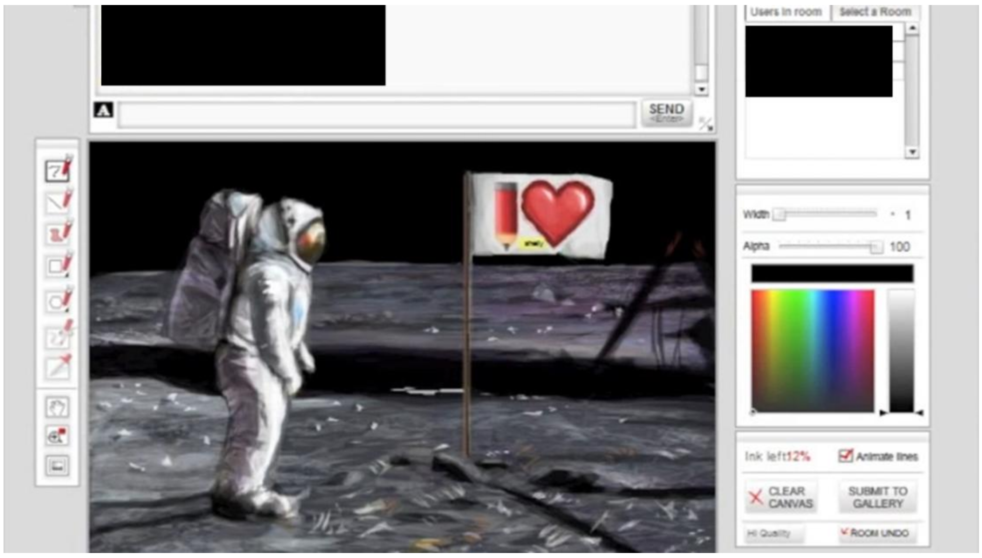
/f/lockdraw’s method of providing room playback is providing a time-lapse video every few days. This mpeg video is manually created and uploaded by the site administrator, so the period and duration varies. /f/lockdraw’s method of providing room playback is crude and not very plausible.

5.4.2 Notable drawbacks of /f/lockdraw

Even though /f/lockdraw does provide a good set of brush settings and tools, it still suffers many issues. The size of canvas is locked at 800x480 pixels, which can be small for people to draw and share in. The lack of layers really limits the painting approach and devalues the good set of settings for brushes. Lastly, the user interface is really basic and slow to use. Even though there are hotkeys the position is widely placed; tools uses F1 to F8, panning requires using the control key and other setting requires using the arrow keys; all these keys are too widely placed for one hand and not suitable as hotkeys. Many frequent interactions like changing colour or chatting cannot be controlled with hotkeys, so the cursor needs to constantly travel to the side of the window. This can cause operations to be slow and can negatively affect the overall experience.

5.5 Rate My Drawings' Draw Chat (Mixart New Media LLC, 2010)

Rate My Drawings started out as a website that aimed to provide a powerful single-user online drawing application. It has built many implementations of its drawing application using Flash, Java and recently HTML5. The implementations of their single-user drawing application contain many complex functions similar to those provided in stand-alone commercial drawing software, especially in the later implementations using Java and HTML5. There are over 60,000 registered users at the time of this project. In 2010 Rate My Drawings modified its Flash implementation to support online collaboration in a system called Draw Chat. Draw Chat provides most of tools that existed in original Flash version but also removed some features. Rate My Drawings is account based, like Pixiv. Any drawings or works submitted are linked to users' profiles. Overall, Draw Chat is primitive and left much to be desired compared to its own single-user applications or other collaborative drawing applications.


[redacted]
[redacted]

Rate my drawings Flash implementation

5.5.1 Notable features in Draw Chat

Account system

Account based participation exists in some online collaborative drawings, but Rate My Drawings have a rule whereby users cannot use Draw Chat until they have been a member of the website for a period of time and have submitted work with the single-user application. The reason for this rule is to ensure that users have invested enough on their account and have been more involved in the community, so they are less likely to behave badly in Draw Chat. This is likely a good solution to keep bad people out of the application, but also created a strong barrier of entry for new users.

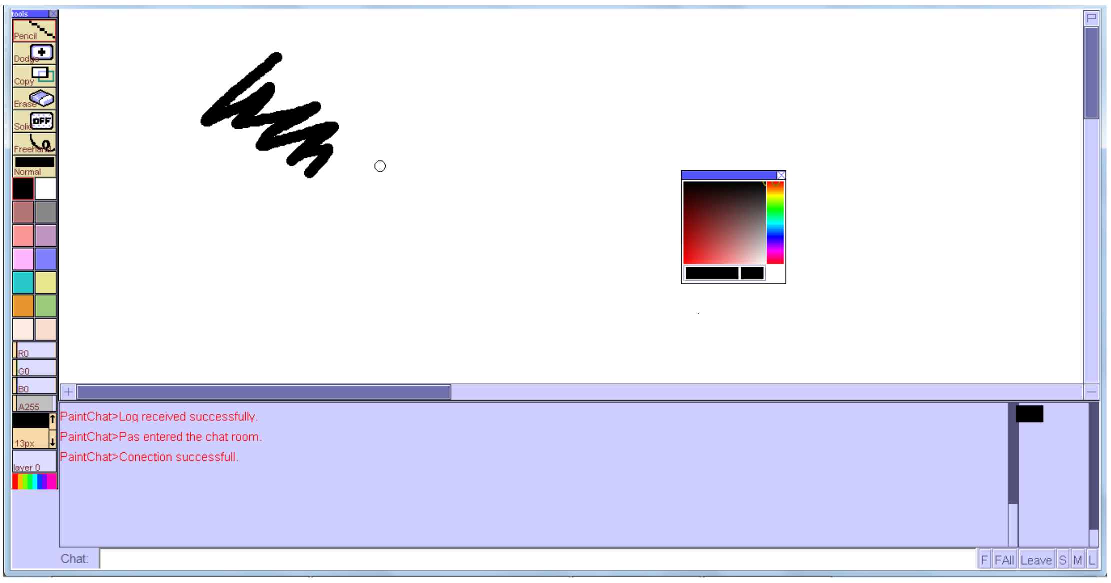
5.5.2 Notable drawbacks of Draw Chat

The canvas is only 640x460 pixels in size which is small. There is an "ink limit" that users cannot exceed. No more actions can be made if the limit is hit. Even though the limit may be high for some drawings, it is still a limitation that may affect the interaction and the final product. The layers function in the original

Flash implementation was removed. The Undo and Redo function has been modified to only a “Room Undo” function. This function similar to /f/lockdraws board undo function, where all actions are remembered in to one history list; if undo is requested, the board steps back in that list regardless of which client made the action. There are no hotkeys supported. The overall experience is not better than most popular Flash based applications.

5.6 Niko’s Paint Chat (Niko, 2009)

Built with Java, Niko’s Paint Chat, at the time of this project, is probably the most drawing feature full online collaborative drawing application. The service requires an account to participate and has accumulated over 29,000 registered accounts since launch.



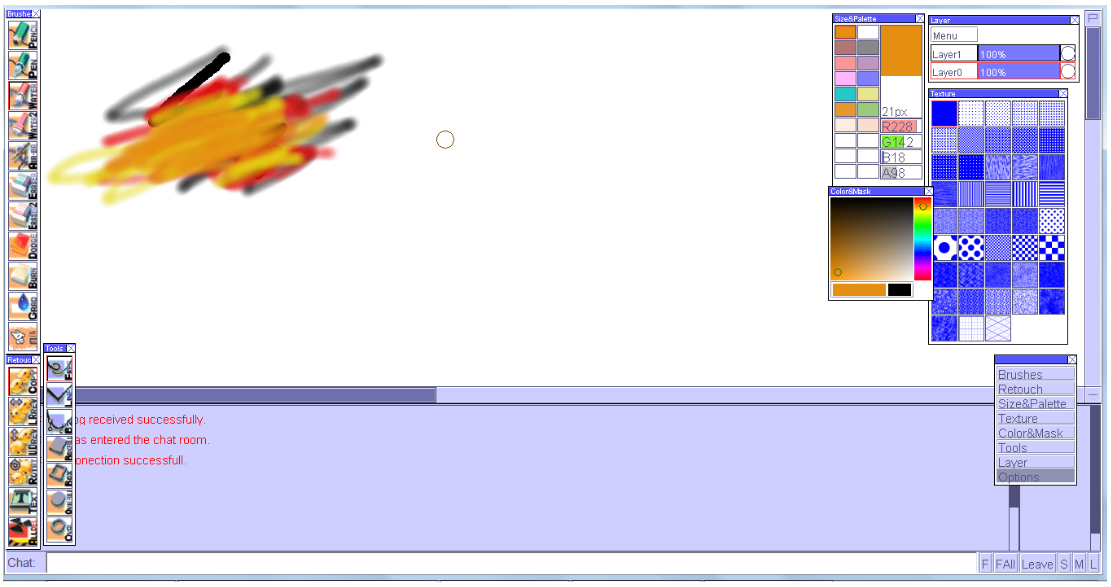
PaintChat>Log received successfully.
PaintChat>Pas entered the chat room.
PaintChat>Conection successfull.

Example image of Niko’s Paint Chat application under “standard” interface option.

5.6.1 Notable features in Paint Chat

Two Toolbar Interface Options

There are two types of interface the user can choose from before joining a room. The Standard will collapse all tools and functions into one toolbar leaving only the colour mask and layers control. The Professional toolbar will provide one toolbar that pops all other tools panels individually.


og received successfully.
as entered the chat room.
onection successfull.

Niko’s Paint Chat under “advance” interface option.

Tools

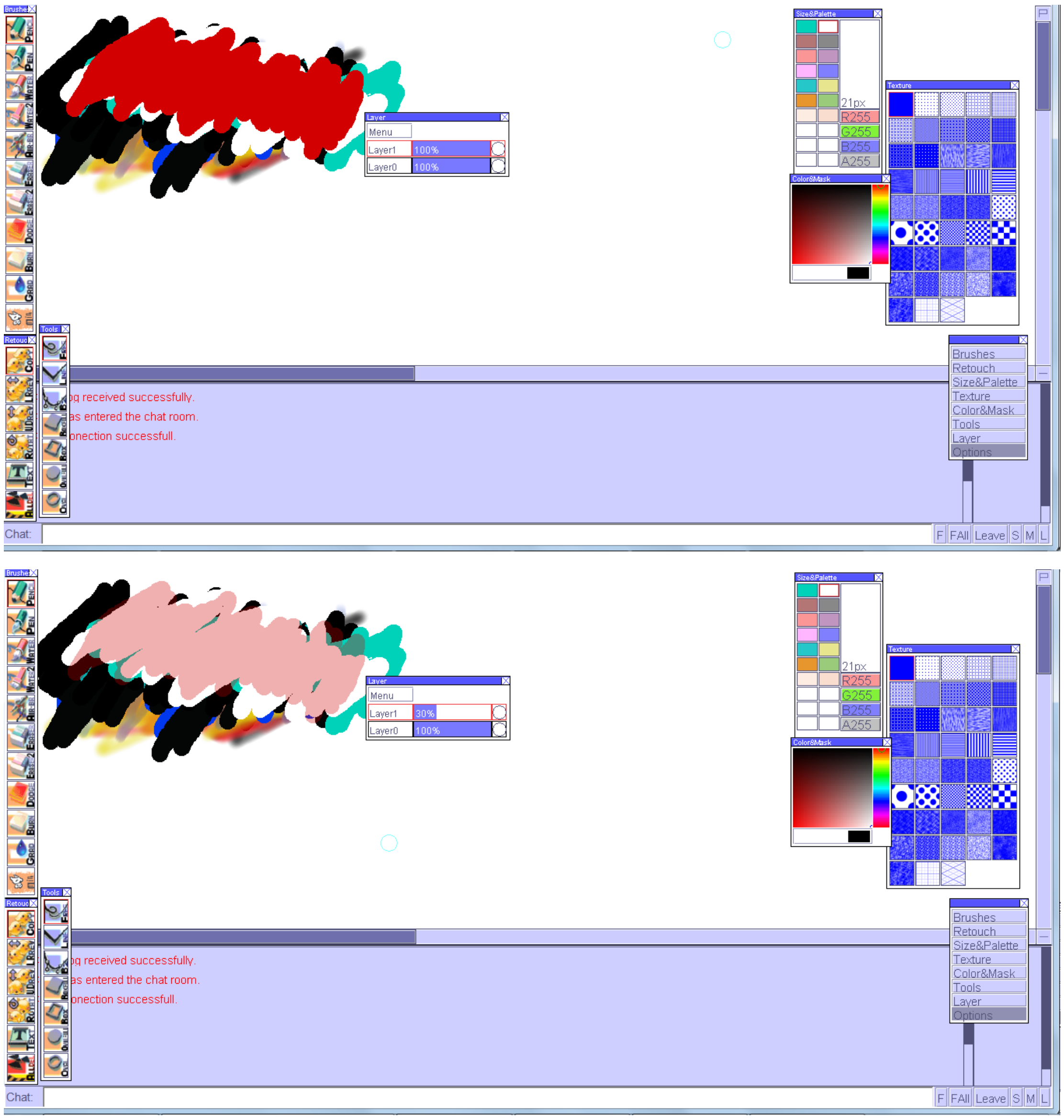
Niko’s Paint Chat has the greatest number of drawing tools available, compared to all other applications surveyed. There are 6 types of brushes, 4 types of erasers, dodge tool, burn tool, shapes, text tool, and many more. Brush tools have feely adjustable radius size and some “modes” similar to /f/lockdraw’s brush settings. Paint Chat has one of the most complex brushes in comparison to other collaborative drawing applications.

Large Canvas

Paint Chat have the largest canvas compared to all studied applications; 3400 x 1800 pixels in size. There are different level of zoom the user can choose. The user can initiate pan by dragging on one of the scroll bars at the edges.

Layers

The application provides two layers. One unique function provided is that layers have a transparency slider control. This allows the users to do overlay guides that other applications do not support.



g received successfully.
as entered the chat room.
nnection successfull.

g received successfully.
as entered the chat room.
nnection successfull.

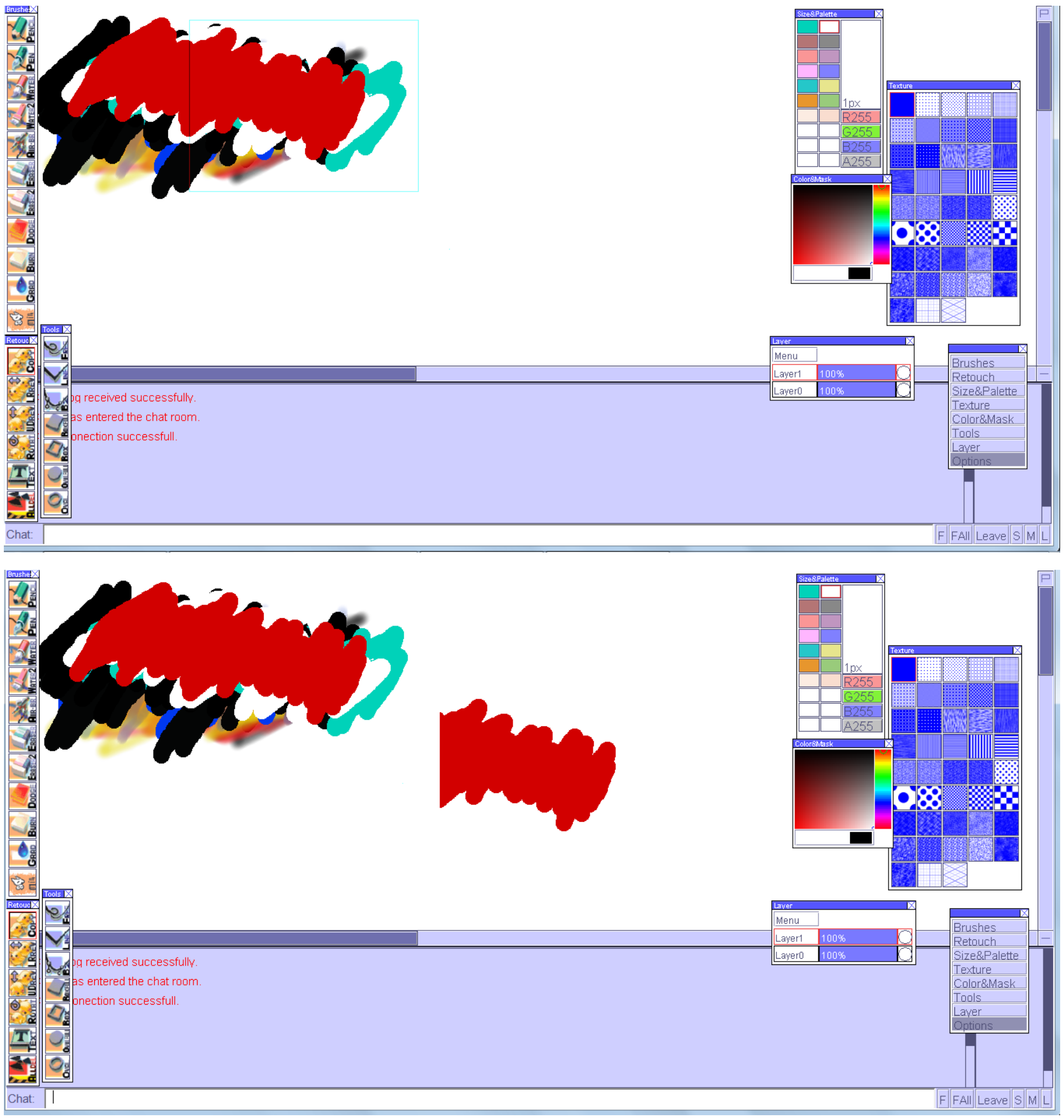
Example of layer alpha adjustment function in Paint Chat.

Floating Panel User Interface

All interface elements except for the chat system uses floating panel controls. Users can freely position panels around to suit their own preferences.

Copy Tool

Brushes, shapes, or text tools are not unusual tools in other collaborative drawing applications. But one tool that stood out in Paint Chat is the Copy Tool. This tool allows the user to drag a rectangle selection over a section of the drawing, then as soon as the user starts dragging the selected section, the area is copied and can be pasted elsewhere in the canvas. This can be a very useful tool in a collaborative environment. The idea and interaction for this tool have great potential, and should be considered in all collaborative drawing applications.


g received successfully.
as entered the chat room.
nnection successfull.

g received successfully.
as entered the chat room.
nnection successfull.

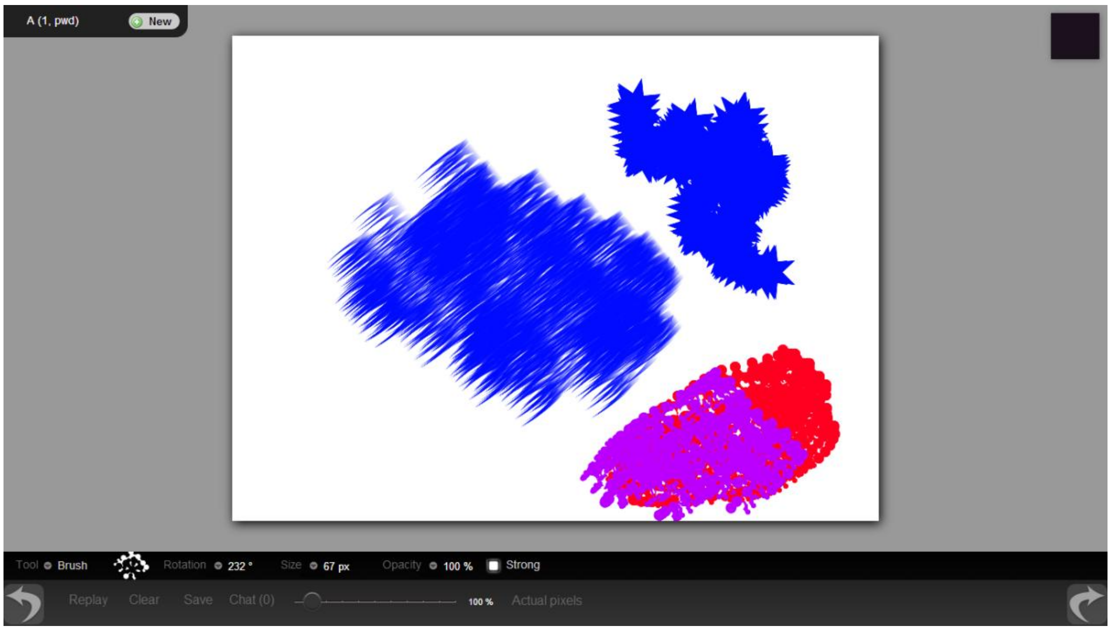
Example use of the Copy tool in Paint Chat; an area was defined, when the mouse drags it, the area is copied and pasted to where the mouse was released.

5.6.2 Notable Drawbacks of Paint Chat

While Niko's Paint Chat has the most functions and tools, it still has major issues that prevent it from over taking any other popular applications. Although it has over 29,000 registered users, room activity at the time of this project was very low compared to other more popular applications. The major issue with the drawing experience is the lack of hotkeys. All functions need to be controlled through the cursor. This really makes the interaction slow and tedious at times. Another major issue with Paint Chat is the lack of social oriented services such as a "Publish" function, video playback, or galleries. The lack of these functions gives a new visitor very little information about the applications and services; this makes it hard to gather interest in new visitors. The requirement of installing and activating a Java plug-in is another barrier of entry to the application. Another notable issue in Niko's Paint Chat is that the community heavily discourages spectators. The main reason is because their service is bandwidth capped. A user can be reported and banned for joining a room and not participating in drawing. Discouraging spectators can make the community less welcoming because everyone is forced to participate. This could be the reason why, at time of research, room activity and community activity was low.

5.7 Queeky's Multi Draw (Queeky, 2010)

Queeky is a website created in 2005, with the aim of providing a powerful single-user drawing tool that records the process of the artwork. All drawings submitted contains a video playback of how the artwork was composed. Queeky's single user paint tool is made in flash and, similar to Rate My Drawings, the tool provides many complex functions otherwise found on commercial drawing applications. In 2010, Queeky released a HTML5 collaborative drawing application. The application provides three tools: pen, brush, and eraser. The brush tool provides a good set of options such as different brush bitmaps and brush angle. However not all tools work in every browser; for example, only the Pen tool work in Firefox. The application has free zoom but no drag panning function; users need to work with scroll bars on the sides. The application does have record and playback functionality similar to other popular collaborative drawing applications. One notable feature Multi Draw provides is that it has a global Undo and Redo of 20 steps like Draw Chat. Another notable feature of Multi Draw is that users can choose custom canvas sizes at room creation. The maximum size is 1000x1000 pixels. However the lack of hotkeys make the view hard to manipulate and interaction slow. The drawing experience and features provided by Multi Draw are still lacking. Years have passed since the release of Multi Draw and no improvement or fixes have been made.

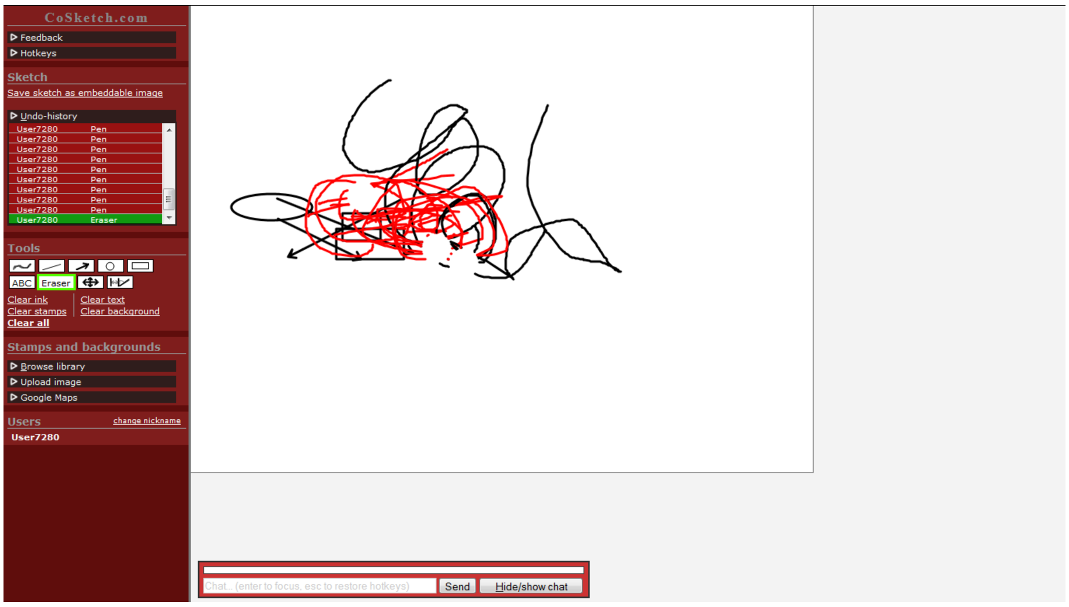

Tool Brush Rotation 232 ° Size 67 px Opacity 100 % Strong

Example image of Queeky's Multi Draw application.

5.8 CoSketch (CoSketch, 2014)

An HTML5 SVG based collaborative application. CoSketch displays the power of HTML5 SVG as a collaborative drawing application. CoSketch provides infinite steps of Undo. But this is also its major shortcoming. Once when enough strokes have been made, a rendering performance problem starts to show. The problem of performance becomes even more apparent when the eraser tool is used. CoSketch have additional features like uploading images, under-layering an instance of

Google Maps, and providing a set of simple stamps. It is clear that CoSketch is not targeted towards a drawing audience because it is not suitable for handling large number of lines. CoSketch however would make a good communication or interface prototyping tool.

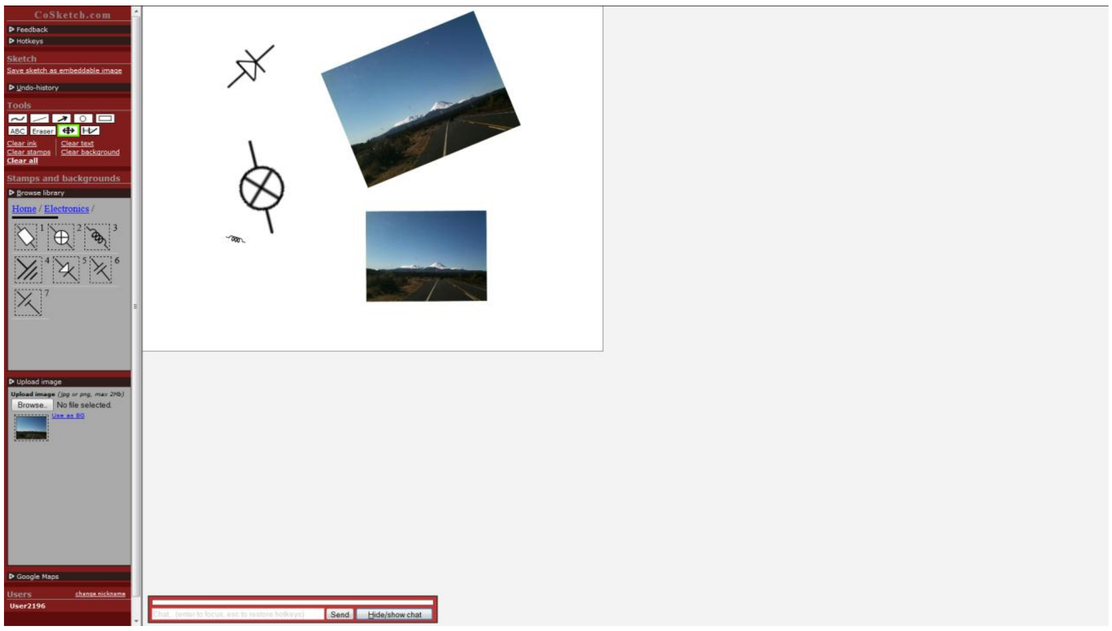


Example image of CoSketch application.

5.8.1 Notable features of CoSketch

Stamp System

Aside from providing some simple stamps, it allows users to upload images to use as stamps. This feature could be really useful in many situations. Group drawing practice often needs a reference to draw from, and the stamp feature could be used to show the referenced image. Collaborations may not want to start from a blank canvas, the stamp feature can allow users to upload an image as a guide or start. Stamping pictures may also be a conveniently quick and expressive way to communicate than text or drawing. This feature can also be combined with the idea from Niko's Paint Chat Crop tool where all cropped sections are remembered as stamps and can be reused. This feature can be a good addition to the drawing experience but is not really considered as a core drawing experience, so will only be considered as an addition in this project if there is time.



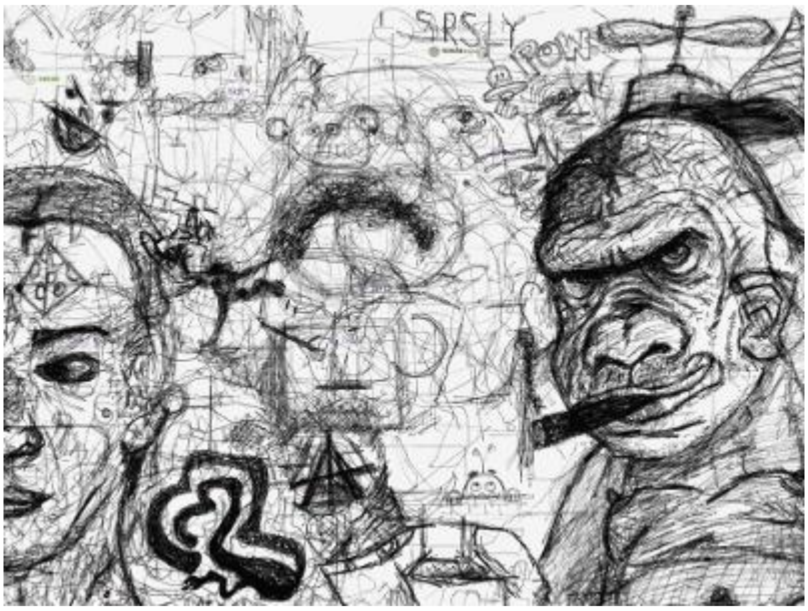
Example use of the Stamp system in CoSketch; users could upload images as stamps and modify them on the canvas.

5.9 Chrome Experiments (Google, 2014)

On announcement and support of HTML5 in Google Chrome, a website dedicated to showing off creative use of HTML5 with Google Chrome was launched. Over time many developer submitted experiments related to using HTML5 Canvas, drawing, and Websocket. Below are some notable experiments that relate to this project.

5.9.1 Multiuser Sketchpad (Mr.doob, 2010)

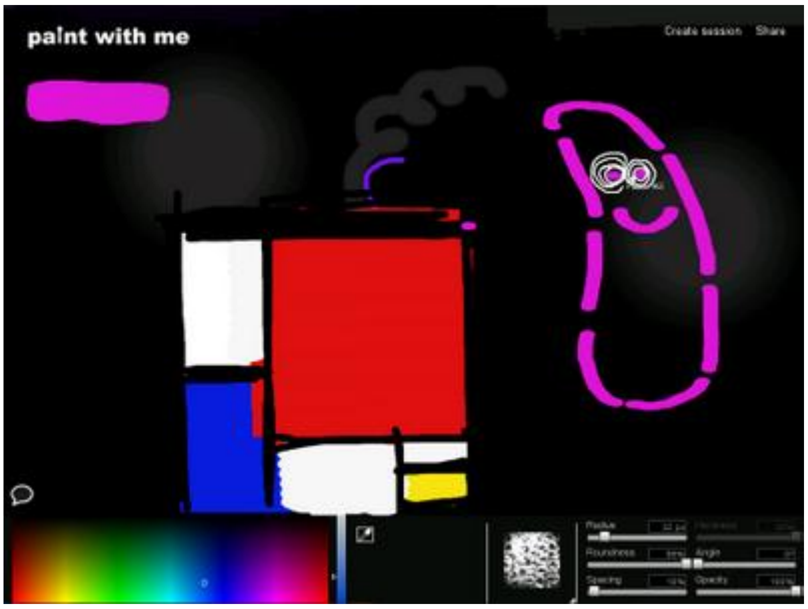
Released in 2010, this experiment demonstrated the capability of collaborative sketching with Canvas, Websocket, and Node.js. This application only provides a pen tool, a large canvas, pan function, and chat system. At the time of this project, the server for the application is down and only drawings from the client side are available.



Multiuser Sketchpad.

5.9.2 Paint With Me (Jones, 2011)

Released in 2011, this experiment is a collaborative drawing application with more features than Multiuser Sketchpad. Unfortunately at the time of this project(2013), the experiment is no longer available. The service has been completely taken down. From the screenshot image, the application contains various brush settings, colour, and a chat system. No further information regarding this application could be found.



Paint With Me.

Chapter 6: Requirements

After studying the existing online collaborative drawing applications, and with suggestions and understanding from the project artists. An idea of features and tools as a goal for this project were developed. The project artists identified the key components this project needed to rival the most popular applications. The researcher than decided what is most likely to be achievable in the given project time.

6.1 No Plug-ins Required

Most popular online collaborative drawing applications are made in Flash. As previously mentioned, this projects wants to reduce software dependency to only a modern web browser. There should be no issue of updating software or installation requirements to use this application.

6.2 Online Collaborative Drawing

This is the target experience of the project. All online collaborative drawing applications studied have the following features. Users need to be able to interact with each other in real-time in a shared drawing environment. As basic functions, users need to be able draw and view each other's lines as they are drawing. A basic chat system needs to be implemented, using Pixiv Chat as a good example. The chat system should be able to hide and only show pop-ups of new messages. Users must be able to connect and disconnect at any point in time and get the latest update of the state of the "room" when connected. Synchronization between users should be maintained while users are still connected to the room.

6.3 Undo and Redo

The biggest missing component on many web-based collaborative drawing application is the function of Undo and Redo. The project artist says that many people do not participate because of lack of undo. Fully Committing to every action is too high of pressure. Achieving Undo will be will a top focus of this project. Many drawing applications that have Undo function have limited steps of undo, this project may require such action when the Undo structure is confirmed. The project artists said limiting steps of undo is not likely a concern. The project artists claims that Undo is most often used as a quick fix to either most recent or accidental mistakes, if large number of Undo is required to fix one large mistake, people might as well cover or erase it over with another tool.

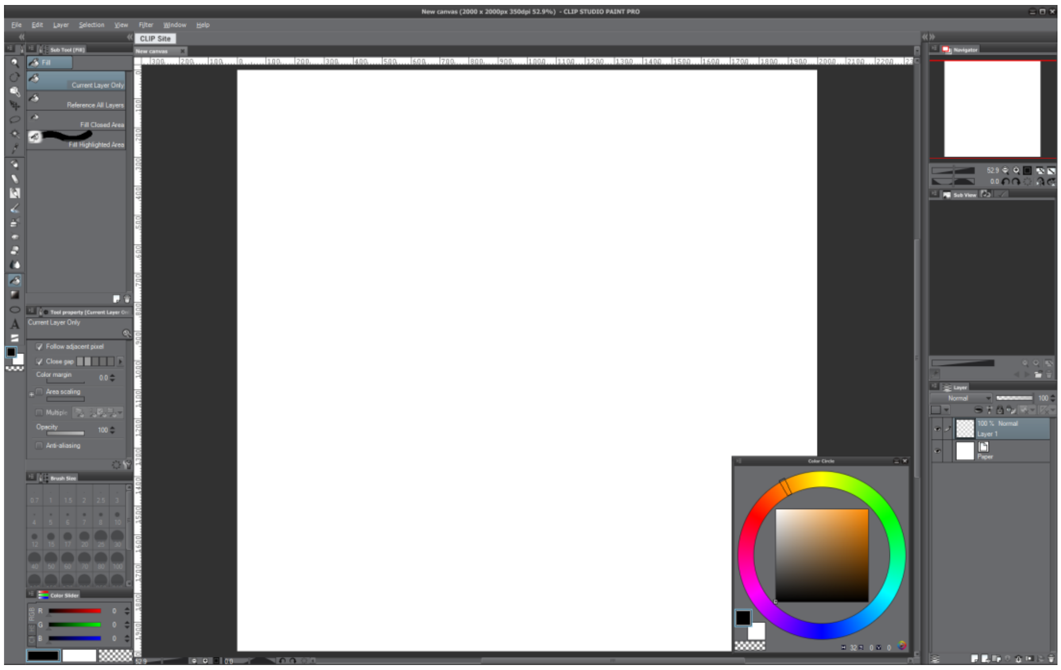
6.4 Layers

Although not many collaborative drawing application have layers, all the most popular ones do. This clearly shows a preference of layers feature to digital artists. This project will try to aim to achieve a structure capable of allowing layers. Although due to the time of the project, complex layer controls such as merge layers, colour adjustments, moving of layers, layer effects etc will most likely not be attempted. The project artist suggests that layers should just be global and three is enough. If layers could be freely created, people would just have layers dedicated to each user; this would remove the purpose of communication and collaboration. Setting the layers to global and at a set number will help reduce resource cost and should benefit the overall experience.

6.5 Basic Tools and Colour selection

Many online collaborative tools have a wide range of tools available. The project artists say that it is pointless to have lots of tools if the main interaction and options are weak. The lack of Undo just makes many tool even more dangerous and unappealing because they can easily destroy a painting for others. Much inspired from Pixiv Chat: a basic pen, eraser and colour sampler is enough as a starting point. Some necessary tool options are pen size, eraser size, and colour selection with transparency control. This project needs to focus on improving the main interactive experience of collaborative drawing rather than making more tools that people will just rarely use.

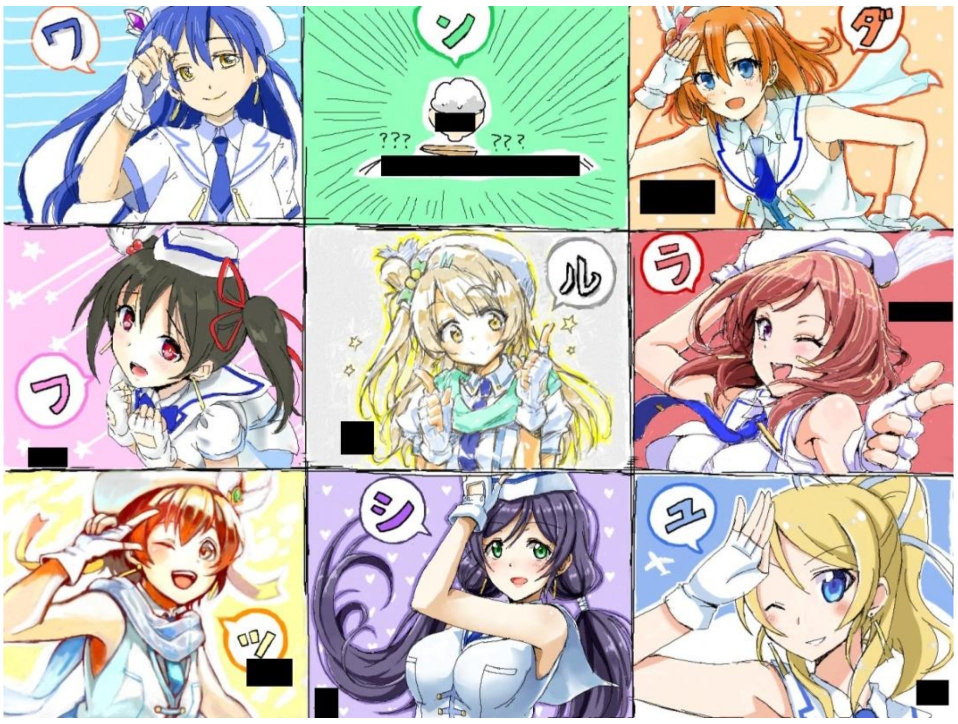
The project artists stresses that having a good colour selection interface and in a wheel format is crucial to the experience. No web-based collaborative drawing applications has a good colour wheel for their colour selection. The project artists a colour wheel makes a great difference in colour picking efficiency and user friendliness. The colour wheel exists in some stand-alone drawing software but still not as common. PhotoShop, one of the most popular and powerful drawing tool, only has the colour wheel tool as a custom plug-in created by other users. The project artist suggest the best example is a colour wheel like Clip Studio (CELSYS Inc, 2014), a Japanese drawing tool. However the colour wheel is still limited to picking general colours, another option to enter an exact colour is still needed. So as requirements two colour controls function will be required. A colour wheel for quick efficient colour picking and a colour selection interface for precise colour picking.



Example of Clip Studio colour wheel; Colour Hue ring with Saturation and Luminosity square.

6.6 Large canvas, Mini-map, and View transformation

One of the biggest reason for Pixiv Chat's success is the large canvas space it provides. The project artists says that social interaction in an online collaborative drawing environment is similar to interacting with people in real life; if users' relationships are not close, their draw-space should not overlap; it's like the idea of giving each other their personal space. Claiming drawing space is often a struggle for new users; most artists wants be humble to allow everyone to draw, yet, they still need a space to draw. This is why having a large canvas space in the application is important; because it allows and encourages more people to participate. Often as a solution to draw-space claiming, draw-spaces will be allocated out by the room owner or the first user when the session is started. This interaction can often be seen in popular applications like iScribble and Pixiv Chat.



Example of draw space grid in Pixiv Chat.

With a larger canvas, the need for functions to control view and keep awareness become more important. Pixiv Chat dealt with this matter by providing a thumbnail as a canvas overview, along with a pan function and zoom function. This project aims to achieve similar functions with HTML5 Canvas’s Transform functions but with greater freedom than most existing applications, such as free zoom. As an additional challenge, canvas rotation will be supported, this helps make canvas interaction similar to stand-alone drawing software and to drawing in real life.

6.7 Hotkeys

The project artist says one of the most important factors in making an application is how efficient one can make its use. This is also why the project artist feels that Pixiv Chat did well and became popular. The project artist suggests focus adding and improving hotkey experience on three major areas: view transformation, colour wheel, and main tools.

6.7.1 Hotkeys for view transformation

One hotkey for each view transformation; pan, rotate, zoom. The interaction required for the transformations are also just as important. Pan should work like all common drawing applications where when the hotkey is held down, the user can drag and pan the view with mouse. Rotate is more complicated: when the rotate key is held, if the user “grabs” and drags the screen, the view should rotate about the centre of the view depending on the amount of angle grabbed. It is important to rotate about the centre of the view. If the rotation is about the centre of the drawing canvas object, the further the point of rotation is to the view, the more likely the user will be dis-oriented as to where the view was focused and need to “find” where the user was originally looking. Zooming of view should work by, when hotkey is held, if the mouse drags away from the centre of the view, the view should zoom in and vice-versa. The metaphor for this interaction is like expanding the size of the paper by dragging it out. The idea of these suggested view interactions came from Clip Studio, drawing software that the project artists uses. The project artists feel they are the most suitable interaction for a drawing application.

6.7.2 Hotkey for a colour wheel

One hotkey for the colour wheel. This is basically the same as Pixiv Chat, where as long as the hotkey is held, the controls will float in front where the mouse was when the hotkey was pressed and allow colour to be changed. Once the hotkey is released the colour wheel will hide away.

6.7.3 Hotkey for major tools and setting

A hotkey assigned to each tool is common in most tool applications, but the project artist suggests a slight variation for interaction that was also available in Clip Studio. When the hotkey of a tool is held down and not released, the application will select and use that tool. However, once the hotkey is released, the application will switch back to the previous tool that was used. The purpose of this interaction is that the user can quickly use another tool such as the eraser, by just holding down one key and releasing it. Comparing to traditional hotkey conventions, where the user would be required to hit the hotkey of the eraser, use it, then hit the hotkey of the brush before continuing painting. This interaction would require twice the number of key presses and also learning and hitting two separate keys. The project artists feel that Clip Studio’s method of “quick hotkey” really improves the experience and efficiency of the interaction. This type of interaction is best applied to tools that are frequently used such as the Eraser tool and the Sampler tool. Aside from switching tools, the project artists pointed out that Clip Studio also has hotkeys for common tool settings such as brush size. When holding the hotkey for brush size, the radius can be dragged out by the mouse to make it bigger or smaller. This is another convenient interaction that reduces the number of times the mouse needs to move away from the point of work focus to the side menus.

6.8 Prevention of negative behaviour

With any social environment, the potential for people with unacceptable social behaviour will always exist. Although previously stated that social oriented functions will be a secondary scope of this project, problem of someone intentionally scribbling over other people’s work is too important to ignore. The account based log-in requirement from popular sites like Pixiv Chat and iScribble potentially reduces the risk of such behaviour. However as shown previously, people can still create dummy accounts specifically for the purpose of bad behaviour. Online video games and chat rooms often provide functions to restrain a user from performing certain tasks or remove and ban them from the room. These functions can be evoked from the room owner/administrator or democratically voted from other users. For this project, the most important task to prevent a bad user from performing irreversible task. Time is an important factor, because there is no point in removing a user if the damage is already done, so an administrative right system is preferred over voting systems. A quick “lock down” function is proposed. The effect is to disable everyone from interacting with the drawing but still allowing chat. This function basically suggest a structure that supports enabling and disabling of individual user interactions with the canvas. In conjunction with the undo system, the administrator should be able to lock down the room and then undo any users actions. This feature should reduce the risk of events like the a session the project artist experienced where a random user connected then intentionally scribbled over everyone’s work.

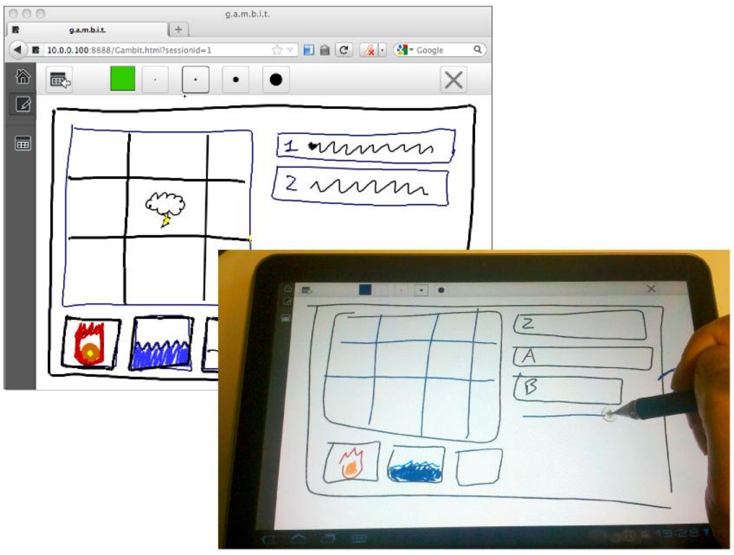
6.9 History playback and publish structure

Many online collaborative drawing sites have a “publish” option to show off work. But few have the feature to watch playback of the drawing process. This feature is really helpful for education, artist authenticity, and skill credibility. The project artists say it is difficult to focus on other users when the they are drawing something as well, so having a playback option gives them more opportunity to learn from another people. No online collaborative drawing applications record chat logs, most likely because of privacy risk in chat content. The project artists say that not everyone will find history playback important, some people are just there for a casual experience. Because features like publishing work and history playback are more about providing additional service on top of the drawing application, it is considered more social-oriented and falls outside the scope for this project. However these features should be considered if the project is to be taken to the mass public. Hence, even though these features are not within the scope of this project, the overall design of the structure should have the capability for these features to be implemented in the future. This means the history of all strokes and actions will need to be recorded in the server and the application will need to have an option to view the drawing canvas as a complete single image.

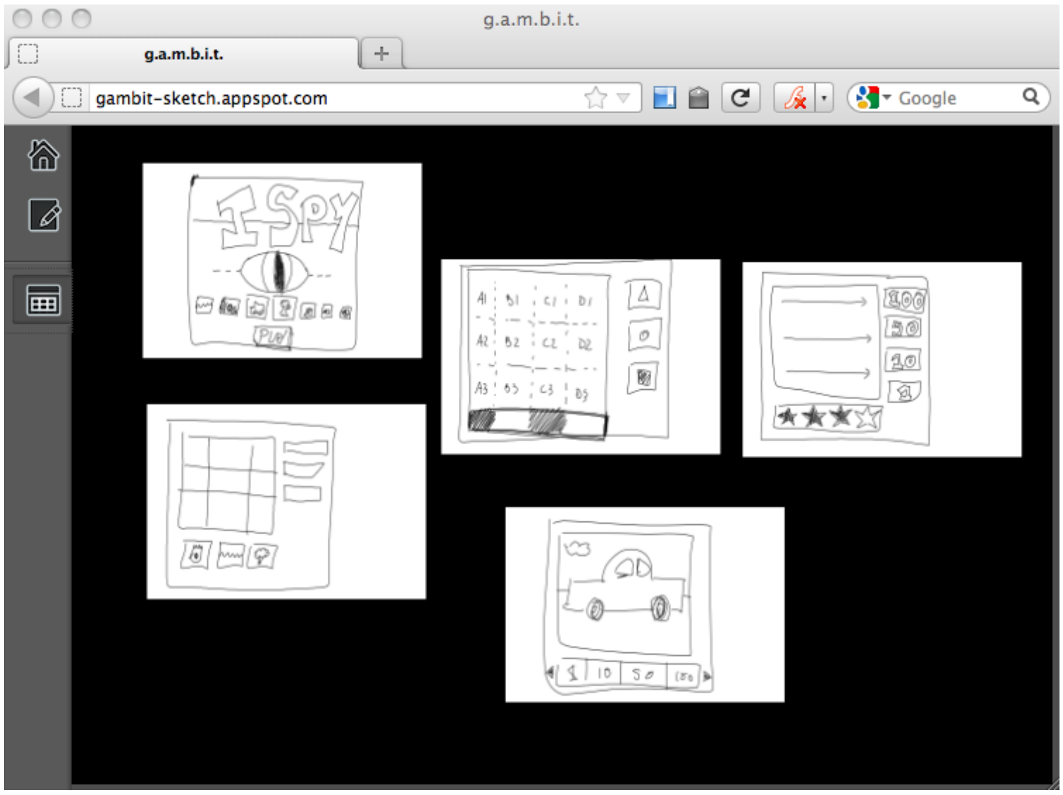
Chapter 7: Related Academic Researches

There is much related academic research that helped formulate decisions for the implementation of this project.

In 2012, GAMBIT (Sangiogi, Beuvens, & Vanderdonckt, 2012) proposed a solution to multi-platform collaborative sketching with HTML5. The focus of the project was to create a tool to improve collaborative user interface design through sketching. Because of this focus, the overall drawing experience is only limited to use of a pen/pencil tool with limited colour and few set stroke sizes. One notable structure in GAMBIT is that communication is established using AJAX but as previously shown AJAX is not ideal for collaborative drawing. The project built functions that focuses more on helping user interface design. One notable function is "Wall Sharing", where users can call back previously drawn pictures and organize them around in a shared space. However functions like these are more about quickly getting communication across and do not benefit the actual drawing experience. In collaborative drawings, getting ideas/communications across only account for the start, the majority of the process is still drawing.



GAMBIT.



GAMBIT's Wall Sharing feature.

Providing Undo/Redo facilities in a collaborative graphic editing environment has been attempted before; however all attempts assume running as stand-alone software. In 2001 the GRACE system (Chen & Sun, 2001) achieved undo using by keeping multiple versions or re-creating versions of objects. The underlying structure of GRACE is object(vector) based. The idea of hiding and re-showing of objects is a suitable consideration for this project. However keeping multiple versions of every single object would not be a suitable solution; this is because in drawing, all drawn objects will become irrelevant over time as more actions are made and remembering various versions of every object will use up unnecessary resources. This project will need to run inside a browser which limits performance and resources greatly. Re-rendering will also be a high consideration inside the browser, so eventually a rasterization phase will most likely be required to reduce the cost of managing a vector based structure.

In 2002, a bitmap(raster) based (Wang, Bu, & Chen, 2002) collaborative undo solution was proposed. Wang's system can also achieve selective undo. The undo function is achieved by calculating inverse functions of the action to be undone through calculating Boolean relationships with other actions. However Wang's method of undo is not suitable with this project because the proposed system assumed that the colours of each pixel in the bitmap could only be determined by one action as the “overlapped” result. Drawing software will have transparency in colours, this means the colour of a pixel could be determined by combinations of multiple actions. Under Wang’s proposed undo system with colour transparency, only the pixels influenced by the undoing action that are also not overlapped by another action will be removed; any other pixels modified by the undo action but also overlapped by another action will remain unchanged. This would make the final bitmap inaccurate to the expected undo effect because the presence of the “undone” action is not completely removed. With understanding of both object based and bitmap based undo systems, the most suitable approach is a hybrid system.

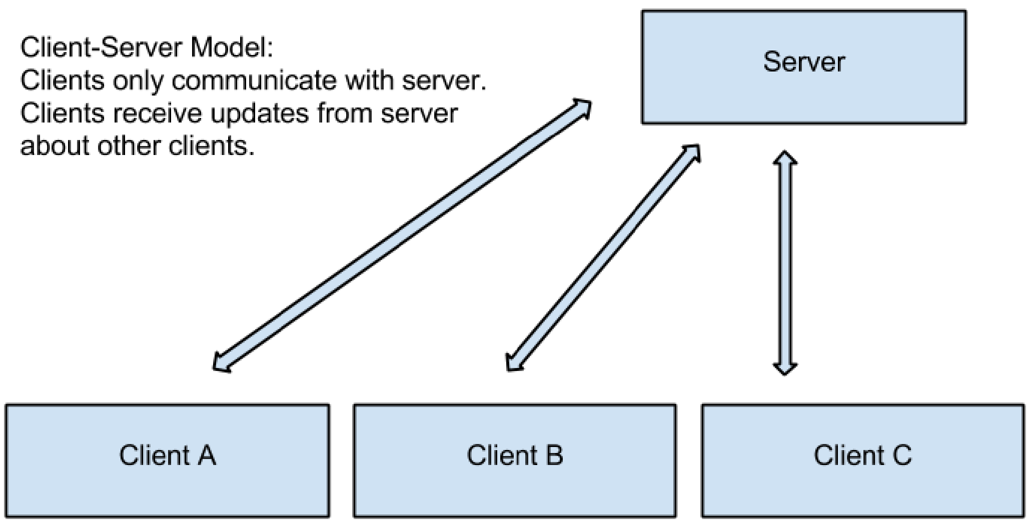
In 2012, a solution to collaborative WebGL visualisation through Websockets (Marion & Jomier, 2012) was proposed. This project focused on displaying a synchronized 3D scene over a Node.js Websocket server. The paper demonstrates achieving network synchronization with Websocket using a client-server model. The paper also gave an example benchmark test between Websocket and AJAX on their system; AJAX had an average latency of 332.4ms and Websocket had an average latency of 149.5ms; thus Websocket provided a higher synchronization rate for their system. Given the knowledge provided by this paper, it is suggested that Node.js and Websocket are suitable technologies for a real-time collaborative environment.

Chapter 8: Implementing Core Structure and Design

With the knowledge gained from studying existing applications and related academic researches, solutions to the requirements were attempted. This chapter describes the process of achieving the core networking and drawing structure of the prototype. The basic synchronization for collaborative drawing was first achieved, follow by attempt to support undo through a vector-raster hybrid system. Once undo was achieved, further modifications and implementations were added to support other tools. The networking was then improved by implementing prediction and having a server authoritative structure.

8.1 Basic drawing synchronization

In order to allow real-time interaction between multiple applications over a computer network, a real-time distributed computing architecture is required for achieving synchronization. One common use of this type of synchronisation is in multiplayer computer games; all game clients first initialise the same game "scene", then each client passes messages in real-time between each other to synchronize events and work towards displaying a common state in the game. As a result, this gives the illusion to players of co-existing in the same realm experiencing the same situation. This method of synchronization can be achieve in both peer-to-peer network model or client-server network model. For this project, the same method of synchronization is implemented to achieve a collaborative experience. This project uses a client-server model for synchronisation. There are many advantages of the client-server over peer-to-peer network that makes it more suitable for this project. One advantage is that all clients are free to connect and disconnect with little affect on other clients. In peer-to-peer, if the main host disconnects, other clients will be dropped or need to go through a re-hosting process. Another advantage of client-server models is that clients only need to communicate with the server, there is only one channel of communication. In peer-to-peer, every client will need to have a communication channel to other clients, this can drastically increase the number of message transfers and bottleneck a network if the number of connected clients becomes too great. One of the most important reason for choosing client-server model is because the order of global events are important. If one stroke happens before another, the result can be completely different; a peer-to-peer does not always guarantee the sequence of events because messages can be sent and receive at different times depending on individual client's network; a client-server model can guarantee sequence of events because clients all listen from the same source. Client-server model can guarantee synchronization, order of events, and reduce message transfer overall compare to a peer-to-peer model. The biggest disadvantage of client-server model is that because everything goes through the server, the amount of traffic and processing have much higher demand. This means the cost of server maintenance and traffic can be high and expensive. One method to reduce the server resource is by making the server dedicated, this means the server provide synchronization without running as a client itself. Dedicated servers have their own method of organizing and keeping track of states compare to client-side software. The server for this project is chosen to be dedicated. This means there is no server side HTML5 execution and the server has its own separate data organisation. The server mainly acts as message authenticator and message relay-er to clients.

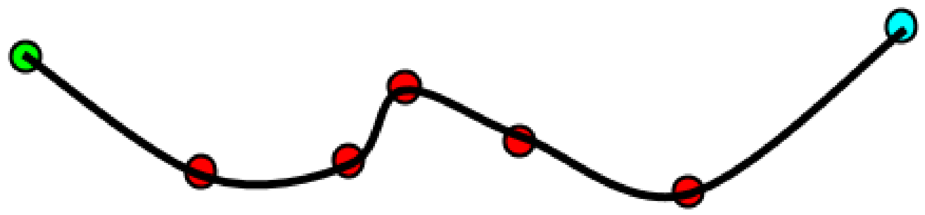

Client-Server Model:
Clients only communicate with server.
Clients receive updates from server about other clients.
Server
Client A
Client B
Client C

Example of Client-Server model.

As an important requirement, users need to be able to connect or disconnect at any point in time and still have the latest drawing state. This posed an requirement for the server to record and the client to perform past events in order to synchronize with connected clients. As an initial approach, same structure as used in some existing web-based collaborative drawing applications was used. All the communication packets were saved inside the server memory and then resent, in order, to new connecting clients. This implementation poses a problem of server memory overload if a session lasts for a long period of time. Many existing applications overcome this potential issue by giving each session a life-time duration or a button to clear its memory. After discussion with artists, it was concluded that resetting sessions due to memory limitation is not an important concern; this is because, according to their observations of these applications, it is not common for a person to participate for more than 24 hours. Thus this project ca achieve real-time synchronized interaction using a distributed computing structure with functionality that allows new clients to connect, synchronize or re-synchronize during a live session. For messaging exchange method, encoded strings are sent between the client Javascript front-end and the Node.js server. Security for the content of these messages is not currently important as interactions aims to be open between connected clients and are not intended to contain personal or sensitive information.

A drawing contains sequence of lines. Each line can be described as an action. Each action may require multiple events to define, update, and finalise. An action of a line contains the start point, end point, and points of the line segments in between. To synchronize an action of drawing a line across the network, three main events/messages are required to be sent through network: start drawing, update drawing, and end drawing. “Start Drawing” message notifies the starting co-ordinates of the line. “Update Drawing” message are the line segments points. To reduce the size of the data sent, “off-sets” from the previous point is used rather than the whole co-ordinate. When the user end the line, an “End Line” message is sent, which finalises the action.

Example of network messages procedure for a stroke.



🟢 Start Draw Packet: Sent on mouse down. Contains the x and y coordinates of the stroke.

🔴 Update Draw Packet: Sent while the mouse is moving. Contains the x and y offsets from the previous sent point.

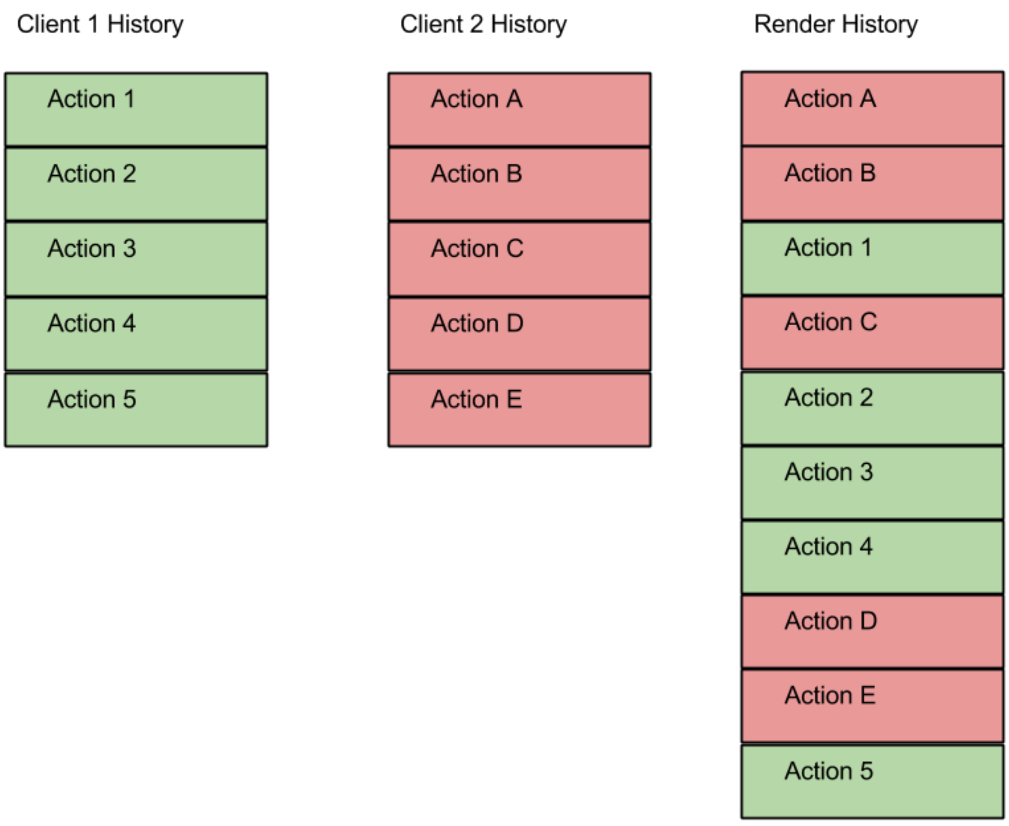
🔵 End Draw Packet: Sent on the mouse release. Contains the x and y offsets from the previous sent point and notifies the system the stroke has ended.

One stroke is stored and referred to as one Action.

Example of basic drawing through network synchronization of one stroke.

8.2 Undo and Redo

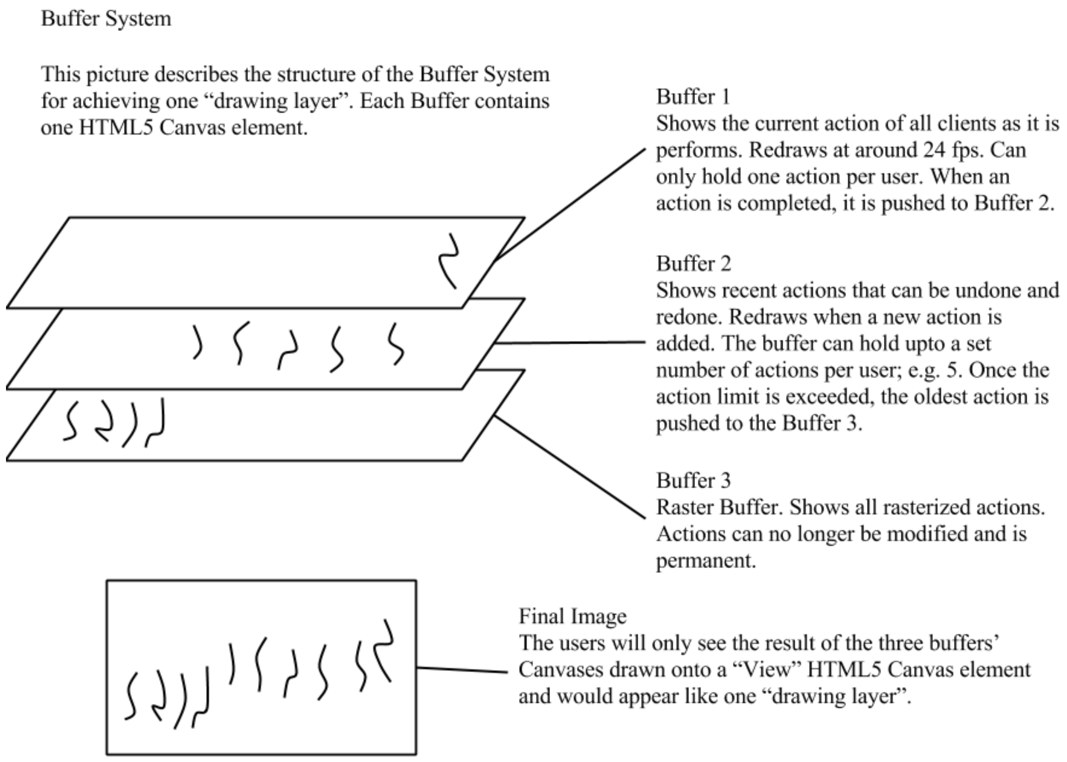
One of the most difficult challenge in this project was achieving Undo and Redo functionality. The idea of Undo and Redo requires the core structure of the application to keep a list of action history for each client and a re-rendering behaviour to display the correct state after on Undo/Redo on each client. To keep track of which client an action belongs to, start action messages need to contain a ID. Therefore a “New Client” message with its ID will be sent from the server at start of new connections to tell all other connected clients to initialise a new action history. Clients will also need to know their own ID, so the server will also send a "Your ID" message to the new client. Due to the collaborative nature of the project, a global list of action history is kept to ensure the correct action order of all actions. This global action history is then used as the source for each render call. The implementation of a render call is basically clearing the current Canvas and re-executing all actions to present the current state.



Example of this projects action history structure.

Frequent re-execution of actions can pose a serious performance problem especially due to Javascript and browser execution nature of a web application. If ten users each performs one action per second, the system would need to re-execute ten times per second all current and previous actions for previous users. What makes this performance problem even more apparent, as a standard among collaborative drawing applications, each client needs to see other clients' action live. This means the execution calls can be as frequent as per-packet(e.g. line segment) sent over network. The size of information inside one action is also an important factor; the larger the number of points in a line, the more process and rendering power will be used per execution. It is found that generally the biggest performance sinks are the pixel manipulation functions such as executing a stroke. Obviously enough, it is more efficient to draw one image of multiple strokes than to execute those strokes individually. If strokes could be all stored in a raster format, this would reduce the stroke execution calls to only once per action. However if all strokes are stored in an individual Canvases/rasters, each user will, either need to have multiple Canvas already dedicated to themselves, or the system will constantly creating new Canvas elements per stroke. Both above options are not suitable because, if large Canvases are to be supported in this project, each pre-constructed Canvases will need be the same size which will risk great amount of memory used to even support one user. The second method of creating Canvases only when a stroke is created will risk heavily affecting the performance of the browser because constant add and removal of html elements are not common and browsers may not be optimized to handle it. Also the same memory issue still exist if large area strokes exist. Another notable risk in storing strokes in raster is that if the stroke must be modified, the stroke will need to be re-executed either way. Storing actions in vector format provides the most flexibility but run the most risk of hitting a performance barrier. For web applications, trying achieve a balanced use of memory and processing is important. Considering that multiple Canvases will be used later for view Transformation, mini-map and other functions, having additional Canvases bind to individual users can easily risk memory overload. It is decided that storing actions in vector format and keeping Canvas elements shared is more suitable for this project.

With the above knowledge, a triple-buffer, vector-raster hybrid structure is implemented to help control performance. The final displayed image is composed from three separate HTML5 Canvas elements referred to in this project as Buffers. The three Buffers reduces the rate of draw calls and executing actions to only when necessary which is how it benefits overall performance. The first Buffer renders most frequently, it render depends on a timer function. The first Buffer is time-base to keep the system perform consistently without worrying the surges of action data packets. The tick time of the timer function is roughly 24 frames per second, a common rate for animations. This frame count could be modified later to suit performance need. The first Buffer only contains only the current action of all users. This Buffer allows users to see other users' action as they perform live because of the render rate. The second buffer is where all previously made actions are rendered. The second Buffer renders at end of every action. When the action on the first Buffer finishes, it gets pushed down to the second Buffer. The second Buffer is what allows undo and redo of actions. When an Undo is requested, the latest visible action will be flagged as *hidden*, and will be skipped on the next rendering call. Redo will un-flag the latest hidden action. When a new action has been pushed down from the first Buffer, and there are previously hidden actions, all the those hidden actions will be deleted. These behaviours matches common Undo/Redo functions. The last of the three Buffers is the *raster* Buffer; it appends old actions on to the Canvas Element and does not get re-drawn like the previous two Buffers. The third Buffer acts like most web-based drawings all actions are executed upon the Canvas once and cannot be modified anymore. The third Buffer avoid the performance penalty needed to render old actions but with the sacrifice of only allowing limited steps of undo. When it comes to final displaying of it all, an additional "View Canvas" is used as base, then the three Buffers Canvases are rendered on it in reverse order. The View Canvas is what the user will see. The size of the second Buffer is fairly flexible as it is dependent on the number of users and performance of browser, however it is recommended to have a fixed size so the experience for all users is more unified. Further experiments with larger group of users are required to find a reasonable size. The second Buffer action history size is set to five at the time of this project, hence five steps of Undo is supported.


Buffer System
This picture describes the structure of the Buffer System for achieving one “drawing layer”. Each Buffer contains one HTML5 Canvas element.
Buffer 1
Shows the current action of all clients as it is performs. Redraws at around 24 fps. Can only hold one action per user. When an action is completed, it is pushed to Buffer 2.
Buffer 2
Shows recent actions that can be undone and redone. Redraws when a new action is added. The buffer can hold upto a set number of actions per user; e.g. 5. Once the action limit is exceeded, the oldest action is pushed to the Buffer 3.
Buffer 3
Raster Buffer. Shows all rasterized actions. Actions can no longer be modified and is permanent.
Final Image
The users will only see the result of the three buffers’ Canvases drawn onto a “View” HTML5 Canvas element and would appear like one “drawing layer”.

Buffer system; allows limited steps of Undo and Redo.

8.3 Structure Modifications

An issue discussed with project artists was the relationship between the size and performance cost of each action. Each line segment sent will increase the number of stroke segments to be drawn. Because this variable can fluctuate greatly, this means one large action can affect the performance on all clients due to the increased number of line segments to draw. We concluded it is safest to put a “limit” on the size of actions to prevent one user ruining the experience for all. So a limit on the number of points in a line has been put in. Also, the project artist said most people do not often draw long lines; many artist draw/sketch by using many short lines, so this limit will not be a concern for most. The line segment cap at time of project is set to 200 points, the project artists said the current cap does not affect their drawing. The line segments(points) were originally sent upon the DOM Mouse Move Event inside the browser, however it was later found that due to different input hardware performance, the capped length of a drawn line can drastically differ. So a mouse with higher dpi and polling rate will trigger more segments per second than a mouse with lower spec. This can cause the experience for users to be inconsistent. To fix this issue, a setInterval() function was used to make the point sampling rate time based. This gives the line segment response a more consistent behaviour that was less hardware dependent. As an additional feature by making use of the multiple Buffers; if an Undo is requested while an action is still in progress, that current action will be “Cancelled” or removed from the first buffer instead. This allows an additional step of Undo as long as the action still exists inside the first buffer. The feature can be further expanded upon in the future like an action that can be previewed to all clients such as cropping or stamping images. With the above modifications and limitations, stable networked drawing of lines was achieved.

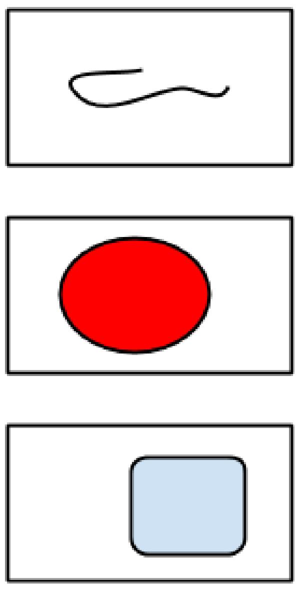
8.4 Eraser and Layers

Another important issue addressed was trying to provide an eraser tool that works with the three-Buffer structure and supports multiple layers. If the project only had one layer, the eraser effect can be imitated by using a stroke effect that paints with the background colour. However this project aims to achieve multiple layers, so the eraser needs to properly remove the pixels rather than covering pixels up. Removing pixels can be achieved using the “globalCompositionMode= ‘destination-out’” setting provided by Canvas which removes all intersecting pixels between Canvas and the next executed stroke. The main issue with the three-Buffer structure is that each Canvas inside the Buffers do not interact; actions naturally do not cross interact until they are inside the same Buffer. This caused a problem in the visual representation because the eraser effect will only be visible in the Buffer the action is in. The action history, undo behaviour, and the final rasterization is all still accurate because it will all eventually be applied onto one Canvas, but the separated rendering of other Buffers’ Canvas is not. To solve this issue, all actions that visually affect multiple Buffers like the erase action will need to be additionally previewed and applied on a separate temporary Canvas before being draw on the actual View Canvas. These actions are referred as “Cross Actions”, where they will re-apply on other Buffers’ Canvas as preview. Therefore when eraser action is made on the first Buffer, the rendering order is as follows: The third Buffer’s Canvas will render on at the bottom of a temporary Canvas, but before second Buffer’s Canvas is rendered on, all Cross Actions from the first and second Buffer will be “applied” on the temporary Canvas, then as the second Buffer’s Canvas is applied, the Cross Actions in the first Buffer is applied following by the first Buffer’s Canvas. This makes the rendering of eraser actions accurate but with some decrease in performance.

Below is an example showing the structure of rendering for one layer and including previewing Cross Actions.

Example of rendering of Buffer system for one layer with cross action

Assume the contents of each Buffer's Canvas element

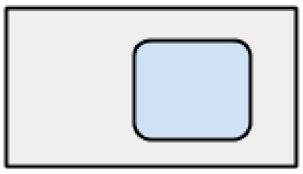


Buffer 1 contains an erase stroke. The stroke is displayed in black for visibility.

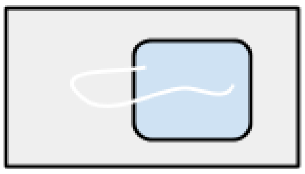
Buffer 2 contains some drawn shape.

Buffer 3 contains some drawn shape.

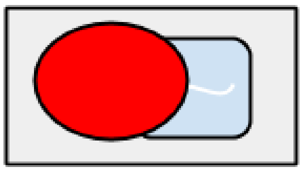
Sequence of rendering for one layer



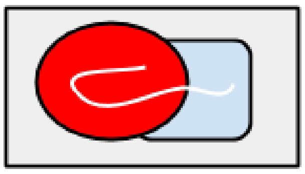
1. Buffer 3 is drawn onto the Final Image Canvas element.



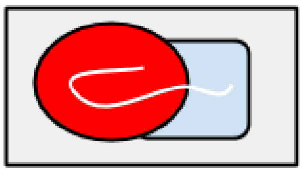
2. Any Cross Actions in Buffer 2 and Buffer 1 is retrieved and applied onto the Final Image Canvas element.



3. Buffer 2 is rendered onto the Final Image Canvas element.

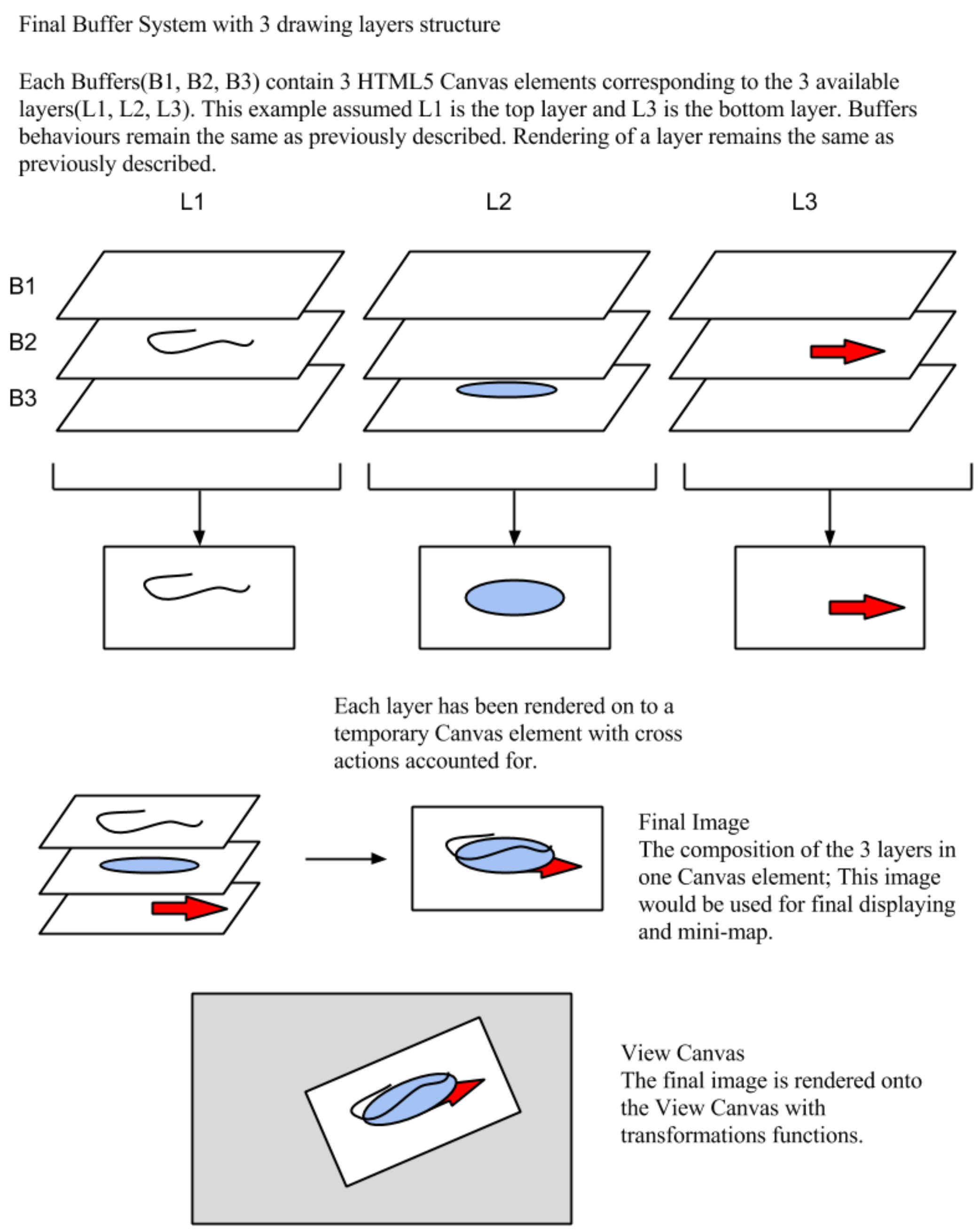


4. Any Cross Actions in Buffer 1 is retrieved and applied onto the Final Image Canvas element.



5. Buffer 1 is rendered onto the Final Image Canvas element. Because erase is a Cross Action, so nothing is applied.

As a requirement to allow complex digital drawings, a layer system is implemented. With the eraser tool working and displaying correctly a layer system can be implemented by multiplying the Canvases inside each Buffer and render the corresponding Canvases in correct order. An additional temporary Canvas is used to allow the mini-map and Show Image function to render in one drawImage() Call. As suggested, all three supported layers are global; users cannot create or remove layers but layers can be moved globally and hide or view locally. Each actions now have an additional “layer” information attached. The three Buffers now, instead of containing one hidden HTML Canvas per Buffer, contains three Canvases per Buffer. Each Canvas represents each layer and actions in the buffer are drawn to their corresponding layers when render function is called. Buffer can render actions of only one layer, or all layers depending on the update need. This structure means, when rendering to the View Canvas, the bottom-most layer Canvases in the three Buffers are rendered first then repeat upwards. By having this structure to achieve layers, the Undo behaviour is unchanged.



Final Buffer System with 3 drawing layers structure

Each Buffers(B1, B2, B3) contain 3 HTML5 Canvas elements corresponding to the 3 available layers(L1, L2, L3). This example assumed L1 is the top layer and L3 is the bottom layer. Buffers behaviours remain the same as previously described. Rendering of a layer remains the same as previously described.

Each layer has been rendered on to a temporary Canvas element with cross actions accounted for.

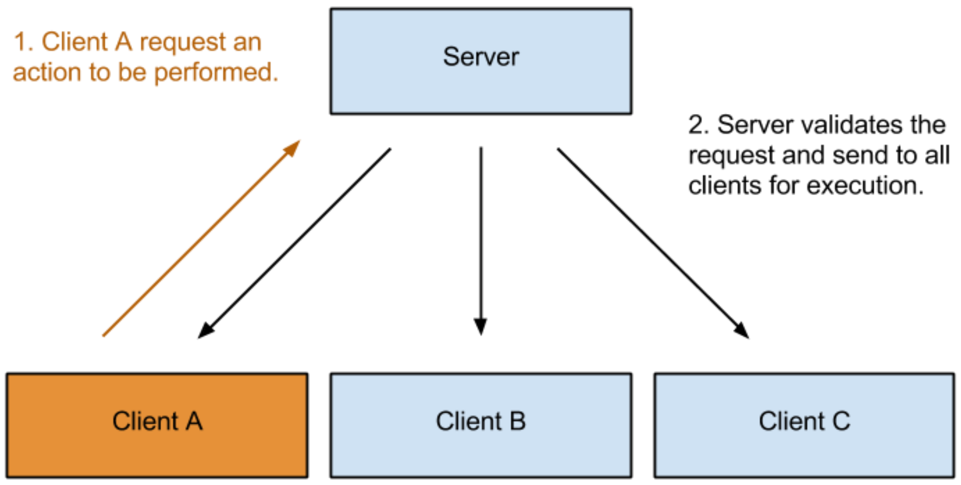
Final Image
The composition of the 3 layers in one Canvas element; This image would be used for final displaying and mini-map.

View Canvas
The final image is rendered onto the View Canvas with transformations functions.

The complete overall Canvas structure to allow undo, redo and layers.

8.5 Server authoritative structure

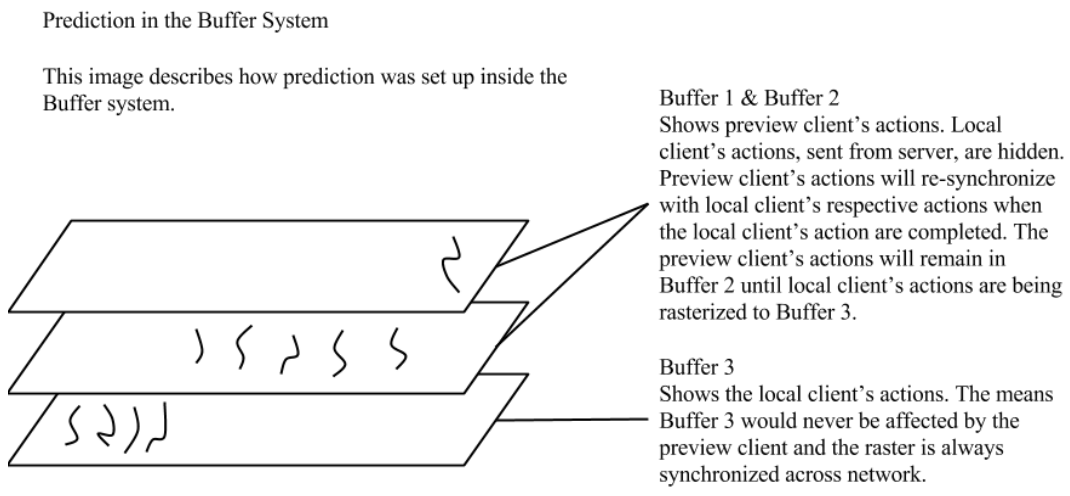
In many networked applications, server security and packet authenticity are important factors. Users always have potential to spoof bad information to attack the server. The server needs to protect itself from bad information that may break the system. If the server's data structure breaks or crashes, all clients will be affected. This is an issue in multiplayer games; where usually an authoritative client-server architecture is used to validate messages and determine on order of events. A drawing application has a similar need to validate and precisely define the order of events. In order to achieve this structure, all messages going through the server will need to be validated before being relayed out to all other clients to display. This also means all clients need to “request” permission of any action from the server before they are executed. In effect, this behaviour results in an obvious network latency that will be discussed later.


1. Client A request an action to be performed.
Server
2. Server validates the request and send to all clients for execution.
Client A
Client B
Client C

To achieve a server authoritative structure in this project, the server will validate all incoming requests and can modify the result freely before notifying all clients to execute. The server will check situations like; user can not request a “start drawing” action while a line is still being drawn, have the 200 line segments been used up, has the user run out of Undo steps…etc. Depending on the action, if its validation fails, the action could be disregarded or modified before relaying out; a “drawing” request can be modified to an “end drawing” request because the user has used too many points in a line. In this project, some users can also request actions to be performed on behalf of another client. This means the server will also need to check whether the requested action and the originating user have the right to do so. As an experimental feature, the server can also request and save Canvas images from a client and can send images to clients to replace their Buffer Canvas element images. This feature has been tried to see if possible to replace or reduce the sending of history logs on a new connection. This improvement can reduce the memory load on the server, reduce initialising time on clients, and remove the need to reset packet history on the server. However due to the instability in synchronization and time for development this feature was put on hold and not completed in this project.

8.6 Client Prediction

Any networked applications need to address the issue of network latency. Any information requires time to be packaged, sent, received, and processed before reaching to an actual display. Even local area networks will have a delay in communication. Among existing web-based collaborative drawing applications, there are still visual delays between mouse movement and a stroke appearing on screen. This delay is caused by the roundtrip time between client's request of action and server's feedback. Because such exchange time will always exist, client feedback are always latency dependent. As in online games, delays of response can negatively affect a user's experience with the application. In order to hide the latency, online games may use techniques often referred as "extrapolation" and "interpolation". Extrapolation is the idea of predicting and displaying results as the server request is made; this will make feedback to actions appear instant but in result desynchronize from the server state and all other clients. During this time, the client will continue to work on its own while the server receives and sends a reply of the true state. Interpolation is the idea where, once the server feedback is received, the application then gradually re-synchronizes its state to reflect the feedback while minimizing visual difference of two states. Example: If the local position of a moving object is different from the server's replied position, the object will try to interpolate towards the server's given position while trying to continue its original movement. Sometimes interpolation is not always preferable to an instant re-synchronization. Sometimes further extrapolation is made after receiving the feedback to account for the latency from server to client. For this project, a similar idea and structure is implemented to hide latency. Extrapolation is achieved by creating an additional local "preview client". The preview client will execute all requested actions as if there are no server. It has a reserved ID of 0, which the server will never use as an ID and all clients will use it as their own preview client. The preview client is set to “ignore” on the third Buffer. This means that the preview client will never rasterize when its actions get to the third Buffer and the third Buffer will always be synchronized with the true state. The real local client is then set to “ignore” rendering on the first and second Buffer. This makes the preview client the main visual display for the local client’s own actions. As a result, the user will see fast feedback from any action including modifying existing actions such as Undo. The technique of interpolation was not suitable for this project mainly because the process that would be required to access, check, and re-write on every incoming packet would be too complex and might impact performance. The rate of incoming messages is high but the processing ability of the system is limited. Instead of interpolated synchronization, a instant overwritten synchronization is used. This is achieved by using an action ID tag attached to every action. The action ID is generated by the local client and sent with the “Start Action” request to the server. The server remembers this ID and then replies with it to acknowledge or modify an action. So when an “end action” for the local client message is received, the application will over-write or change any information that is different from the preview client’s action to update to the server’s response. The actions in the preview client will only be removed when the actual host actions in the local client are rasterized to the third layer.


Prediction in the Buffer System
This image describes how prediction was set up inside the Buffer system.
Buffer 1 & Buffer 2
Shows preview client’s actions. Local client’s actions, sent from server, are hidden. Preview client’s actions will re-synchronize with local client’s respective actions when the local client’s action are completed. The preview client’s actions will remain in Buffer 2 until local client’s actions are being rasterized to Buffer 3.
Buffer 3
Shows the local client’s actions. The means Buffer 3 would never be affected by the preview client and the raster is always synchronized across network.

Structure to hide latency in the Buffer system.

8.7 Final Technical Structure

With all of above implementations combined, the core collaborative drawing mechanism is completed. It has the ability to undo, redo, hide latency, is server authoritative, all while maintaining synchronization.

After all the core mechanics combined the server to client initialisation handshake is as follows: When a new client fully loads the web page, an automatic request to make a Websocket connection with the server is made. When successful, the server will first send all the existing history stored and will continue sending any other messages currently being exchanged with other clients after it to maintain synchronization. If this client is the first client, the user will also be set as the administrator and a “Set Admin” message will also be broadcast. Once all the history is sent, the server will broadcast to all clients an “initialise new user” message along with its unique ID number. This message allows all the clients to create space and structure for the new client, ready to interact. The server will also send a direct “You ID” message to the new client. Note this message is direct and is not saved in the server history because its only sent to one client. “You ID” message tells the new client to bind the reserved preview client to this ID, allowing the prediction to work as intended.

The system synchronization process for a general action on the client side is as follows: The user initiates a “Start Action” request containing the user ID, action ID, tool, layer and any additional info like cursor position, pen size…etc. Messages in this project are sent using strings, so actions are identified using the starting character e.g. “s,3,2…” means to start drawing a line, for user 3, and action ID is 2. The size of the message could be further compressed in the future with encoded strings but the current implementation is easier to debug. The user ID is to help identify which client the action is to be applied on. The action ID is a local initiated ID from the originating client that help with synchronization later when server replies. As soon as the request is made, the preview client will assume the server validation will pass and execute the “Start Action” as the preview client. If the server validation passes, the message is relayed out to all clients which is when the action is executed on other clients. As previously explained, the host client's true actions will be hidden by the preview client until the third Buffer. If the action has an "in progress" phase like drawing a line, offset line segments are also requested at a set intervals. These requests are referred as "Action Updates", which notify alteration of the currently executing action. Once the user ends the action or the server decides to end the action, an "End Action" message is requested or received. If the user initiates the "End Action", the preview client will assume the validation passes and push the action down to the second Buffer. When a client receives any "End Action" message from the server and the client ID is its own, the Buffer will try to find the preview client's action inside the first Buffer or the second Buffer using the action ID as reference. Once this action is found, the data of the action ended by the server will overwrite the preview client's action's data. This means when it comes to displaying the preview client's action, it will be completely synchronized with the server action. If the preview action is found inside the first Buffer, this means that the preview client's action is still in progress and the server has force ended the action. The completed server reply action is then pushed down from the first Buffer to the second like all actions. When a client's action is about to be rasterized to the third Buffer and the preview client is bind to the same ID, the preview client action is then removed.

Chapter 9: Client Side and User Interface

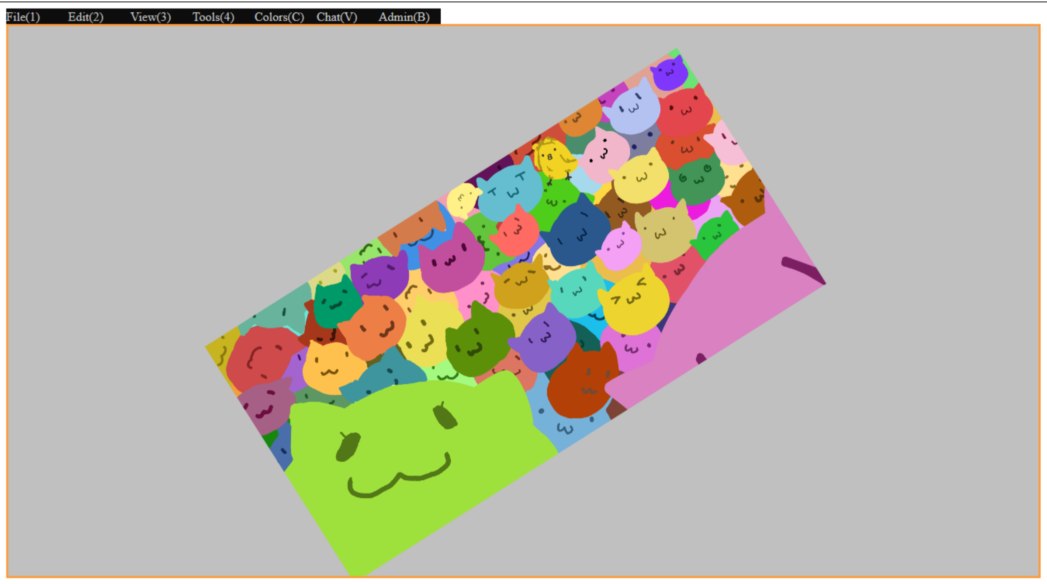
The front-end development focused on incremental development from project artists’ feedback. Almost every function addressing towards the requirements would have comments stated from project artists testing it. The process started with getting the transformed canvas displaying correctly, then implemented a suitable general interface system. Implementations addressing the requirements are then attempted.

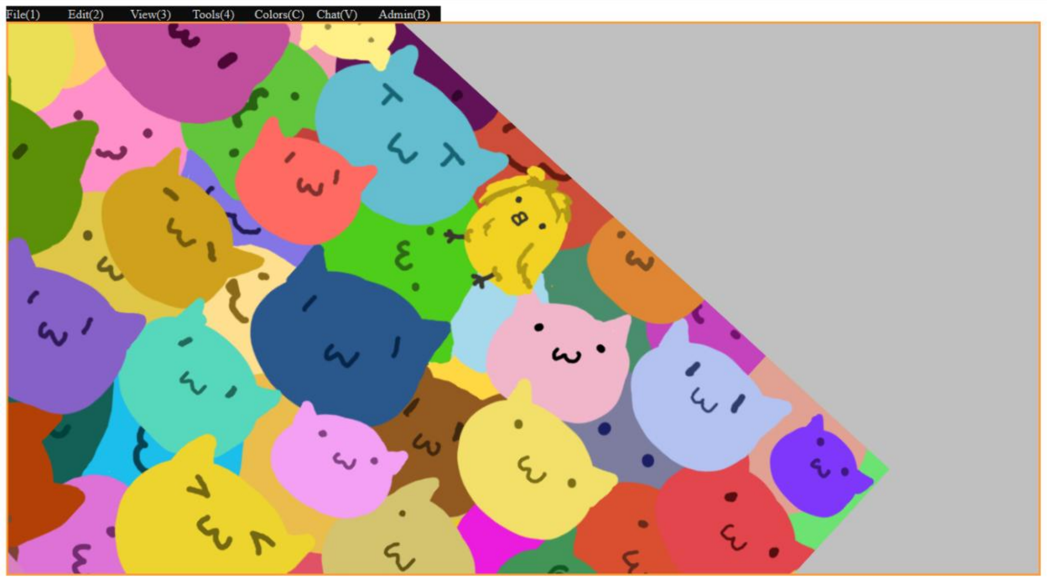
9.1 Overall structure

As previously stated, the client front-end interaction and the client back-end is written in Javascript that runs through web browsers. The overall structure of the front-end is similar to a computer game; the visual updates are time based, independent of backend or the server. Line segments can appear at a rate of hundreds per second, it is important to use time based rendering so the visuals can perform at a consistent rate no matter if the back-end or server is under heavy stress. As previously mentioned, the only Canvas element that the user sees is composed from multiple hidden Canvas elements. This Canvas element is referred as the View Canvas. It acts like a camera view to the main drawing. Because the main drawing canvas is separated, this allows graphical user interface or additional aids to be drawn on without affecting the real Canvas that users draw on. By making use of Canvas element’s transformation functions, the real Canvas can be Rotated, Translated, and Scaled. This gives the effect of rotating, panning and zooming on the view. Note the View Canvas is local, so transformations and drawn graphical aids do not affect other clients. The HTML5 Canvas transformation functions works like a “setting” on the Canvas, once the values are defined, everything drawn on will have the transformation applied. Even the co-ordinate system does not change. There are individual functions of Rotate, Translate, and Scale; or all transformations can be set with one

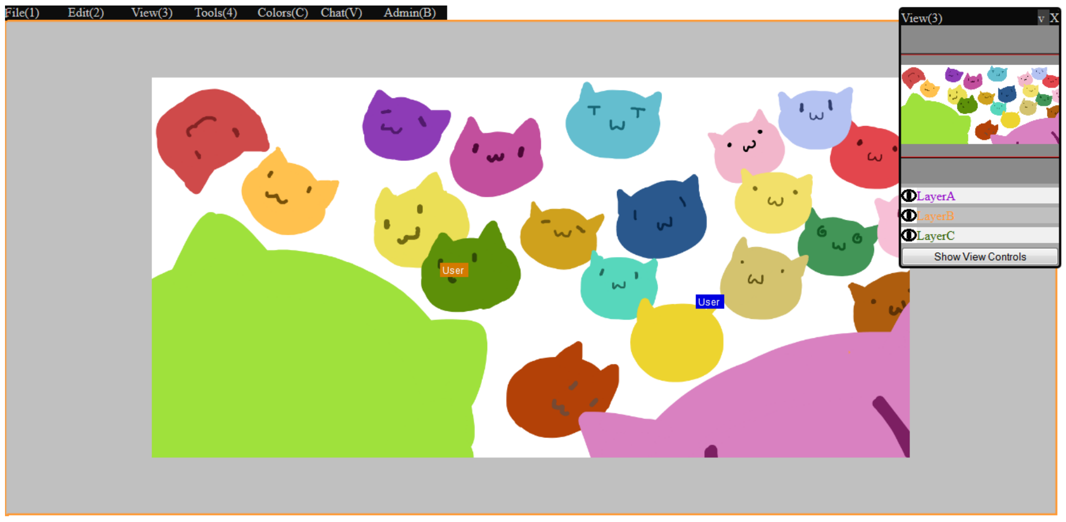
SetTransformations function. Because this project needs to achieve transformation based on the view centre which requires transforming from various origins. Individual transformation functions are therefore used in preference to SetTransformation. The View Canvas transformation for this project is as follows: Translate the draw space to half of the view area width and height, this will allow the rotation function to rotate the draw space about the centre of the view. Rotate to the user set angle value. Scale the draw space by the user set zoom value. Translate the draw space to the user set X and Y value. Finally the real Final Display Canvas is drawn on. The newly set transformations will take effect and display it correspondingly. As explained previously, the Final Display Canvas is composed from the multiple Canvases inside each Buffer. Once the Final Display Canvas is drawn, the transformations are reset to default so all other GUI elements can be drawn without being transformed.

One GUI requirement of this project is to see which user is currently performing which action. This is achieved by drawing name tags on the View Canvas above each users' current Action.



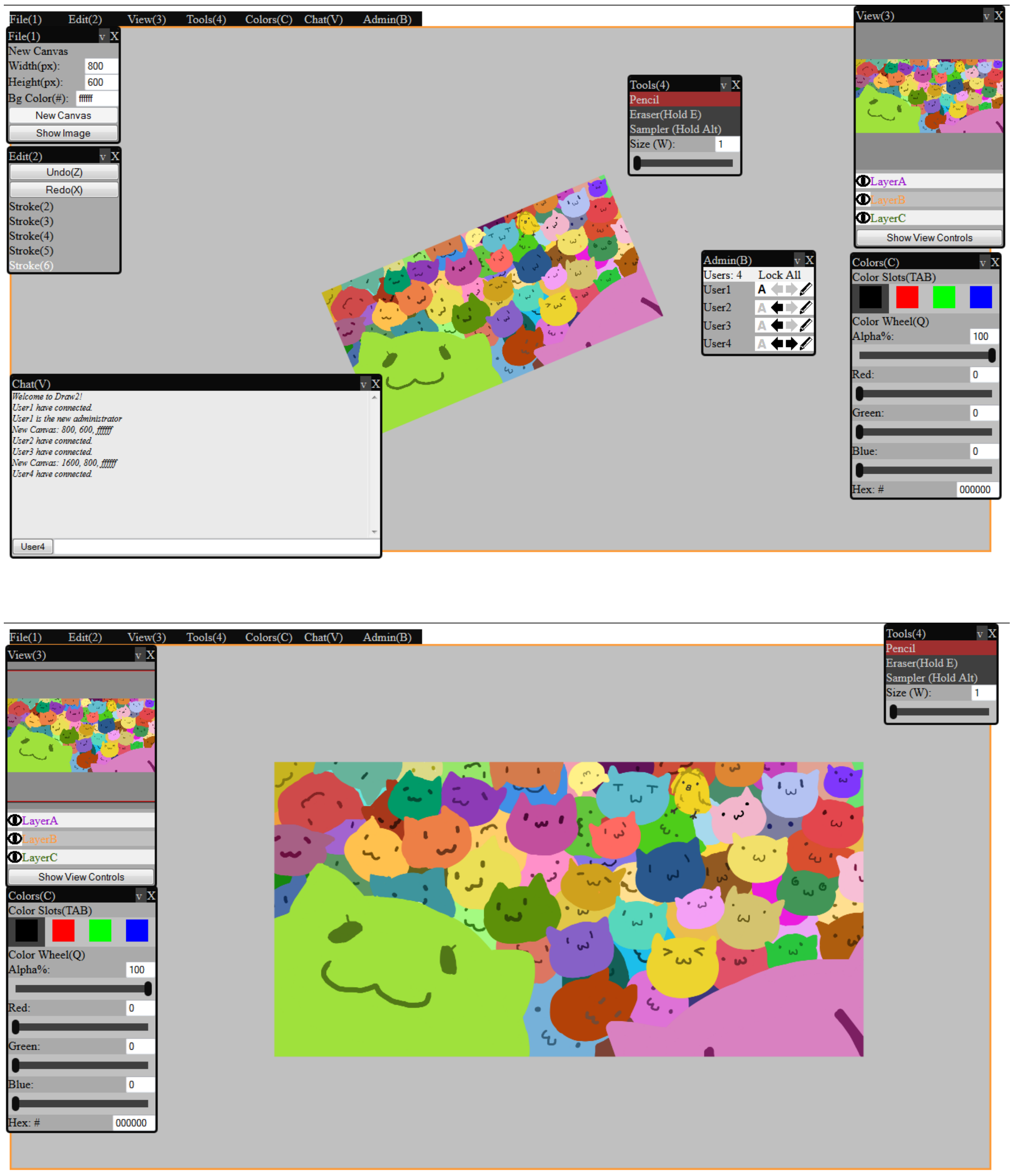


A freely transformable canvas; able to Pan, Rotate, and Zoom.



User’s name tag is shown when they are performing a action.

The project artist observed that the appearance of the application will heavily affect the attitudes of its users; the more casual or child-like the application seems, the less likely people will be to put serious effort in using it. For overall user interface design, a floating panel system was implemented. Floating controls are a common approach to many tool applications. Controls are separated from the main interactive space and can be repositioned freely. The biggest strength of this system is that users can create a personalized work environment for themselves by showing, hiding and positioning controls of their own choice. This approach to the interface gives the application a more professional sense and freedom in interaction style. The panels system are created by manipulating HTML Div elements and their CSS specification through Javascript. Panels can be shown, hidden, pinned, and freely positioned around inside the browser view. A menu bar is placed along the top of the web page to control the display state of each panel. When each menu button is clicked, their respective panel will show or hide depending on the display state. If a panel is set to show, it will show below its menu button, or display at the pinned position. The interface and menus are intentionally setup with controls categorised similar to a stand-alone applications drop-down menus such as File, Edit, View…etc.



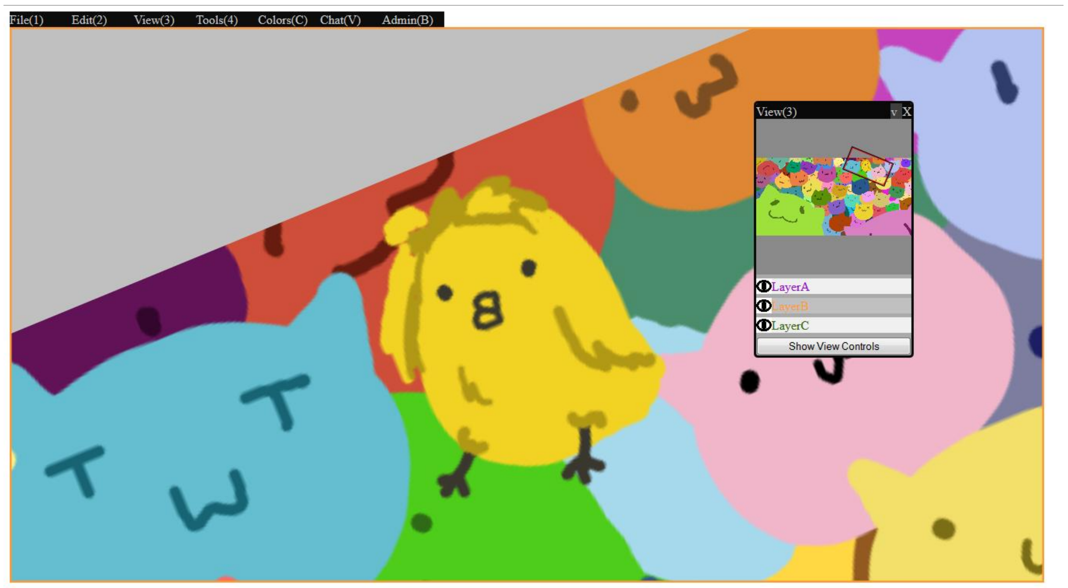
Width(px): 800
Height(px): 600
Bg Color(#): ffffff
Size (W): 1
Users: 4
User1
User2
User3
User4
Alpha%: 100
Red: 0
Green: 0
Blue: 0
Hex: # 000000

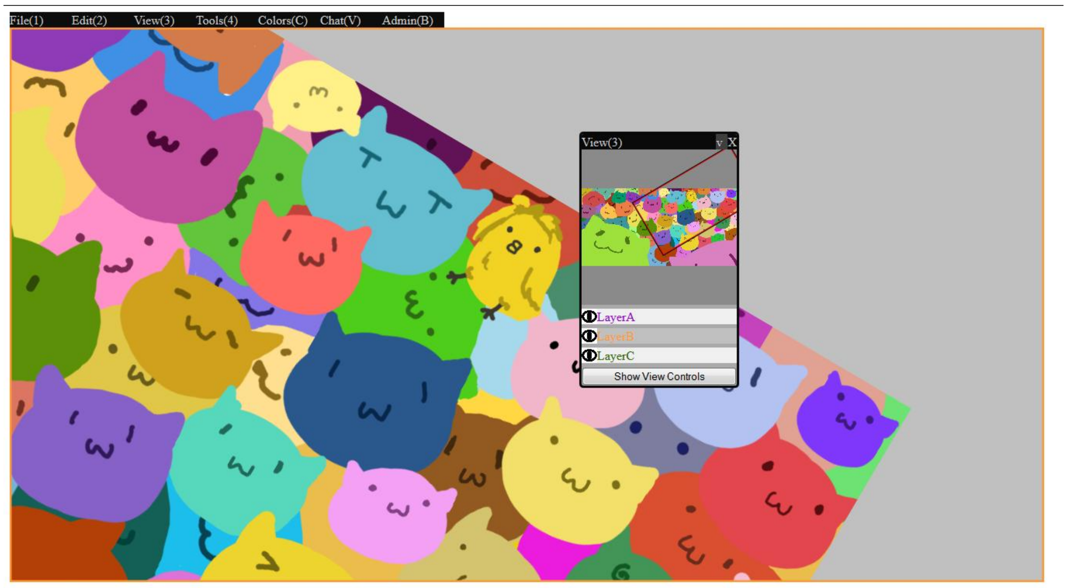
Welcome to Draw2!
User1 have connected.
User1 is the new administrator
New Canvas: 800, 600, ffffff
User2 have connected.
User3 have connected.
New Canvas: 1600, 800, ffffff
User4 have connected.

Panels allows custom interface layout.

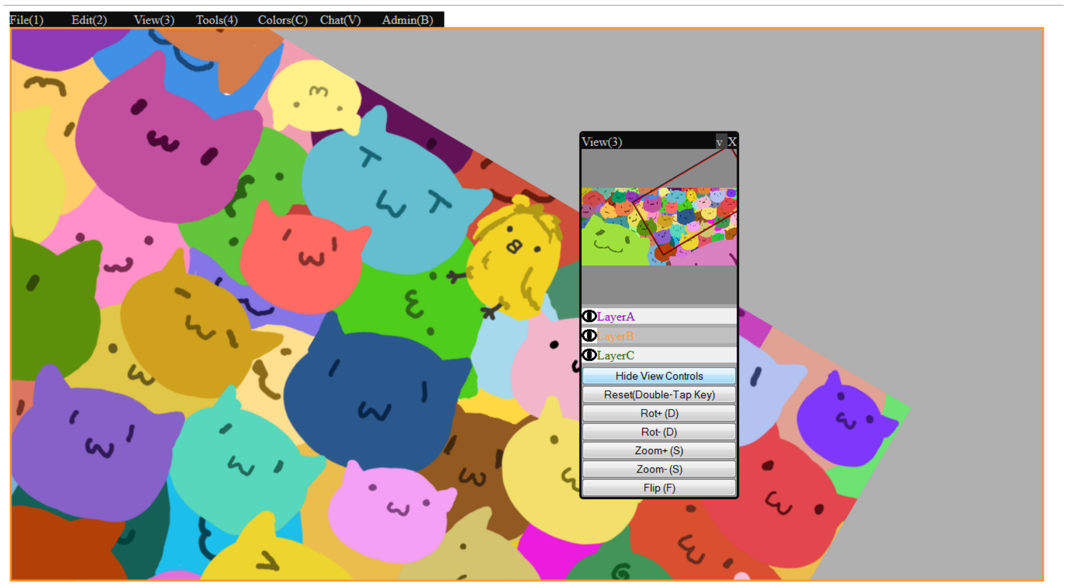
9.2 Mini-map and Layers UI

As a requirement to match popular online drawing applications, a mini-map is implemented. The mini-map is basically achieved by drawing another copy of the Final Display Canvas on a smaller Canvas element. As an expected behaviour of mini-maps, the size of the mini-map should be consistent regardless of size or ratio of the drawing paper. This means the Final Display Canvas will need to be stretch to the edges of the mini-map while maintaining its own display aspect ratio. To help accommodate all ratios, the mini-map size is set to a square of 200px. To achieve the correct display a series of calculations are needed. The system needs to first determine which of the height or the width of the drawing canvas is the greater and dominant. This dominant value will be used to determine a scale for the Final Display Canvas inside the mini-map's Canvas element. The dominant axis will be stretch to completely fill one dimension of the mini-map and the other axis will be padded and centred inside the mini-map. A rectangular visual guide of the current view is added on top of the mini-map. The rectangle is achieved by reversing all transformations and incorporated scale ratios. The mini-map is interactive; if a user clicks or drags mouse on a position inside the mini-map, the pan position will be moved to that position immediately.



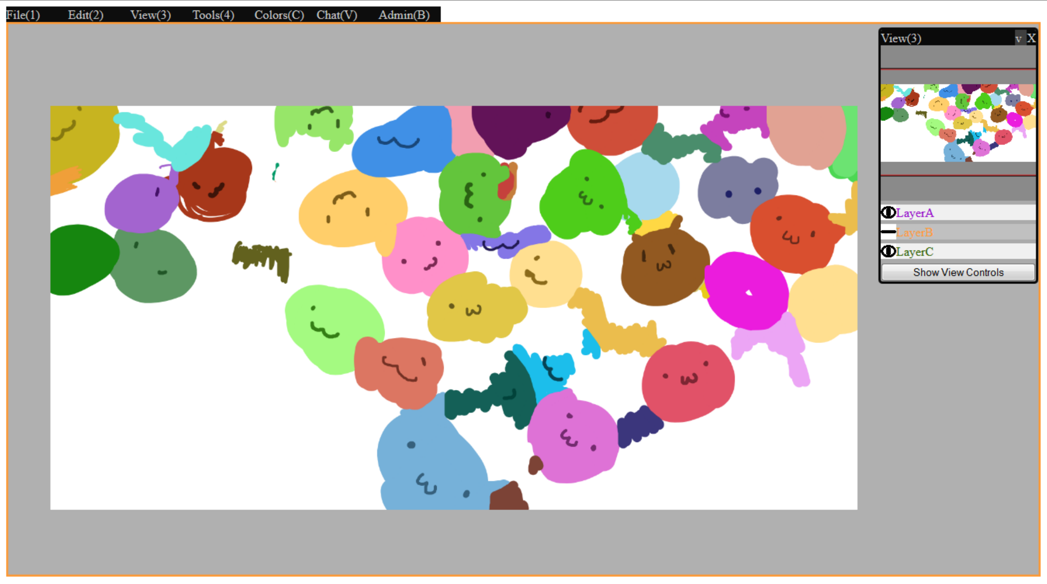


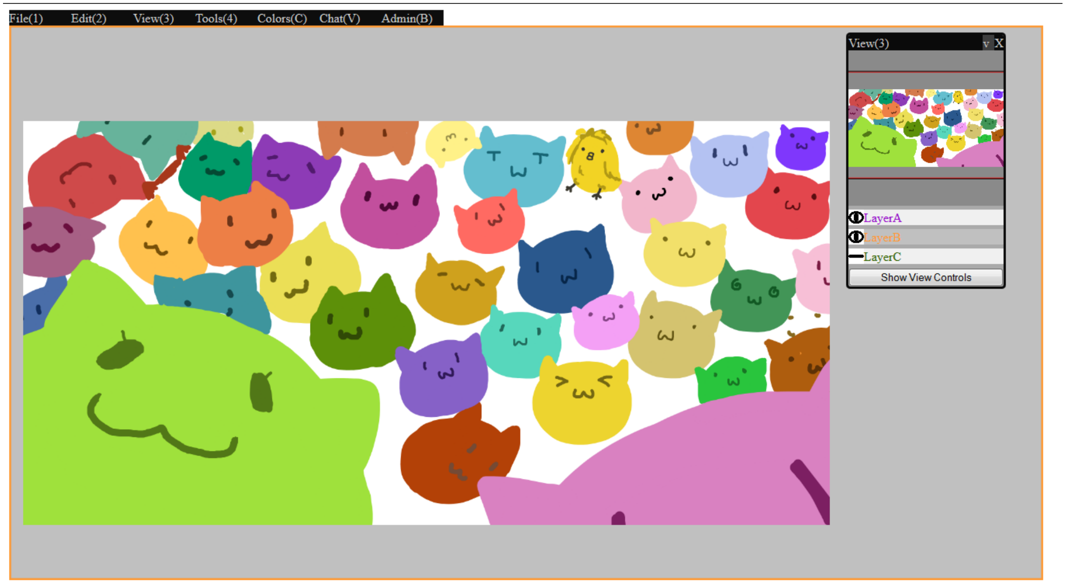
Mini-map for overview, camera state and interaction.



The view controls can be expanded or hidden.

Below the mini-map is the layers control. It displays the current layer the user is on and also provides the controls available for the layer. The layers can be locally hidden or shown in order for the user to get visual information. Because the three layers are a given set, each is given a name and colour as a identifier. The layer order can be globally repositioned and renamed by the administrator. The reason for allowing renaming of layers is to allow the administrator to indicate to other user the use of the layer. The main reason for allowing the layers to be reordered is to allow layers to be reused. Artists often draw drafts or sketches on the bottom-most layer, but once a cleaner version is drawn at a higher layer, the draft layer is useless; movable layers will allow the draft layer to be cleared and further re-used for further drawing work. Following the same method used by iScribble, a thin coloured border is placed around the View Canvas to prevent people accidently drawing on the wrong layer. This colour is also applied on other GUI displays such as the pen radius or transformation visual aids that will be discussed later. The coloured border is visible regardless of the pan, orientation or zoom of the drawing canvas. Merging of layers can be achieved under the project's system structure, but it will have a great effect on all user's actions and the back-end structure. After discussion with the project artist, we concluded that the risk and time required to implement a suitable layer merge function was too great and therefore considered to be outside the scope of this project.





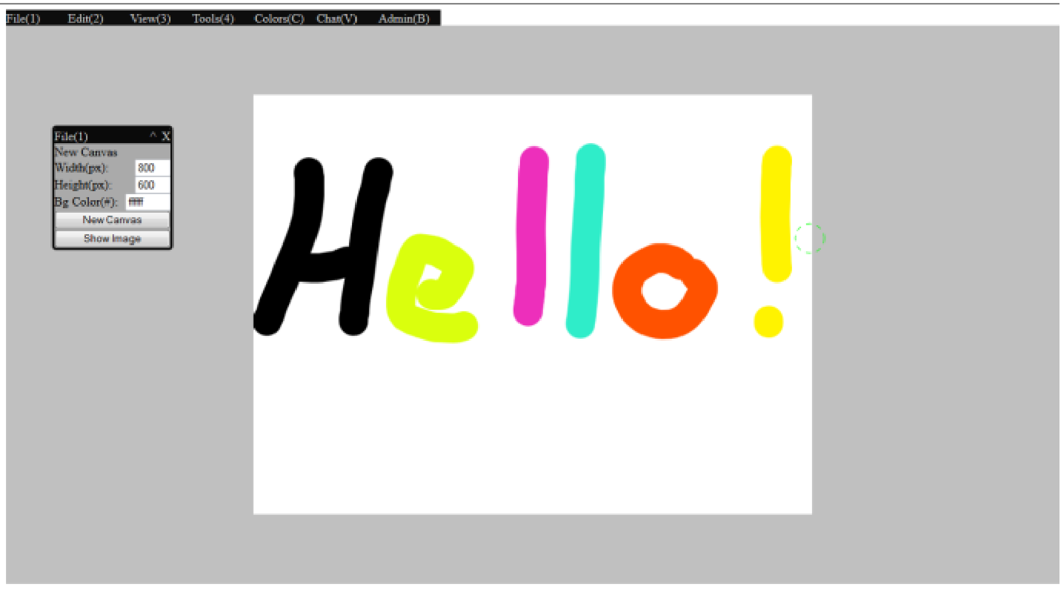
Users can locally hide and show layers

9.3 Tools

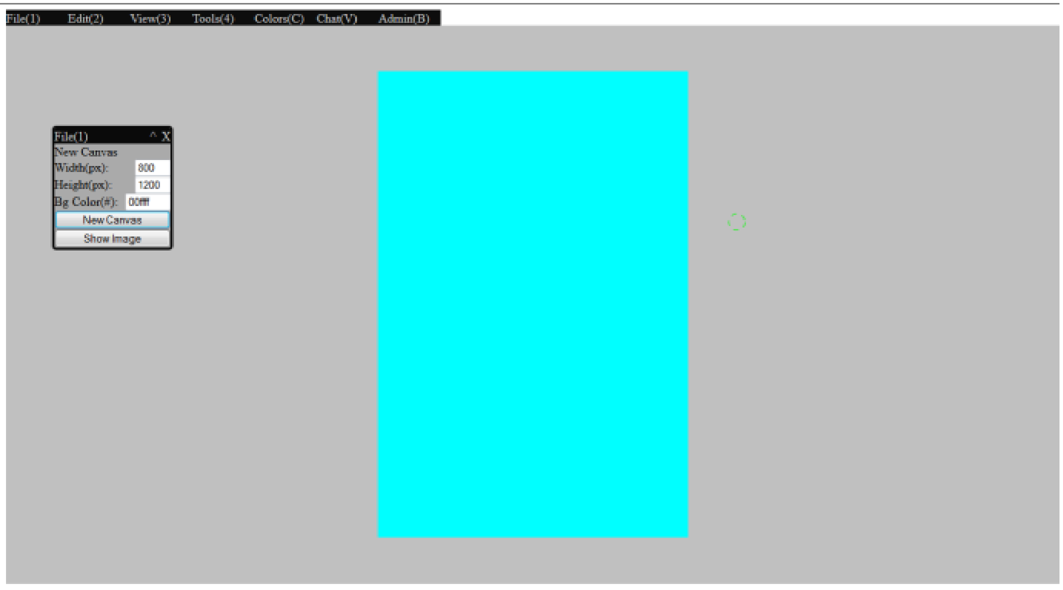
Many common tools and functions have been implemented to provide better experience in the application. All these tools have their own interface, built using Javascript, available HTML elements, and CSS. At the time of development , the HTML5 “Range” input element was not available in Firefox, only in Google Chrome and some other web browsers. In order to keep the experience consistent, custom sliders were built using Javascript. All adjustable options have a slider with matching text input. This gives the user multiple options for adjustments; being precise with text input or rough with sliders.

Furthermore the “Color” input element was only available in Chrome and Opera; other browsers’ versions were still in beta or not yet released. This element prompts the user to pick a colour from a browser dependent colour selection dialogue. However these dialogue do not have alpha control and the user cannot interact with other parts of the browser until the dialogue is closed. Therefore this element was not used and the project resorted to custom made colour selection tools to insure consistent experience across browsers.

The “File” panel contains major canvas or project controls. Users can clear and restart a new drawing with custom height and width. This “New Canvas” function also clears the history cache on the server. There is also a “Show Image” function that displays the current drawing in actual size, in a different window. This function is for when the user wants save the canvas. The Show Image function converts the Final Display Canvas to a PNG file using the Canvas’s getImageData() function, then puts that image into a newly opened window.

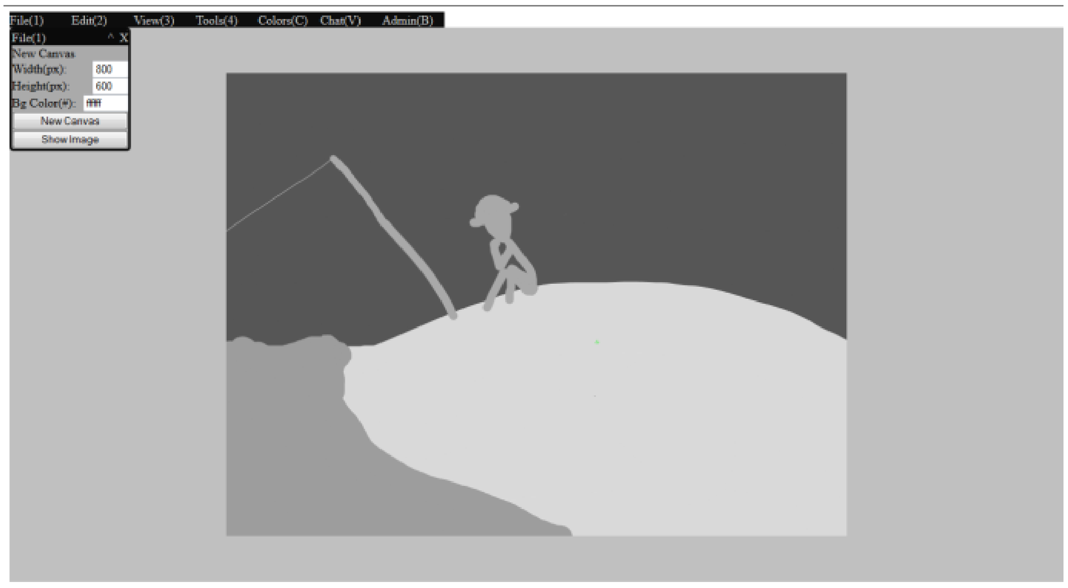


Width(px): 800
Height(px): 600
Bg Color(#): ffffff

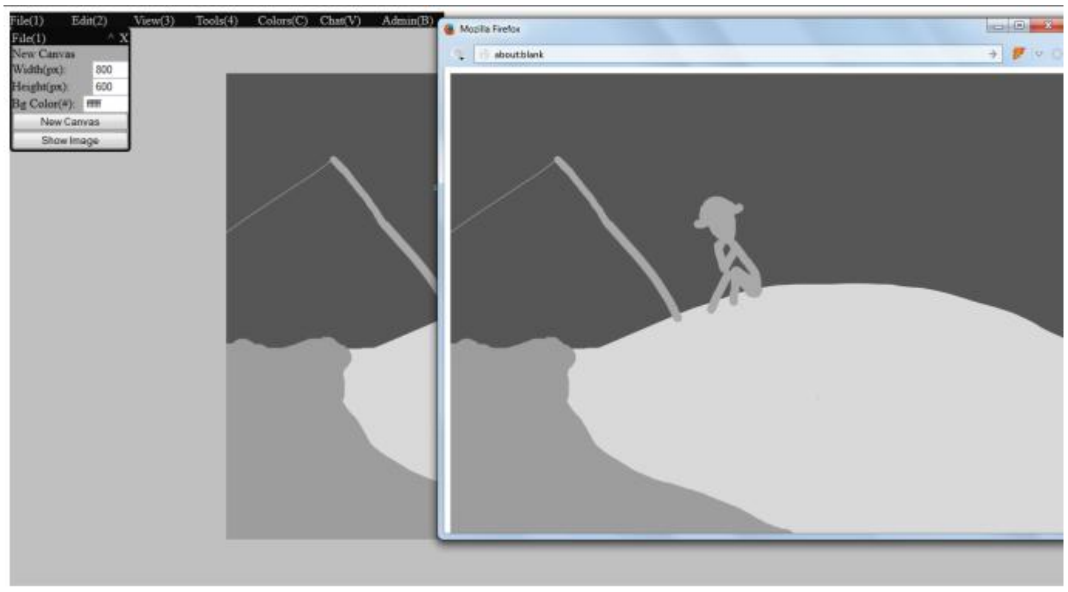


Width(px): 800
Height(px): 1200
Bg Color(#): 00ffff

New Canvas function: Creating a new canvas with a new background colour.



Width(px): 800
Height(px): 600
Bg Color(#): ffffff

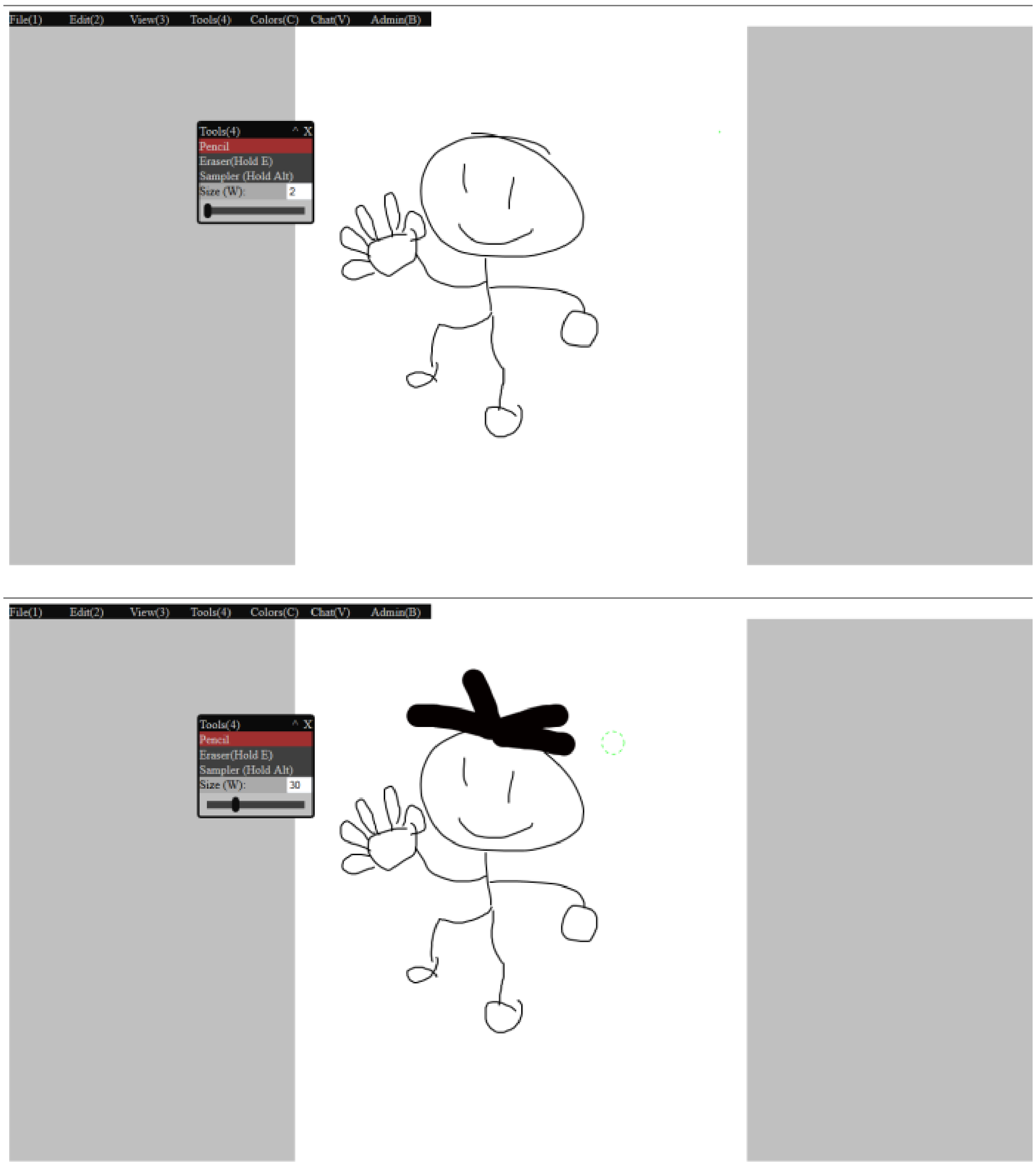


Width(px): 800
Height(px): 600
Bg Color(#): ffffff

Show Image function: Showing canvas as an image in actual size in a window.

The Pen tool is implemented using the HTML5 Canvas's inherit functions. To make the line smoother, the sample points are connected using the quadraticCurveTo() function instead of basic lineTo() function. All sample points in an action are stored in an array. When the action is called to redraw or execute, the sample points are looped through with the quadratiCurve() function using the previous two points as start and control point for the curve. The lineTo() function is used when rendering less than three points. The eraser tool, before the layers system was added, was achieved by using a stroke with the same colour as the originally-set background. Later the eraser effect was achieved by setting the

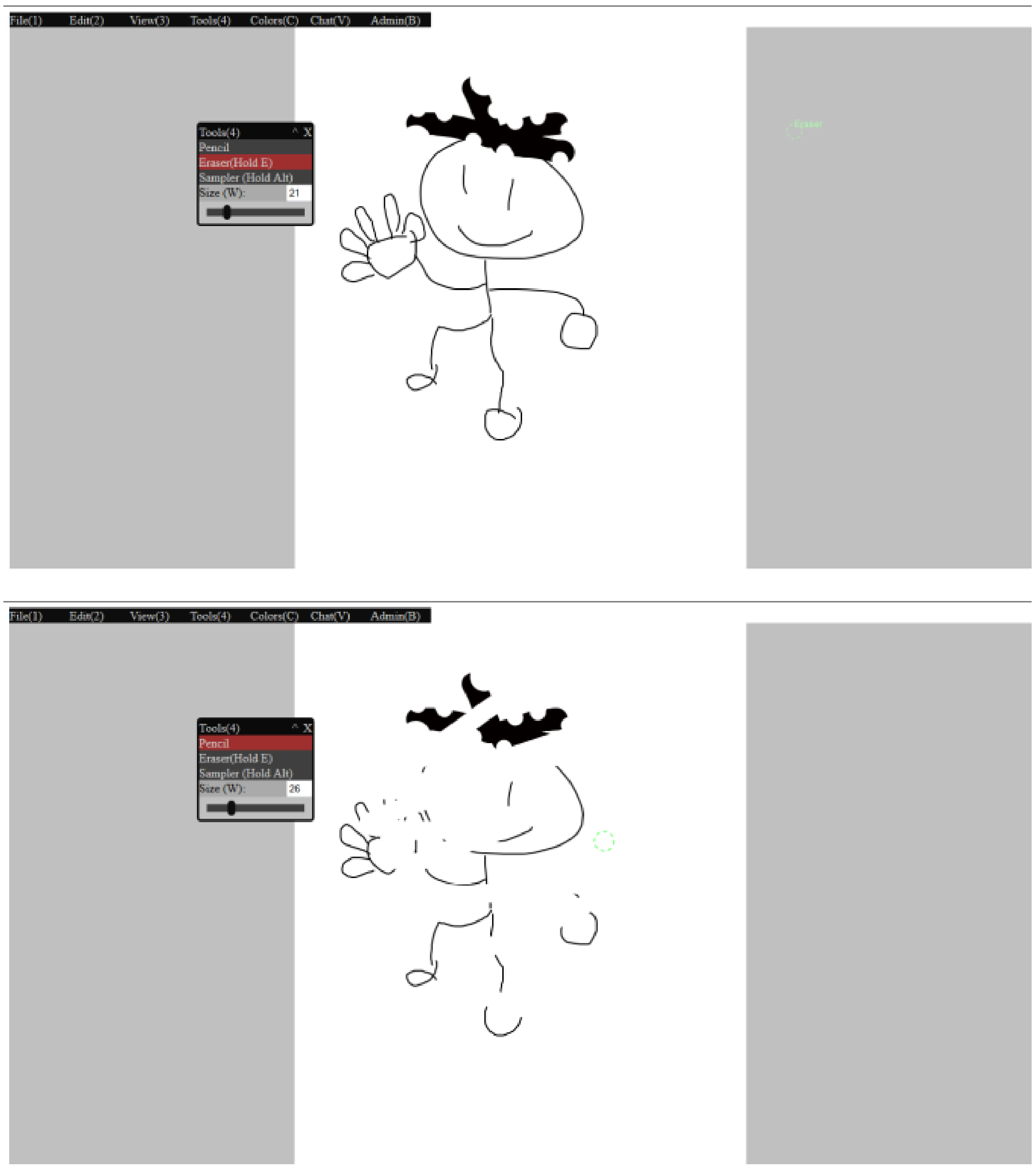
Canvas's globalCompositionOperation to "destination-out" with same code as Pen. "destination-out" makes all pixels within the stroke area "empty", which allows other layers to show through. Pen and eraser tool have their own size control slider that will appear when the tool is selected.



Size (W): 2

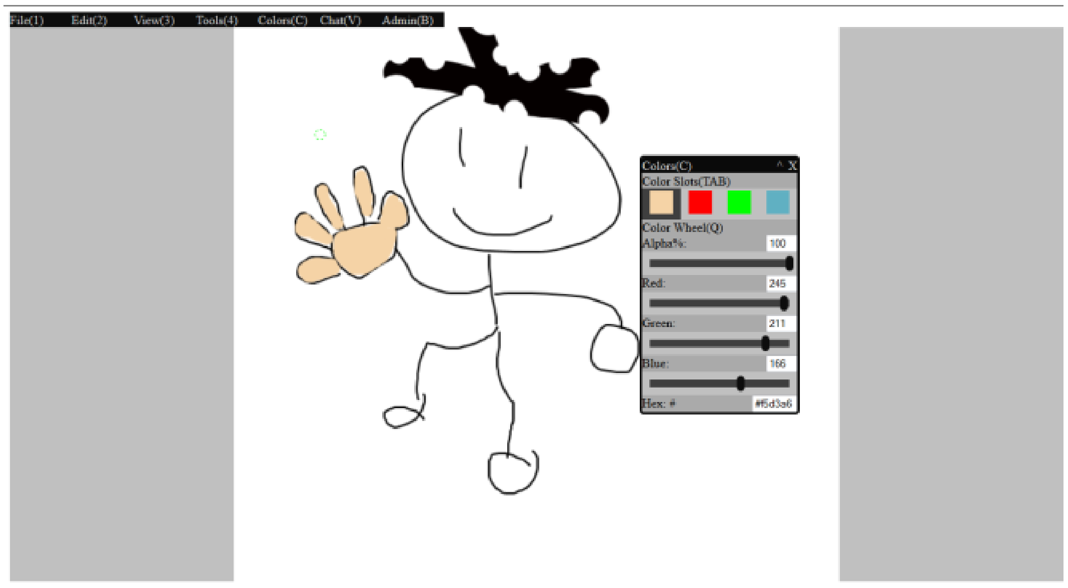
Size (W): 30

Examples of the Pen tool.



Examples of the Eraser tool.

The colour selection tool is implemented as a separate panel. Colours can be adjusted using ARGB channel sliders, text input or HEX input. As previously stated in the requirements, there are 2 facilities for choosing colours; a colour selections tool for precise colours and a colour wheel launched by a hotkey for quick colour selection. As an addition, colour slots are implemented for remembering colours and acting as a live preview while changing colours. The colour preview slots are created using small canvases to show accurate colours. Four colour slots are made available in the current version. A colour sampler tool is also implemented for convenience. As recommended by the artists, the sampler will sample the exact pixel colour seen on the canvas, which might be the result of layering colours with alpha. The project artists states that the value of colour slots varies between artists. This is due to habits gained from using other commercial drawing applications; even though modern commercial drawing applications have two colour slots, some artist have built a habit purely using colour selection and colour sampling when they need to reuse colours. Another alternative way to keep track of used colours is by drawing patches of colours in a spare space on the canvas. This way is very common in collaborative environments said the project artists; it is beneficial because colours are seen by all users and can be sampled from easily. Saving frequently used colours in slots may be important to some artists, but not for the project artists.



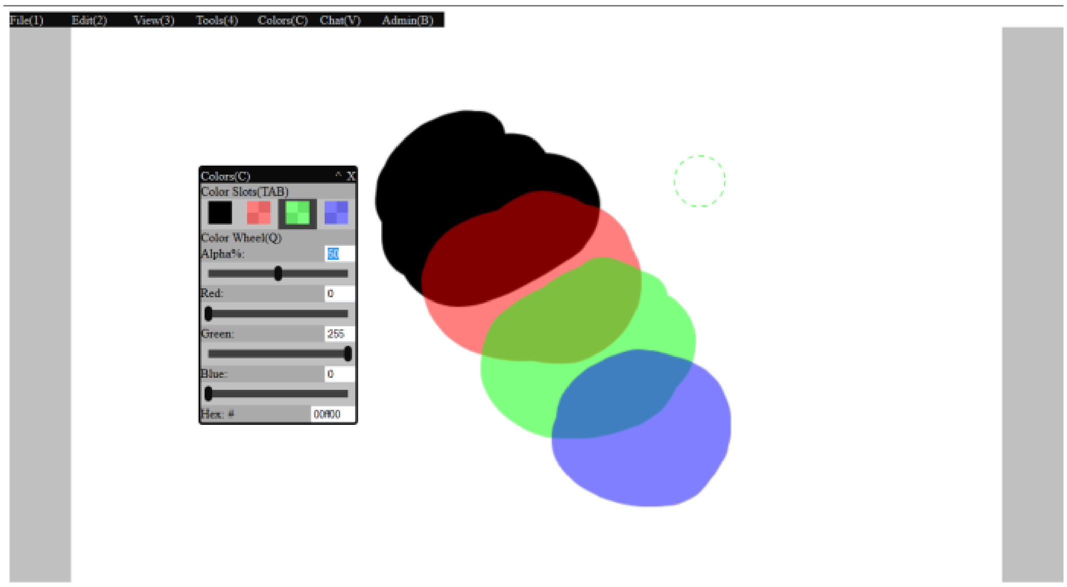
Alpha%: 100

Red: 245

Green: 211

Blue: 166

Hex: # #f5d3a6



Alpha%: 50
Red: 0
Green: 255
Blue: 0
Hex: # 00ff00

Colour selection panel: for precise colour selection.

9.4 Hotkeys

One of the crucial issue with a tool-based application is efficiency of interaction. Hotkeys for tools and functions help increase speed of interaction and make the overall experience friendlier. With help and suggestion from artists, pan, rotate, and zoom all have hotkeys and graphical information displays when their hotkey is held. As previously mentioned in the requirements, most interaction behaviours were influenced from Clip Studio as suggested by the project artists. Pan will show the centre of the view, this helps positioning any other view transformations. Rotate will show the origin and angle rotated in degrees. Zoom will show the transformation centre and zoom level. Pan can be done by holding the hotkey and dragging with the cursor. Zoom by holding the hotkey and cursor dragging out or in. Rotation can be achieved by holding the hotkey and mouse dragging in circular motion. Double-tap functions are also implemented to quickly reset each of the transformations. Whenever any hotkey is active, a small piece of text is drawn in the view canvas indicating a change of state. This helps notify the user that a change of interaction is in progress.

Achieving these interaction required many additional calculations to match the transformation function input of the HTML5 Canvas element. All transformation inputs are based on data from HTML DOM mouseMove events. The Pan effect is achieved by first calculating the vector of the movement using the current mouse position and subtracting the previous mouse position. Because the co-ordinate space in the Canvas is also transformed by any transformation functions, the global mouse movement vector needs to be rotated to the local space inside the Canvas element. Also the scale of the pan movement is dependent on the scale of the Canvas; the larger the Canvas and view, the smaller the scale of the movement in local space and vice versa. To achieve this task, a formula for rotating 2D vectors about the origin is used.

Rotate 2D vector about (0,0) formula:

$$X' = x*\mathrm{Cos}(\theta) - y*\mathrm{Sin}(\theta)$$

$$Y' = x*\mathrm{Sin}(\theta) + y*\mathrm{Cos}(\theta)$$

Where x and y is the original vector; θ is the rotation; X' and Y' is the result vector.

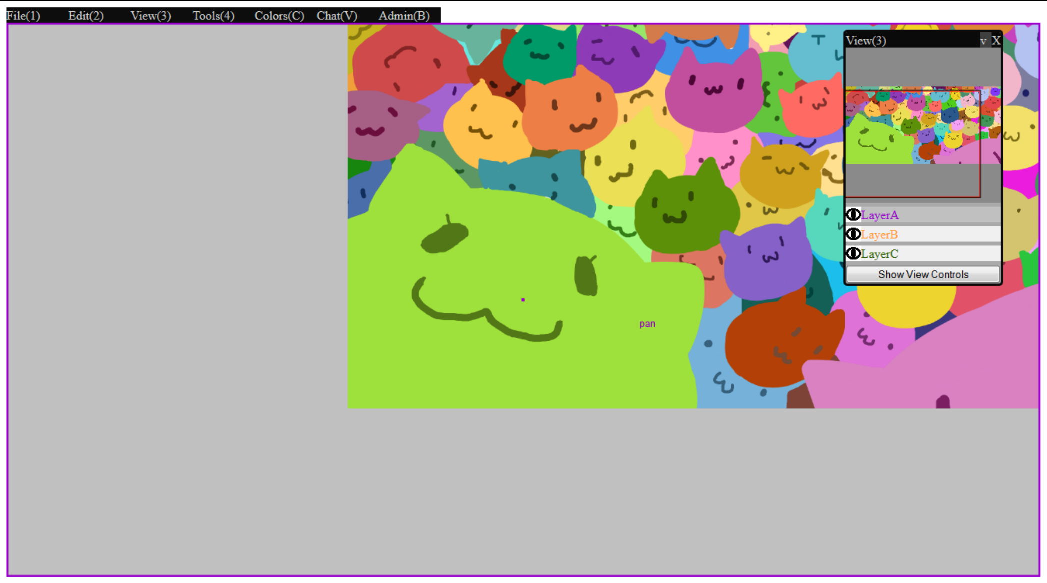
Formula used for paning the Canvas element in project:

$$X' = (dX/z)*\mathrm{Cos}(r) - (dY/z)*\mathrm{Sin}(r)$$

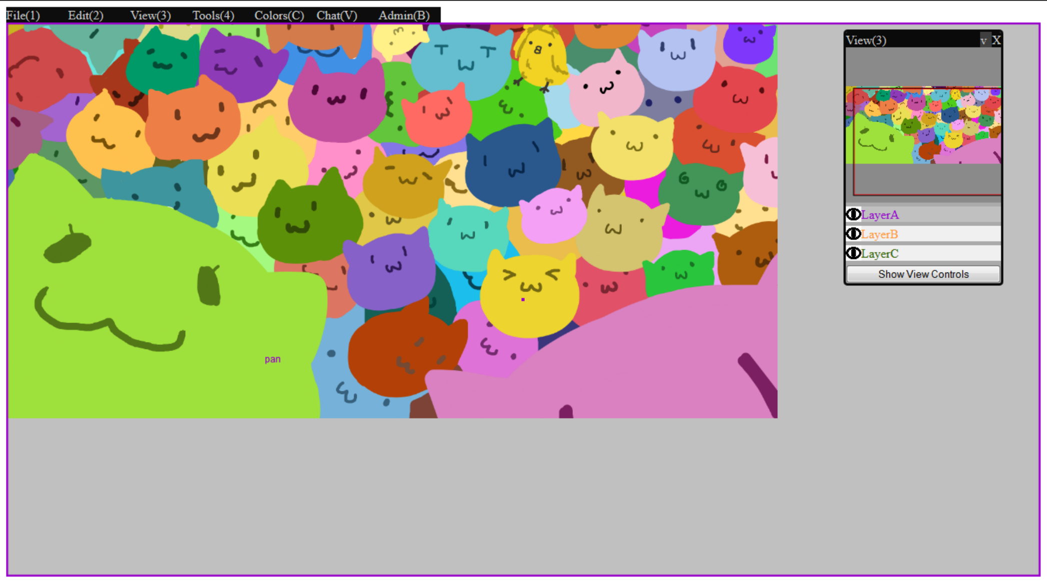
$$Y' = (dX/z)*\mathrm{Sin}(r) + (dY/z)*\mathrm{Cos}(r)$$

Where dX and dY is the 2D global vector derived from the difference between current and previous cursor position; z is the current scale of the Canvas; r is the current rotation of the Canvas in radians; X' and Y' is the result vector.

The resultant vector is then added onto the Position variable of the Canvas. The Position variable is the translated position used when rendering the transformed Canvas. As a result, the user can pan the view freely in one-to-one pixel ratio because canvas scale and rotation are accounted for.



When the Pan hotkey is held down, ‘pan’ is displayed on the mouse cursor to give feedback on the key held.



Panning view to the right.

Canvas rotation interaction is achieved by making the previous and current mouse input points into vectors coming from the centre of the view. The angle between the two vectors is calculated. Lastly, a direction of rotation is calculated using the cross product of the vectors. The direction can be determined by calculating the third component of the cross product between the vector, from view centre to previous input point, and vector, from previous input point to the current input point.

Cross product formula for vector U(u1,u2,u3) and V(v1,v2,v3) under axis of i, j, k:

$$U = u1i + u2j + u3k$$

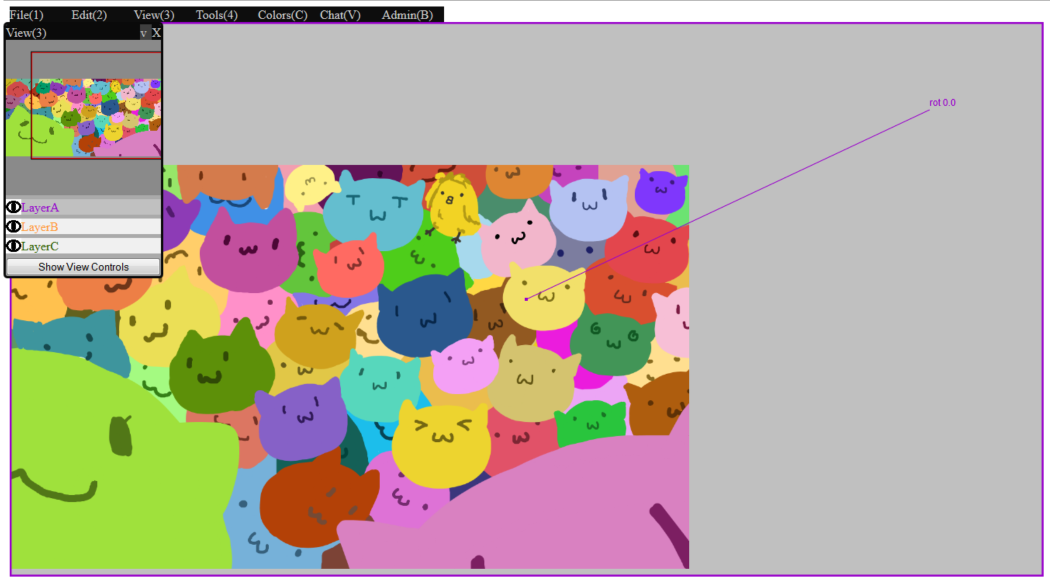
$$V = v1i + v2j + v3k$$

$$U \times V = (u2*v3 - u3*v2)i + (u3*v1 - u1*v3)j + (u1*v2 - u2*v1)k$$

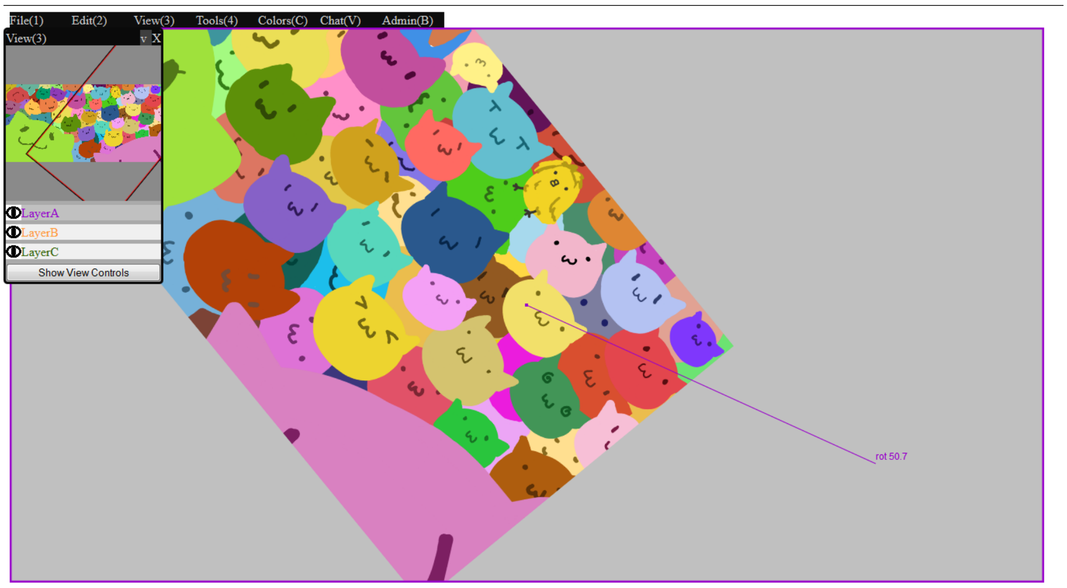
From the above formula, if the third component of both vector U and V is taken as 0 because we only have 2D mouse input, the formula can be simplified to:

$$U \times V = (u1*v2 - u2*v1)k$$

As shown, the third component is naturally left as the result of the cross product. The cross product helps determine the direction in which the canvas needs to rotate. If the cross product of those two vector is greater than 0, canvas should rotate clockwise and if less than zero, anticlockwise.

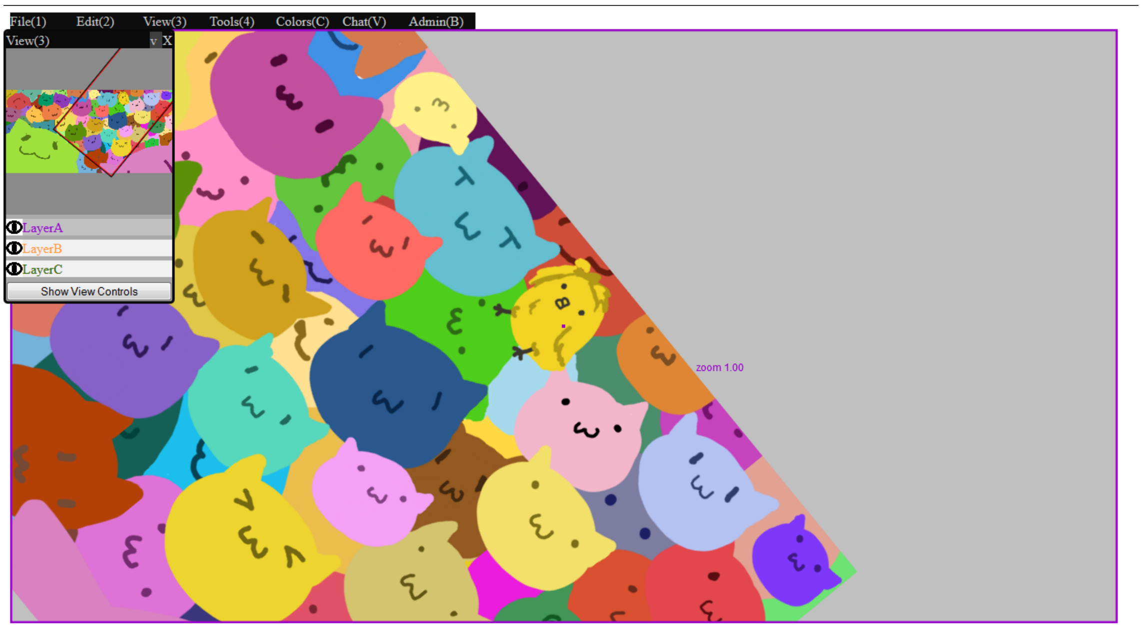


Rotate hotkey is held; display ‘rotate’, current canvas angle and line towards centre of rotation to the user.

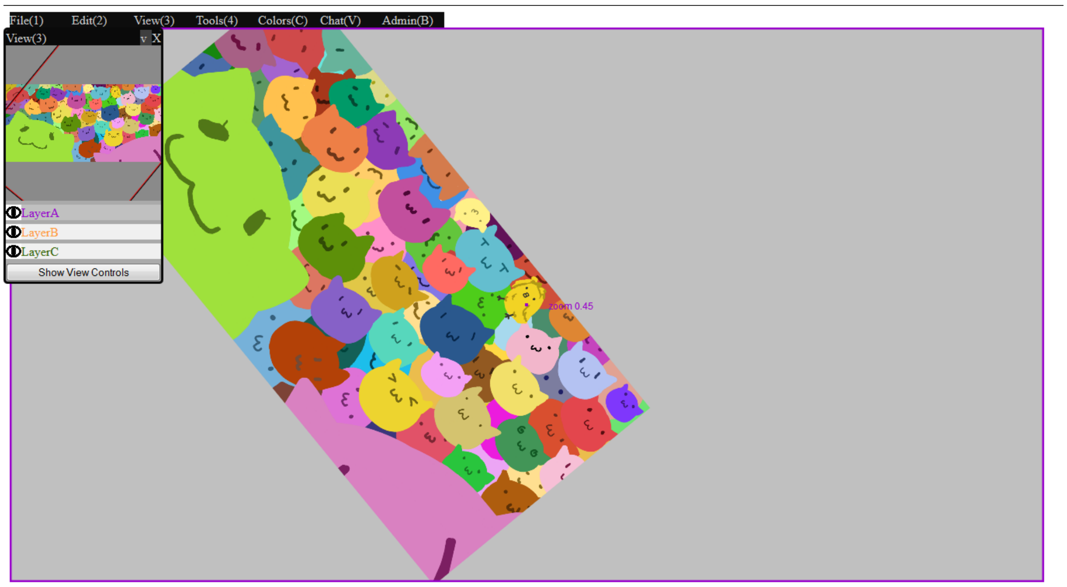


Rotating view by dragging the cursor down.

Canvas scaling is implemented using trigonometry and the dot product. The direction vector of the change in mouse position determines the direction of scaling. The distance between prior and current mouse points determines the magnitude of the change. If the dot product between the vector towards centre of view and the vector of mouse movement is greater than 0, then the canvas scales down, otherwise up. The magnitude of the scaling is determined by the magnitude of the mouse movement vector through using Pythagoras' theorem and then dividing by the current scale.



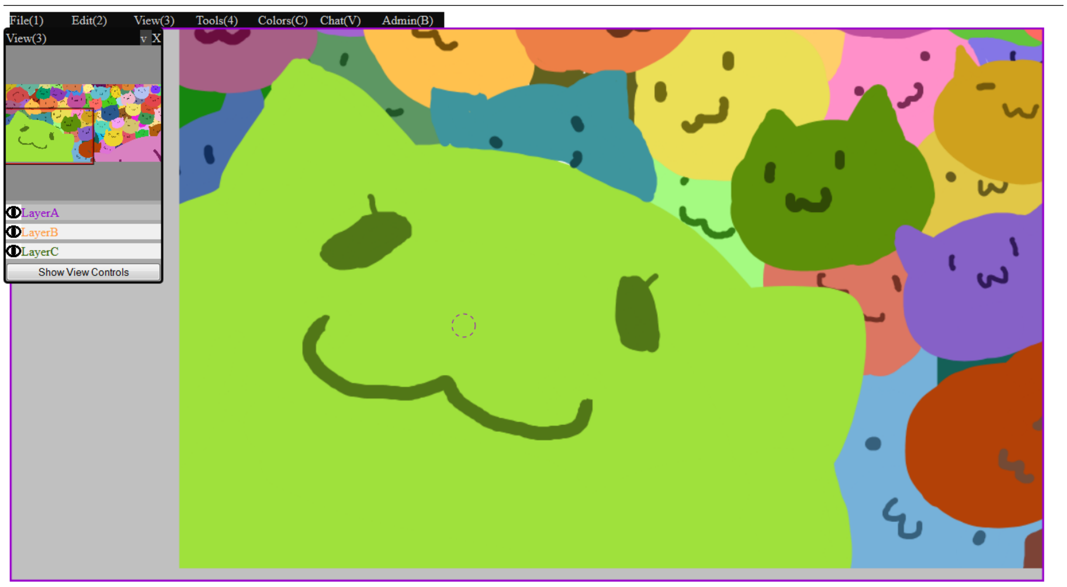
Zoom key is held down, display ‘zoom’ and the zoom level on top of cursor.



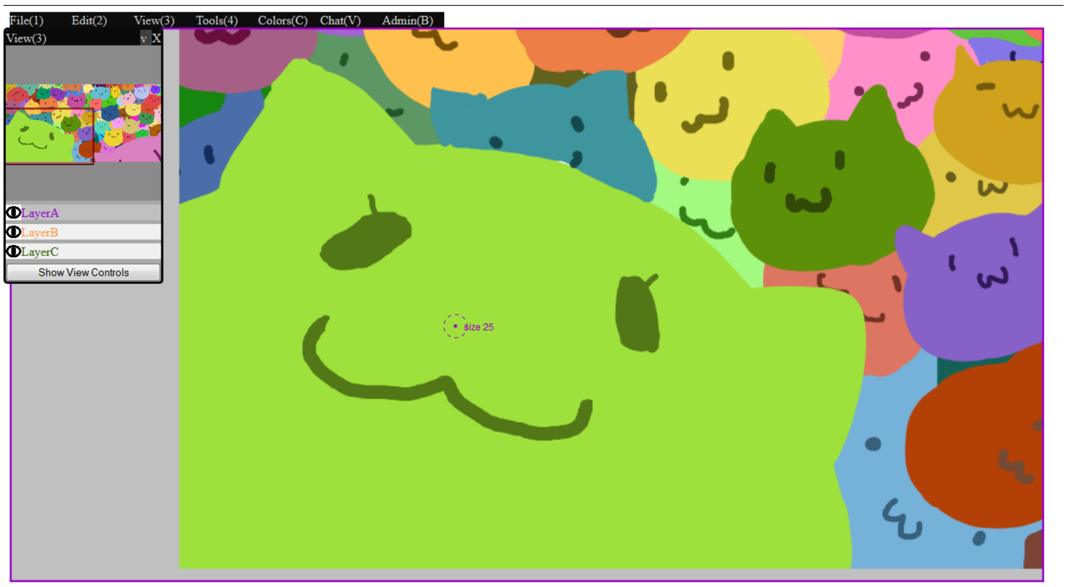
Zooming out by dragging towards the centre of the view.

The project artist states that having a clean, unobstructed work view is more important than features and functions that are rarely used. Following examples from existing software, all tools, except for the default pen, are given a held-down hotkey function. This means while the hotkey is held down, a tool is selected and used. However once the hotkey is released, the tool that was been previously selected(most likely the pen tool because it is default) will be re-selected automatically. This gives the user the ability to quickly use different tools without the need to explicitly switch back. An example might be to quickly switch between the pen and sampler tool by holding the hotkey for the sampler, sample, then when the hotkey was released, the pen is automatically re-selected, ready to draw straightaway.

Aside from selecting tools, common adjustments to the tools itself is as frequently used. The adjustments to pen size is given a hotkey interaction similar to Clip Studio’s. When its hotkey is held, the size of the current tool is shown with the cursor at the edge of its radius. The user can then drag in or out adjusting the size of the tool freely until the hotkey is released. The eraser size can also be adjusted the same way with that same hotkey while the eraser is selected or while the eraser hotkey is also held.

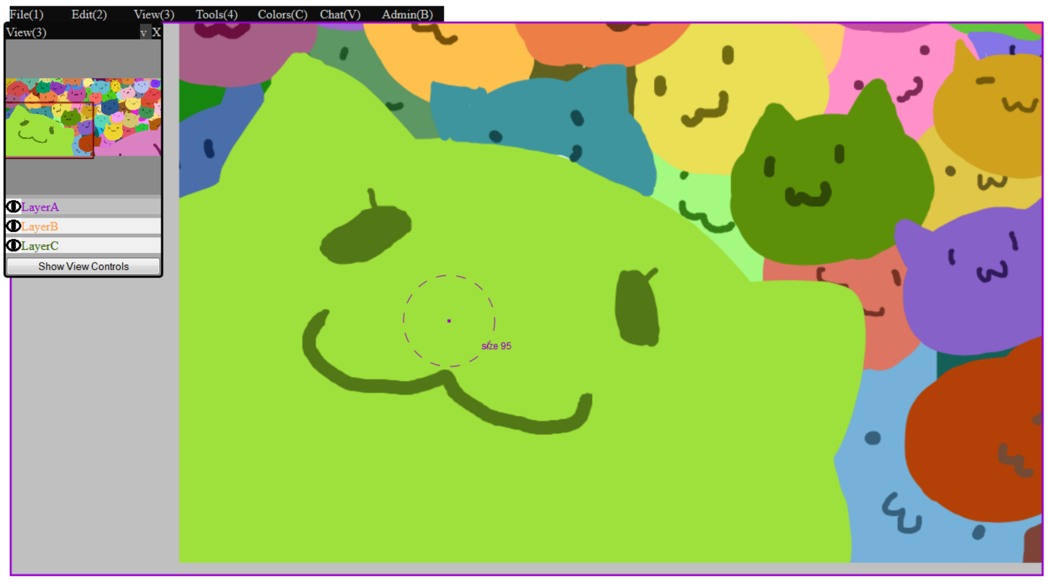


By default, the pen radius is displayed around the cursor to give a preview of the current stroke size.



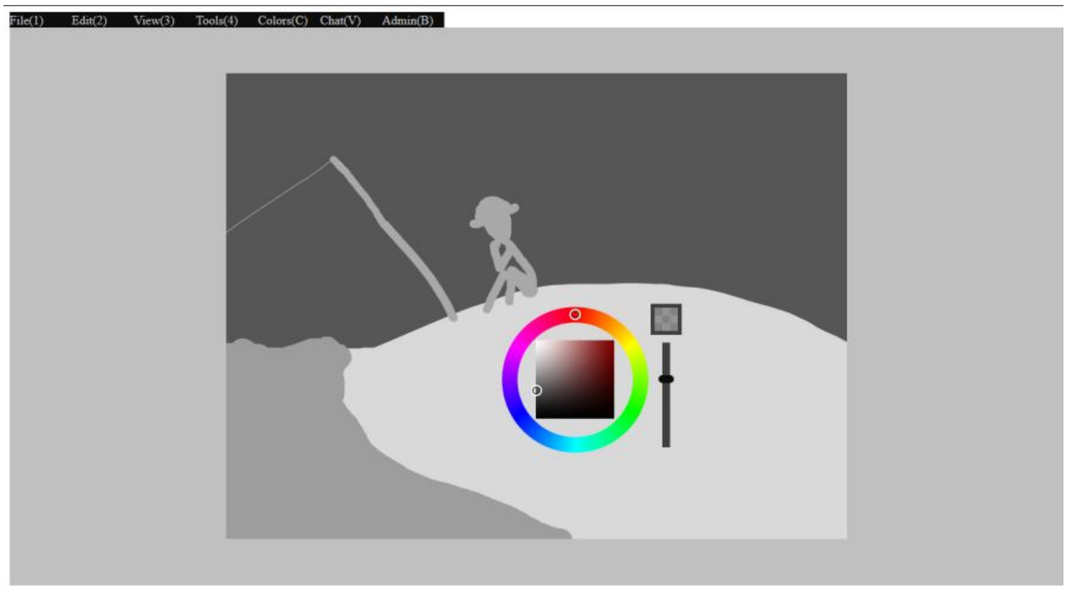
When hotkey for size change is held, the pen radius visual will move to top left with the edge of the circle touching the cursor. The current size is also displayed.

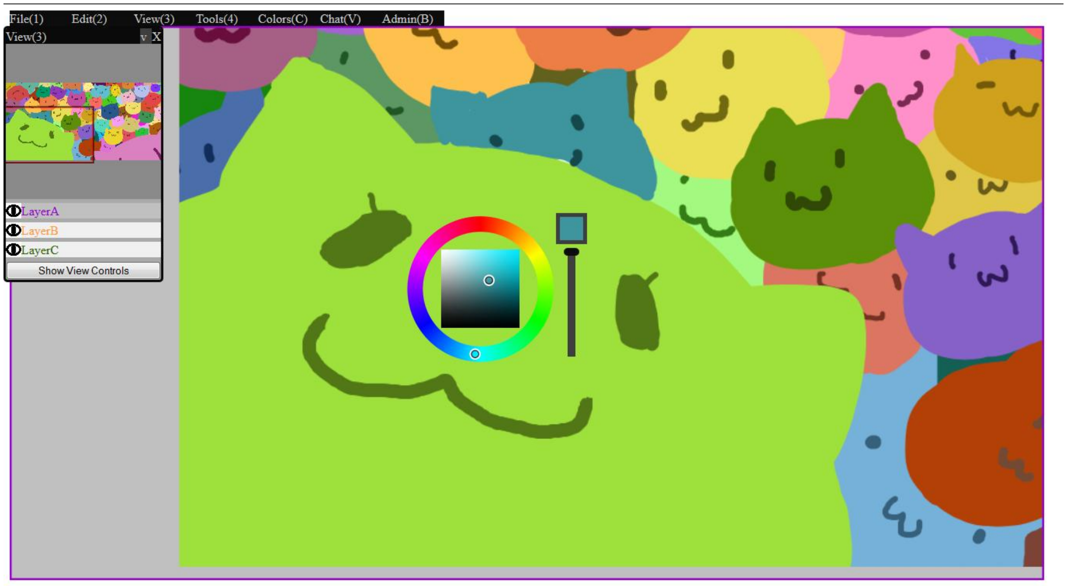
The user can click and drag in or out to a desired radius.



Once the hotkey is released, the new size is set and drawing can resume.

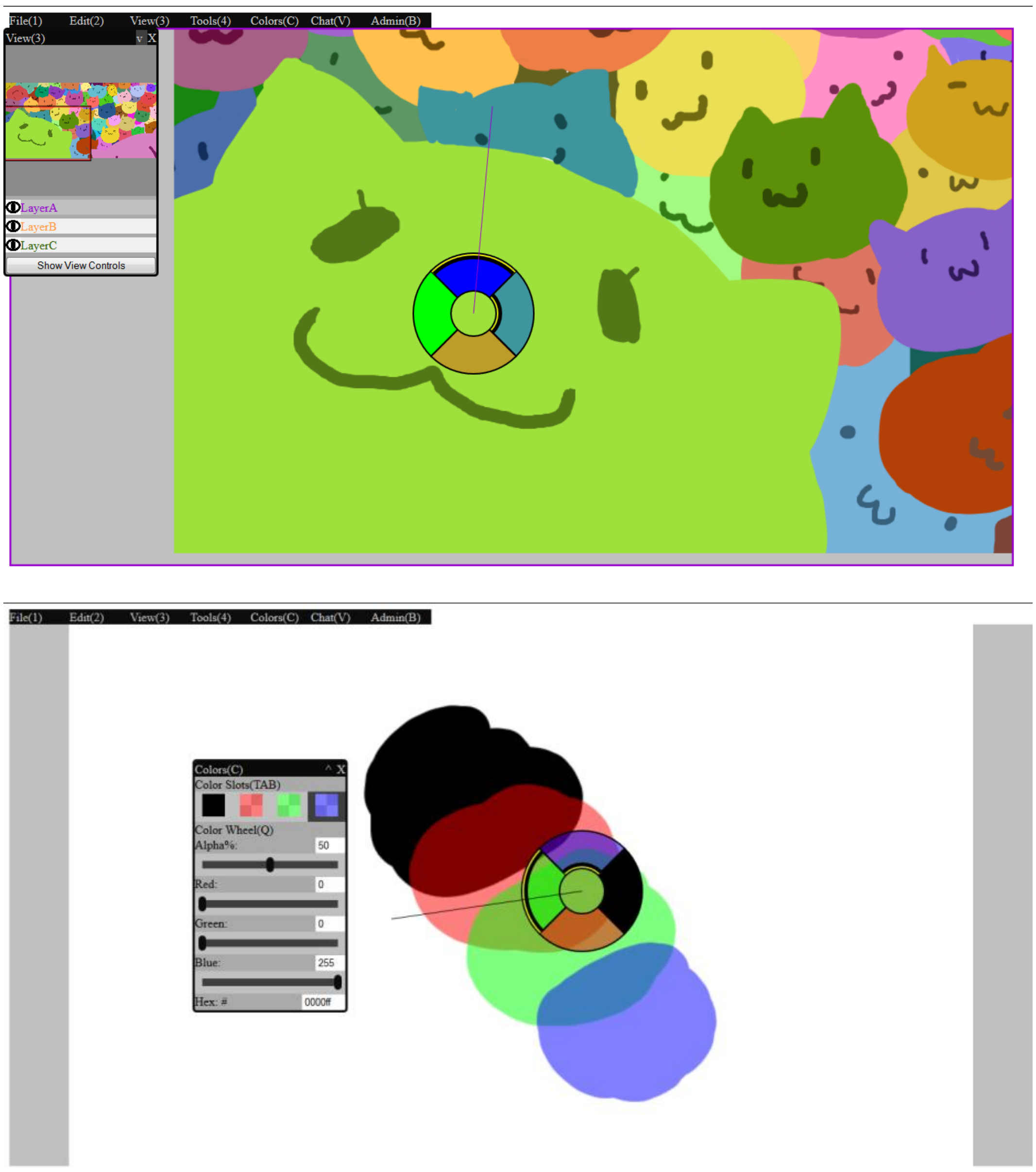
As specified in the requirements, a custom colour wheel bind to a hotkey was implemented as an alternative quick method of colour selection. The goal of the colour wheel is to speed up colour selection by allowing the user to make less actions. The colour wheel would need a maximum of three actions to select a colour; selecting colour hue, saturation with luminosity, and alpha. Traditional sliders would need a maximum of four actions; adjusting each of the three colour component channels and alpha. Therefore a colour wheel is a faster method for selecting colours, but with sacrifice of selecting precise values. The implemented colour wheel has an colour preview area at the top right and a vertical slider for alpha channel below that. When the colour wheel hotkey is held down, it will pop up where the cursor is for interaction. This allows the user to change colours without having to move the mouse too far. The floating colour wheel is created using combination of HTML DIV elements, CSS, and Javascript. The colour wheel, centre colour mask, and colour preview are just background images that get their background colours modified through CSS. The vertical slider is a modified version of the custom slider made for other panels. The colours are calculated in Javascript rather than sampled from the image. This structure allows freedom for modification if the colour wheel needs to be re-scaled in the future.





Custom floating colour wheel

As an experiment, a radial menu for the four colour slots was implemented. The idea of a radial menu is that all selectable objects are placed evenly in a circular format; when the user is trying to select an object, only the angle of motion of the mouse from the centre of the circle is relevant to deciding what object is going to be selected. This allows the user to quickly swing the mouse at a vague position and still be able to make their selection. The major limitation of a radial menu is, if too many selectable objects exist, the finer the margin of selection becomes, and the closer it is to a normal menu in usage because exact position becomes more important. This limitation however is not an issue with four options. One notable property is that the radial menu is purely drawn in the View Canvas, it has the potential to interact with the Canvas in the background if needed. This effect is obvious when the line from the centre of the Radial Menu is drawn out towards the cursor. Potential benefits from this type of GUI could be further explored in the future. As previously mentioned colour slots might not be used as frequently depending on the user's habit. Hopefully with colours slots being more accessible with the radial menu, colour slots might increase in usage.

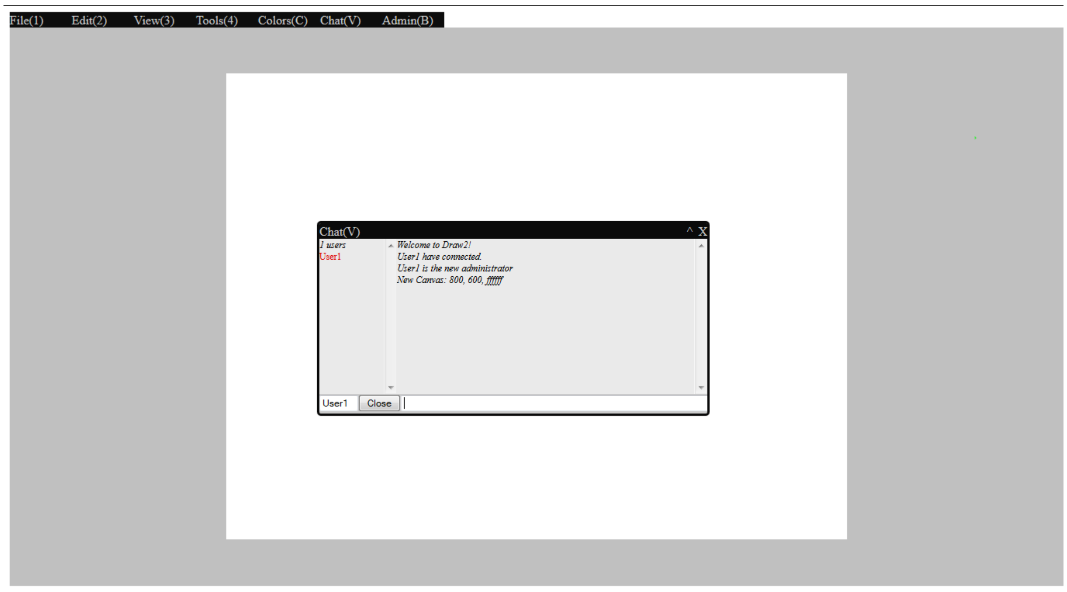


Alpha%: 50
Red: 0
Green: 0
Blue: 255
Hex: # 0000ff

Floating radial menu for quickly choosing colour slots.

9.5 Social Functions

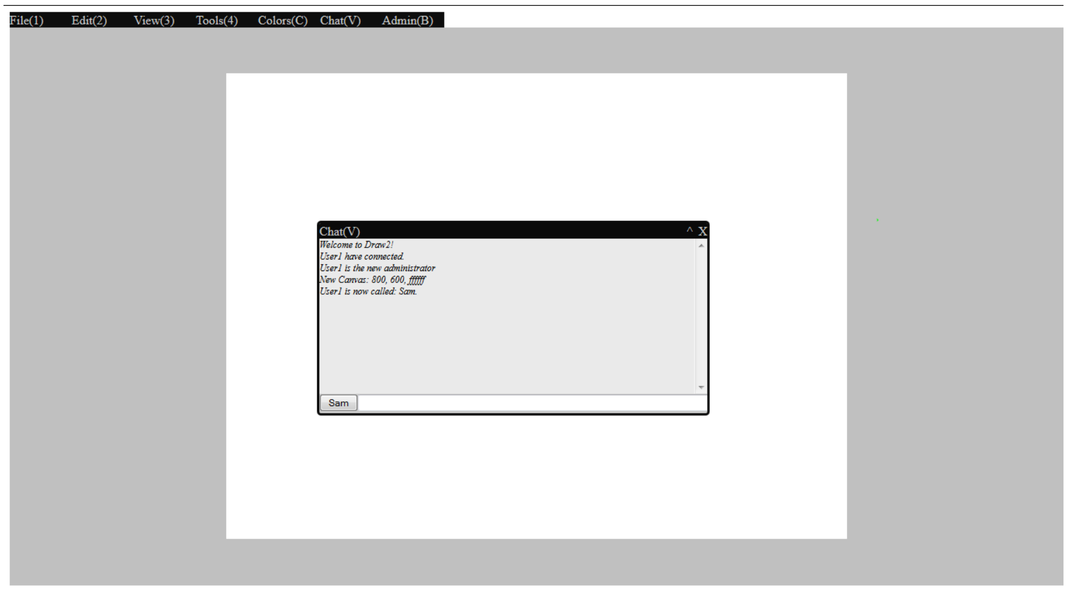
With any social interactive application, the ability to communicate with text is required. A custom chat system was implemented. The chat system allows users to see other users, communicate with text, change their name, and see any important system events that happened. As a privacy measure, chat messages are not recorded in the server history. Each user is given a unique colour connection, so even if they have the same name, they can still be identified. This colour is also consistent with the name tag that is displayed when they are performing actions on the canvas. Following from example of Pixiv Chat, a "chat hint" system is implemented. It allows the convenience of "listening in" in on conversations without the need of having a full chat system interface displayed. The hints will show up for ten seconds before disappearing or until it is showing the tenth oldest message. If a hint is clicked, the chat panel will display at that position for the user to interact with.



1 users
User1

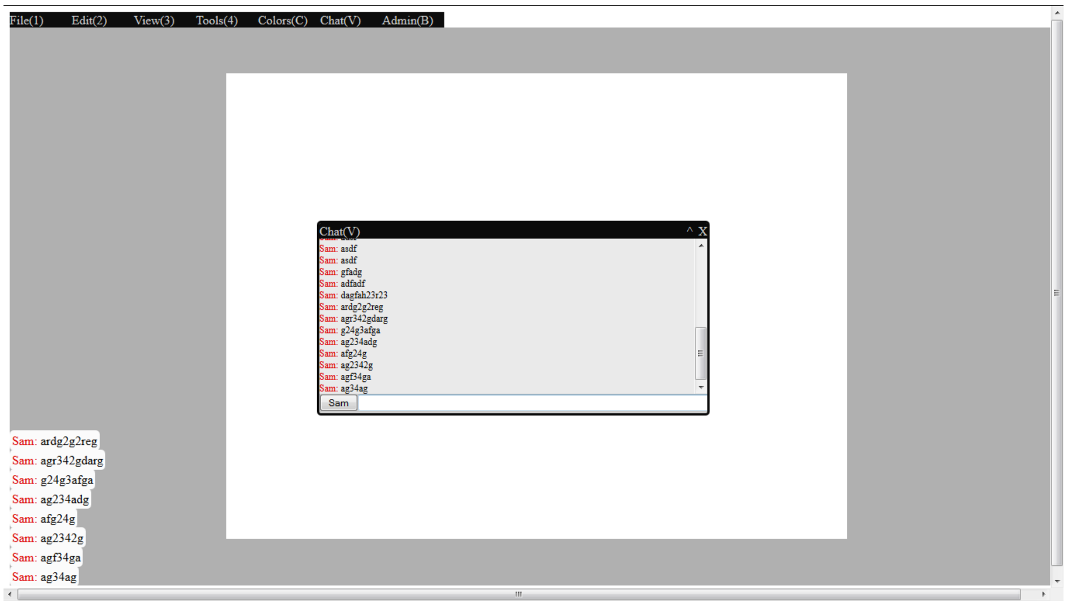
Welcome to Draw2!
User1 have connected.
User1 is the new administrator
New Canvas: 800, 600, ffffff

The chat system inside its own panel.



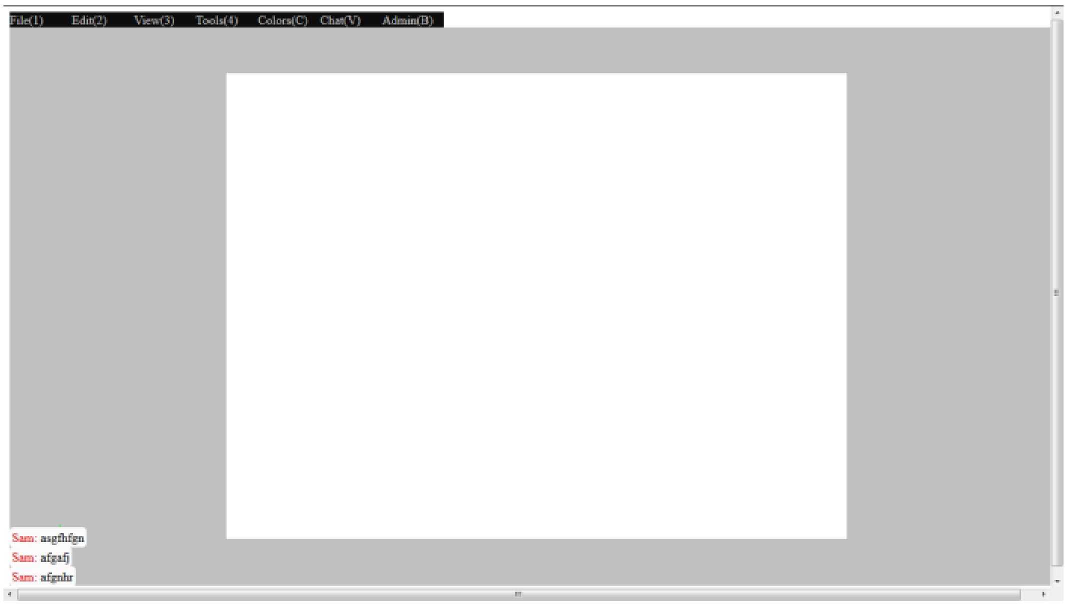
```
Welcome to Draw2!
User1 have connected.
User1 is the new administrator
New Canvas: 800, 600, ffffff
User1 is now called: Sam.
```

Chat panel view can be simplified to just the chat log.



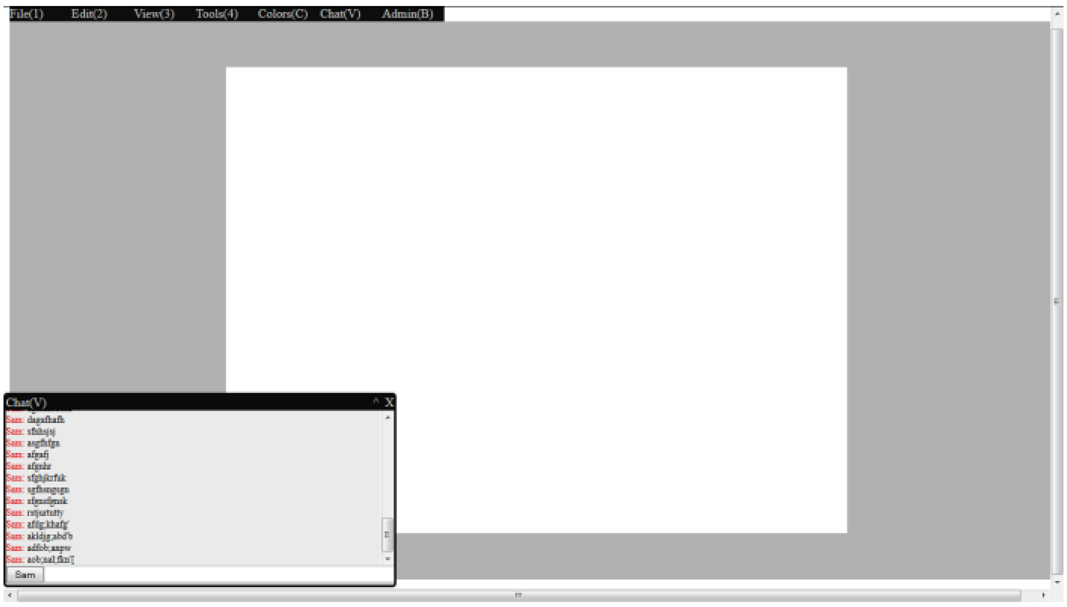
```
Sam: asdf
Sam: asdf
Sam: gfadg
Sam: adfadf
Sam: dagfah23r23
Sam: ardg2g2reg
Sam: agr342gdarg
Sam: g24g3afga
Sam: ag234adg
Sam: afg24g
Sam: ag2342g
Sam: agf34ga
Sam: ag34ag
```

```
Sam: ardg2g2reg
Sam: agr342gdarg
Sam: g24g3afga
Sam: ag234adg
Sam: afg24g
Sam: ag2342g
Sam: agf34ga
Sam: ag34ag
```



Sam: asgfhfgn
Sam: afgafj
Sam: afgnhr

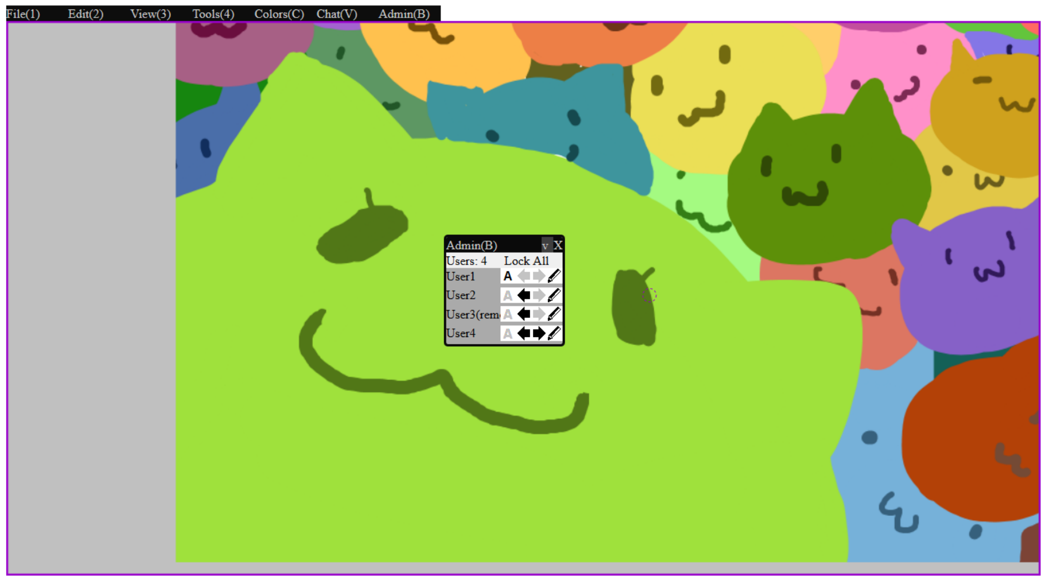
Temporary message alert will appear in canvas without the need of keeping the chat log open. These messages will disappear in 10 seconds or stack to maximum of 10.



Sam: dagafhafh
Sam: sfshsjsj
Sam: asgfhfgn
Sam: afgafj
Sam: afgnhr
Sam: sfghjkrfsk
Sam: sgfhsngsgn
Sam: sfgnsfgnsk
Sam: rstjsrtutty
Sam: afdg;kbafg'
Sam: akldjg;abd'b
Sam: adfob;anpw
Sam: aob;nal;fkn'[

Temporary messages can be clicked to initiate chat.

As a requirement, functions to prevent negative social interactions are addressed. As discussed previously, one of the biggest threats is one user with ill intent erasing or ruining other users' drawing. As a solution, an administrator system and a set of administrator-only actions has been implemented. The room administrator is initially chosen to be the first user connecting to the room. The administrator has the right to start a new drawing, hence wipe the history and canvas clean. The administrator can disable and enable users from drawing on the canvas. A “Lock All” button is also available to quickly disable all users from interacting with the canvas. This button can be used as a “panic pause“ function for everyone to stop and observe the situation. If a user is in the middle of performing an action when the disable command is sent, the action will be cancelled. While users drawing is disabled, they can still interact through the chat system. The administrator also has the right to request Undo or Redo on every users’ actions. Undo and Redo from the administrator can still be requested to server even if a user is disabled; this allows the administrator to pick and roll back on any recent actions made. The administrative rights can also be passed on to another user if the current administrator chooses to. Mass public user testing is outside the scope of this project, so some of these functions or user interface may require future modifications. The project artists claim that having these functions available is already a step towards a better collaborative drawing experience.

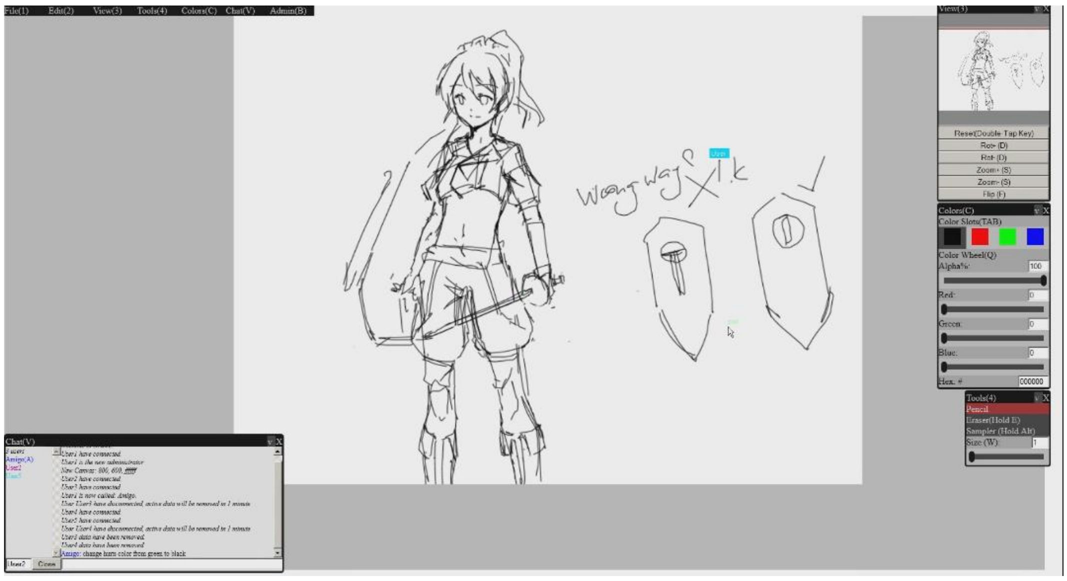


Administrator panel, Admin right can be given away, Undo, Redo, Lock drawing.

Chapter 10: Final Testing Sessions and Evaluations

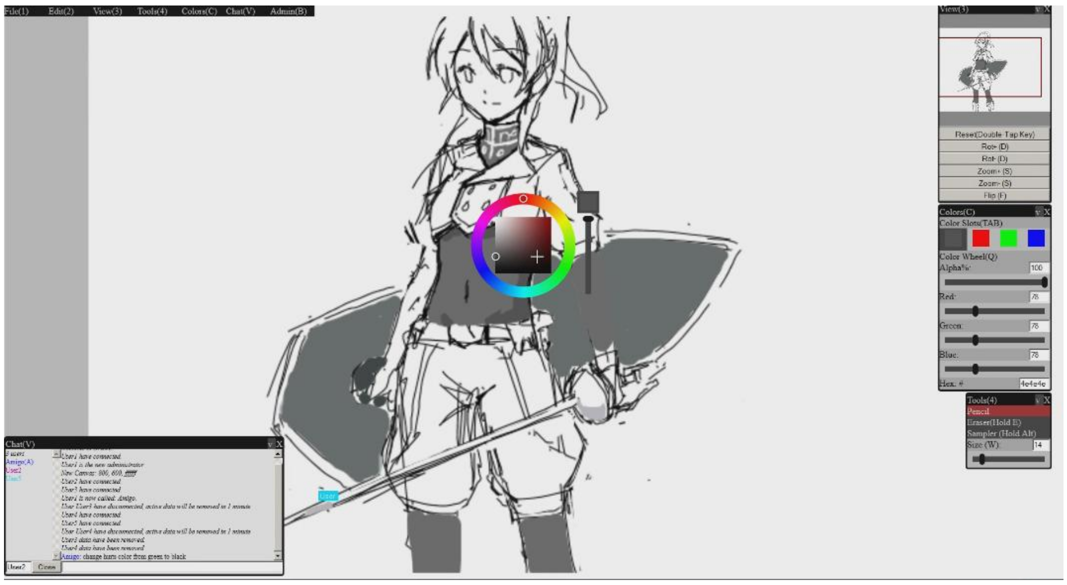
Throughout the development process of implementing required features, small testing sessions were held between the project artists and the researcher. These sessions were used to test and evaluate the artists satisfaction with features addressing each requirement. The small testing sessions were generally informal and only used whichever project artist was free at the time. Once all requirements were addressed and evaluated on individual basis, larger testing sessions were held to evaluate the overall experience. These sessions involved both project artists collaborating on a given drawing task. The task were either drawing a reference photograph or drawing an agreed topics. Earlier sessions results were mainly about fixing odd bugs and minor interface issues. Later sessions evaluated major issues with the overall implementation of the application.

Below are some time lapse screen recordings of an early larger test session. Collaboration task given: draw an image of a knight. Duration: 26 minutes.



3 users
Amigo(A)
User2
User3

User1 have connected
User1 is the new administrator
New Canvas: 800, 600, ffffff
User2 have connected
User3 have connected
User1 is now called: Amigo.
User User3 have disconnected, active data will be removed in 1 minute
User4 have connected
User5 have connected
User User4 have disconnected, active data will be removed in 1 minute
User3 data have been removed.
User4 data have been removed.
Amigo: change hurts color from green to black

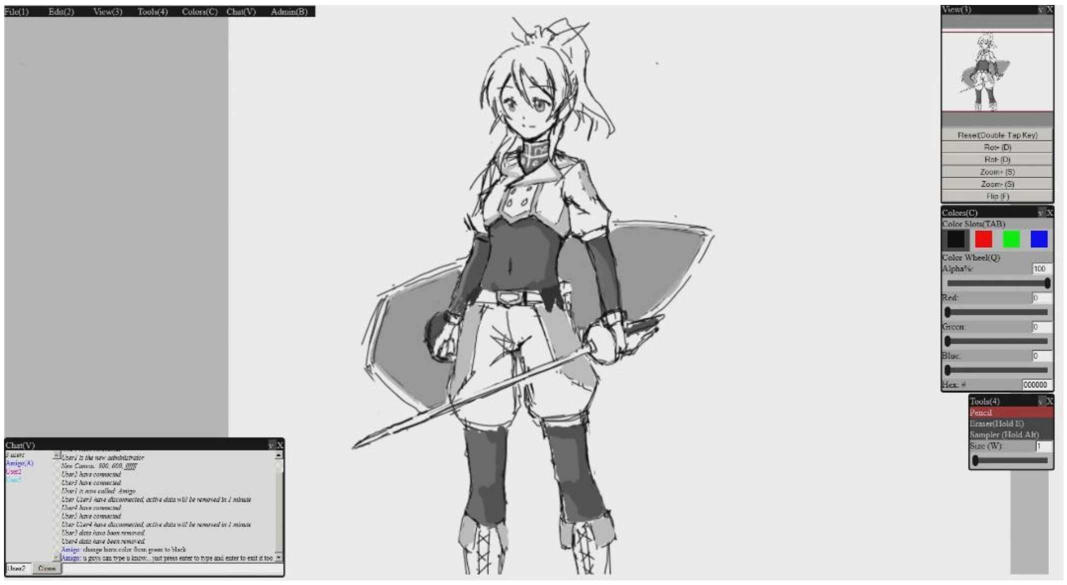


Alpha%: 100
Red: 78
Green: 78
Blue: 78
Hex: # 4e4e4e
Size (W): 14

```
3 users
Amigo(A)
User2
User5
```

```
User1 have connected.
User1 is the new administrator
New Canvas: 800, 600, ffffff
User2 have connected.
User3 have connected.
User1 is now called: Amigo.
User User3 have disconnected, active data will be removed in 1 minute
User4 have connected.
User5 have connected.
User User4 have disconnected, active data will be removed in 1 minute
User3 data have been removed.
User4 data have been removed.
Amigo: change hurts color from green to black
```

User2



Alpha%: 100
Red: 0
Green: 0
Blue: 0
Hex: # 000000
Size (W): 1

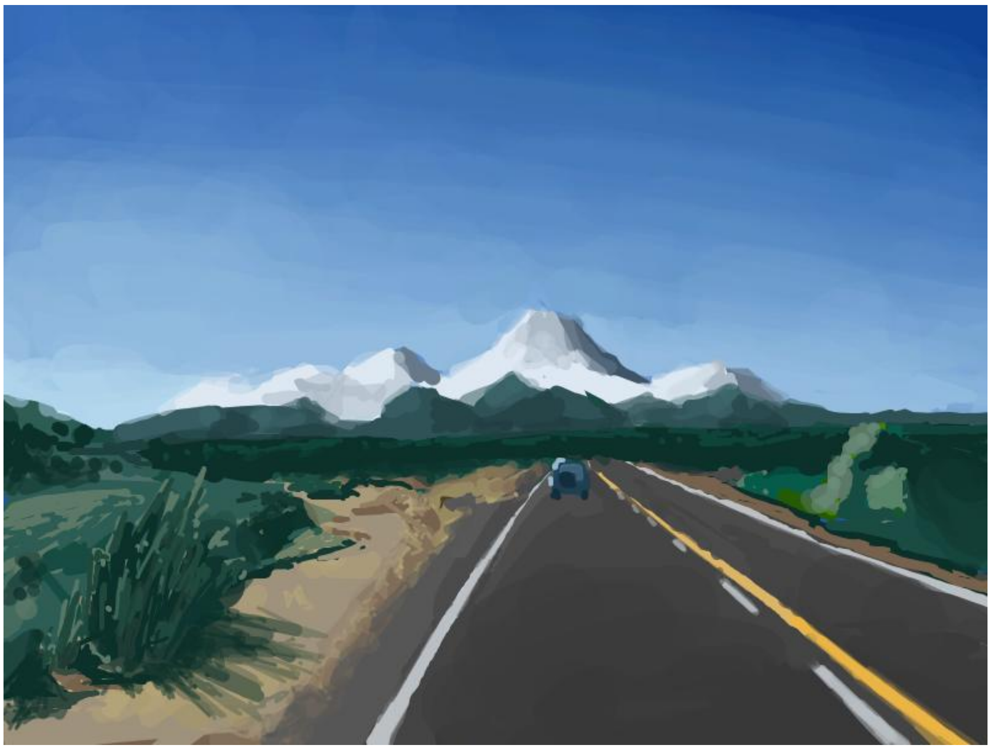
```
3 users
Amigo(A)
User2
User5
```

```
User1 is the new administrator
New Canvas: 800, 600, ffffff
User2 have connected.
User3 have connected.
User1 is now called: Amigo.
User User3 have disconnected, active data will be removed in 1 minute
User4 have connected.
User5 have connected.
User User4 have disconnected, active data will be removed in 1 minute
User3 data have been removed.
User4 data have been removed.
Amigo: change hurts color from green to black
Amigo: u guys can type u know... just press enter to type and enter to exit it too
```

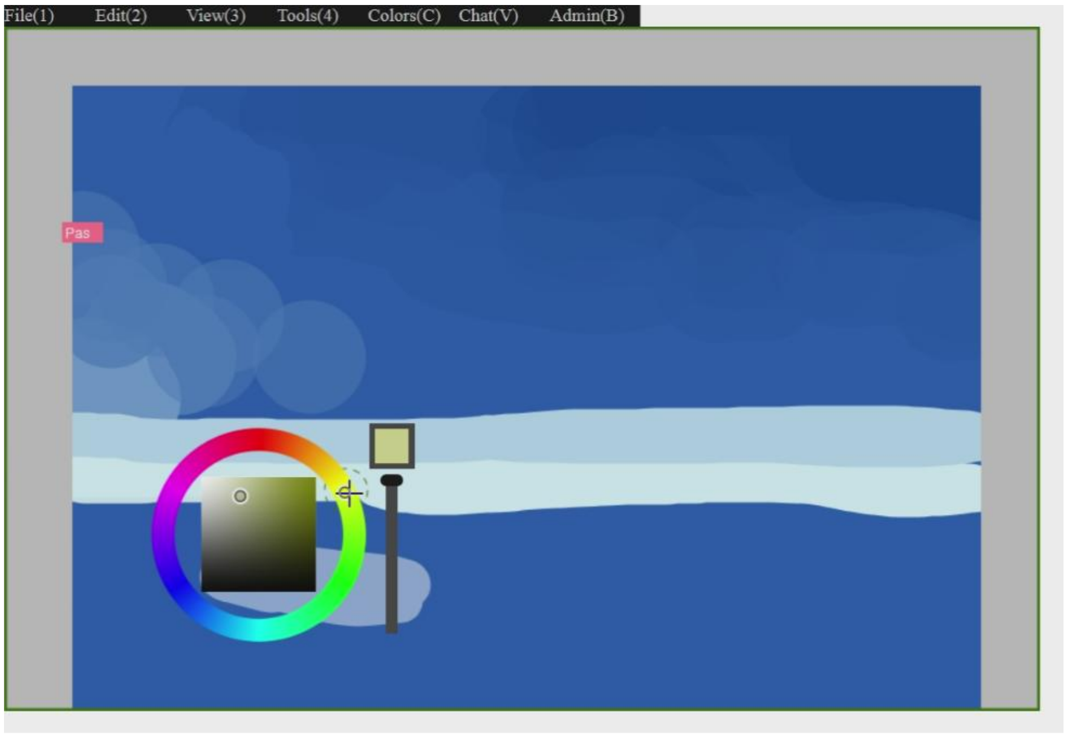
User2

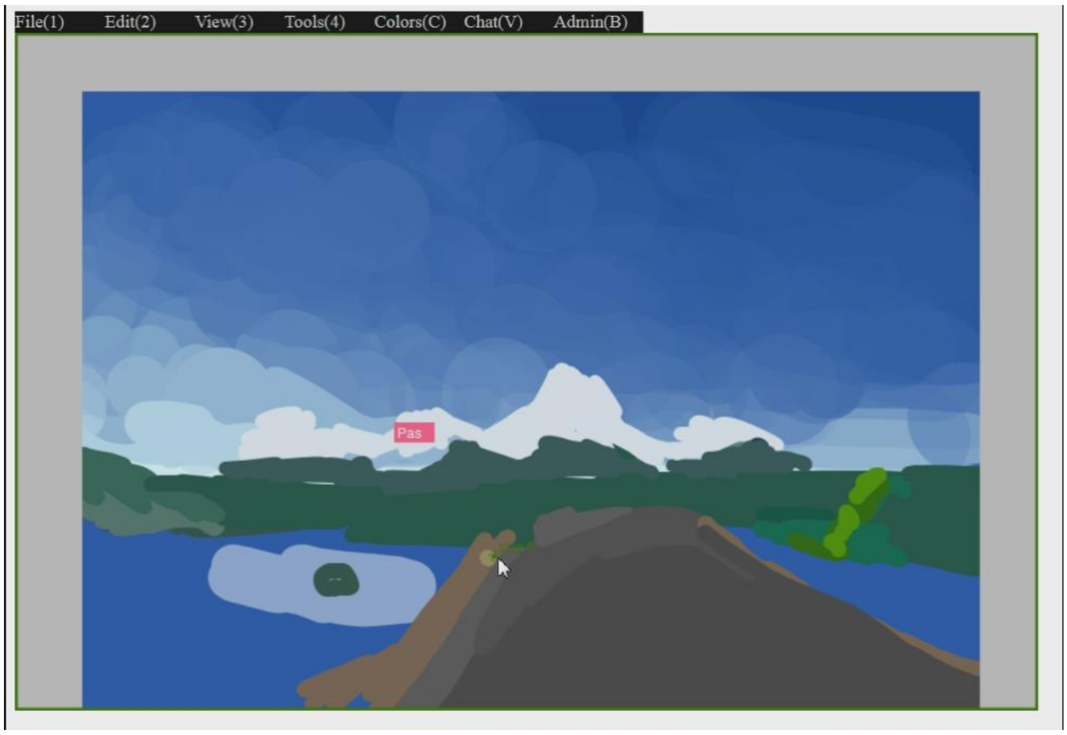
Below is a time lapse screen recording of a larger test session. Collaboration task given: Reference drawing from a given scenic photograph. Duration: 29 minutes.

Photograph that was referenced from.

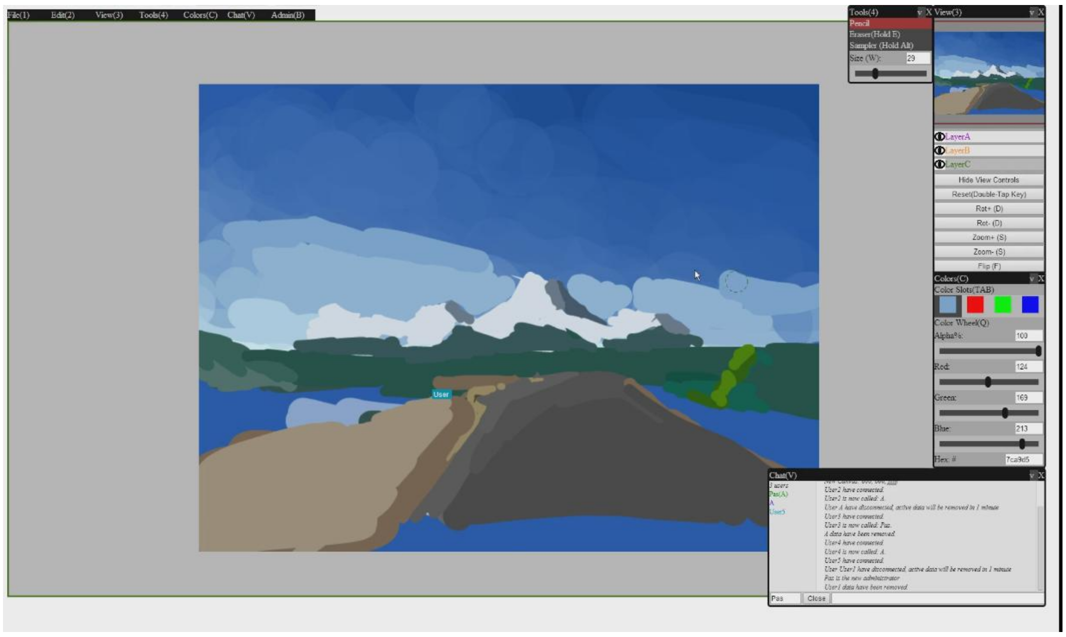

Result drawing after the given time.

Below is some screen recordings of one of the project artist during the session. One interesting observation was the project artist preferred not opening any panels.



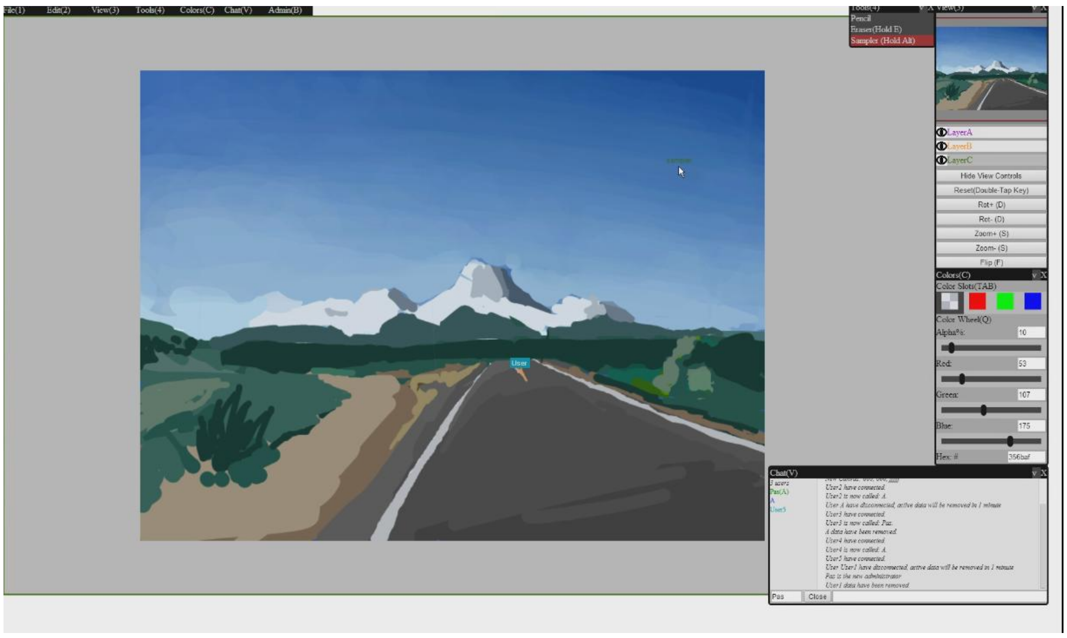


Below is the same collaboration from another project artist's screen recordings. This project artist had most frequently used panels all stacked to one side when drawing.



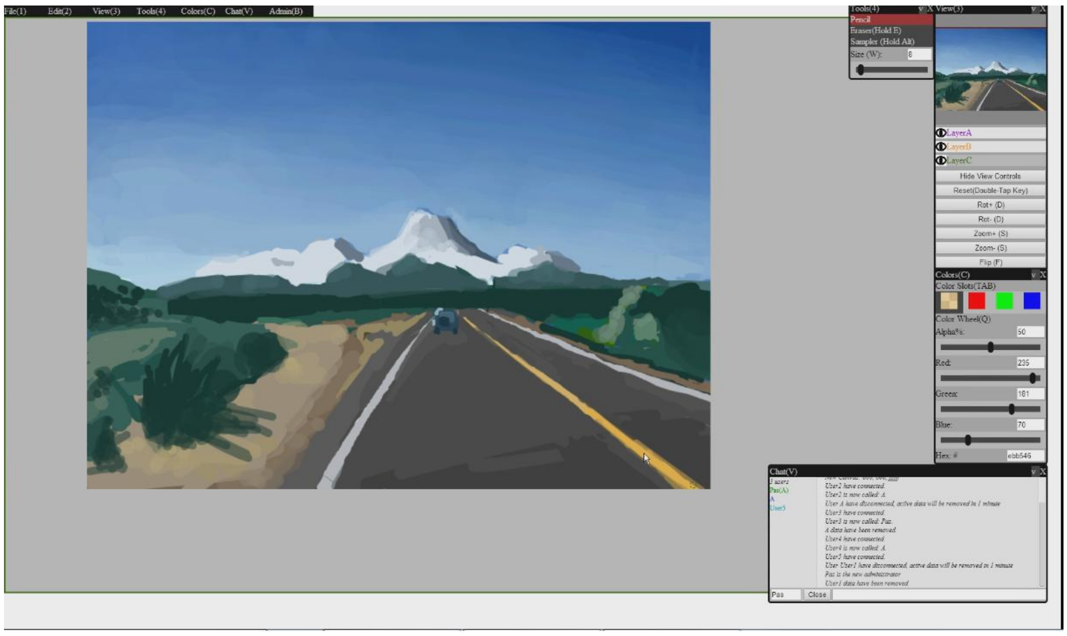
```
Chat(V)
3 users
Pas(A)
A
User5

User2 have connected.
User2 is now called: A.
User A have disconnected, active data will be removed in 1 minute
User3 have connected.
User3 is now called: Pas.
A data have been removed.
User4 have connected.
User4 is now called: A.
User5 have connected.
User User1 have disconnected, active data will be removed in 1 minute
Pas is the new administrator
User1 data have been removed.
```



```
Chat(V)
3 users
Pas(A)
A
User5

User2 have connected.
User2 is now called: A.
User A have disconnected, active data will be removed in 1 minute
User3 have connected.
User3 is now called: Pas.
A data have been removed.
User4 have connected.
User4 is now called: A.
User5 have connected.
User User1 have disconnected, active data will be removed in 1 minute
Pas is the new administrator
User1 data have been removed.
```



Size (W): 8
Alpha%: 50
Red: 235
Green: 181
Blue: 70
Hex: # ebb546

```
3 users
Pas(A)
A
User5
```

```
User2 have connected.
User2 is now called: A.
User A have disconnected, active data will be removed in 1 minute
User3 have connected.
User3 is now called: Pas.
A data have been removed.
User4 have connected.
User4 is now called: A.
User5 have connected.
User User1 have disconnected, active data will be removed in 1 minute
Pas is the new administrator
User1 data have been removed.
```

Pas

All requirements set out were satisfied by the final prototype. The prototype is built in HTML5 and Javascript, therefore no plug-ins required. A robust client-server network model was implemented with good network synchronization, this allowed multiple users to draw and chat in a shared environment. Users were able to connect at any time, and get the latest drawing. Limited steps of undo and redo was achieved from a hybrid vector-raster Buffer system. The prototype was able to support three layers and Mini-map support. Users could enter custom canvas sizes. All common tools listed were implemented. All hotkey interactions specified where achieved with additional features like radial menu. A room-administrative system was implemented to address negative social behaviours. The prototype back-end structure supports history playback because all drawing packets were saved in the server.

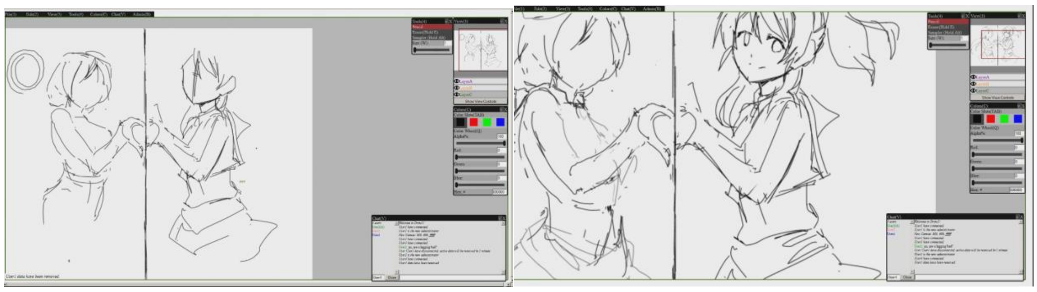
As a result, the project artists were pleased with the availability of many functions thought to be impossible in a collaborative environment. The application provided a much more comfortable environment compared to other systems because of canvas transformations and undo functions. Multiple layers allowed them to experiment and draft work without affecting the final drawing. The fear of committing to all actions as in other collaborative drawing applications has been greatly reduced. The simple hotkeys and interaction makes workflow more efficient and the application more enjoyable to use. Everything is working, smooth, and synchronized while under WAN connection.

Even though the project artists were impressed with the overall result, there are still many important issues that will need to be addressed before any public release or testing. Some issues discussed could be due to preferences of the project artists and some are overall issues that could impact on all users.

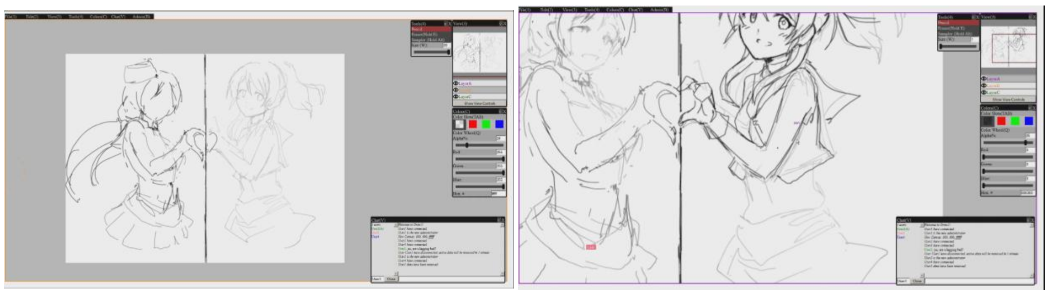
As previously said, HTML5 is still at an early stage of implementation in all browsers. Throughout project development, the system ran smoothly with no visual delays in Mozilla Firefox(version: 28.0). Internet Explorer(version: 11.0) also performs very well when tested. However Google Chrome(version: 33.0) and Apple Safari(5.1) showed serious signs of consistent visual performance delay. Opera(version: 20.0) also has similar consistent delay but to a lesser extent. These delays are not dependent on the number of actions, they can be observed from the start from doing canvas transformations. After some researching on various forums most web developers are suggesting it is mainly due the way which different browsers manage their resources. Different browsers have different priorities given to different functions and hence resources regarding HTML5 Canvas elements can be capped or restricted. The most likely cause of delay on some browsers are suggested to be storing and accessing large off-screen Canvas elements; this structure cannot be avoided in this project due to the requirements of layering and canvas transformation. Achieving more optimized code on the client side Javascript might reduce some performance cost, but the serious visual lag on Chrome and Safari suggests that major front-end structure changes or browser specific modifications are needed. It can be theorised that Chrome, Safari, and Opera have similar back-end management as do Firefox and Internet Explorer. It is unfortunate that the project could not fully achieve browser independence for these issues. Further experiments will be needed to pinpoint the causes and find solutions.

One notable issue with the Buffer system is the order of actions. Although, as the project artists expects, actions between different users do not often overlap, but there are still times where they do. The Buffer system is layered such a way that rasterized actions will always appear below undoable ones, and rasterization of users' actions are invoked by themselves. Users who perform more actions faster will rasterize actions more often. This means, even though order of events are synchronized across the network, the action orders might not be fully preserved because different users perform actions at different rate. This problem has caused some confusion during testing sessions; where one artist drew a large stroke but stop for a while, the other artist drew over the stroke with many small strokes eventually hitting the rasterization phase; the small strokes ended up on the other side of the large stroke unexpectedly. This is not a problem with network synchronization, as stated, order of events are synchronized across network by the server. The issue is in the design of the Buffer system currently not fully preserving action order. To fully preserve the order of actions globally, rasterization phase will need to be delayed until the older actions are rasterized first. A solution could be that whenever an action is to rasterized, it will be flagged. When the oldest action in the Buffer is ready to be rasterized, it will also check upwards and rasterize any other flagged actions. This method will risk one action blocking other actions, so a count-down timer should be used to force rasterize any blocking actions. Therefore when an action is ready to be rasterized and finds that the previous action is not, a timer of five seconds will start ticking for the action to be either undone and removed or be rasterized. As a result, this method will preserve all actions order but will force rasterize old actions slowly removing the undo history of the slowest user. Worst case scenario is the other artist is constantly performing five actions ahead of a slow user and forcing all the slow user's actions to rasterize every five seconds. After a discussion with the project artists, they said the timer would not be likely pose any problems because, as mentioned, people most often use undo for very recent mistakes, even in the worst case scenario, rarely does an artist take five seconds to realise they have made a mistake. The validity of this method could not be tested given the time of this project, but knowing the order of events are still synchronized, there should be no problem implementing this method.

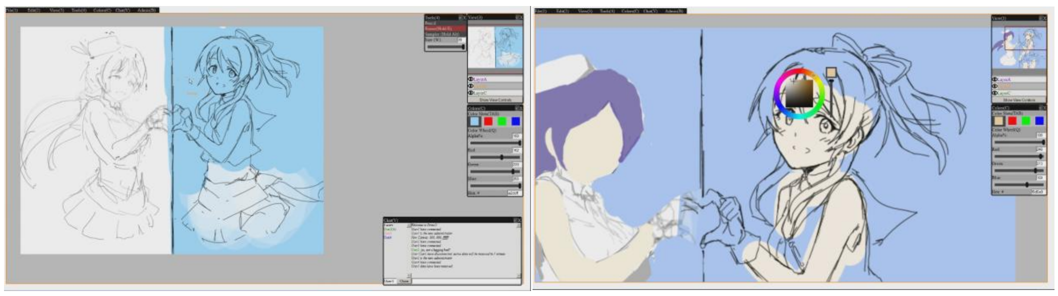
Below is a time lapse of screen recordings of another larger testing session. Task given: Draw a picture of two people side by side. Duration 50 minutes.



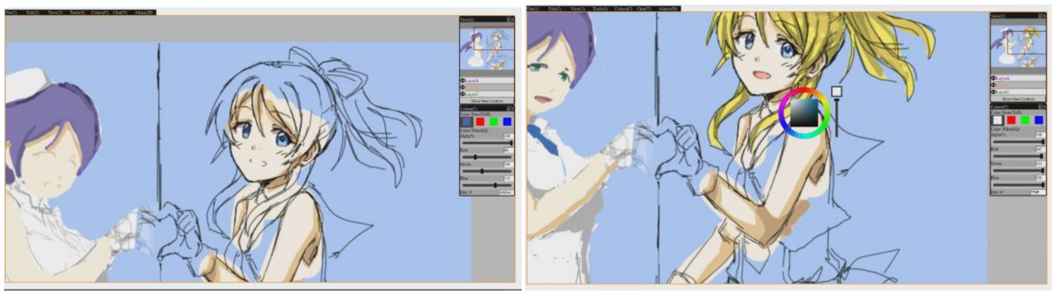
The project artists line drafts at the second layer.

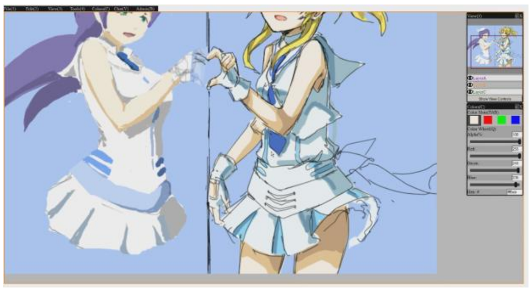
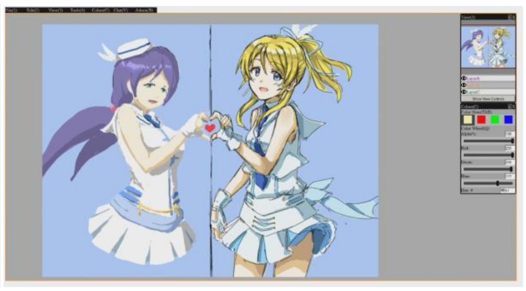


The right project artist painted a semi transparent white over his draft then started putting clean line work on the top layer . The left project artist decided to not have line work and started shading on the top layer.



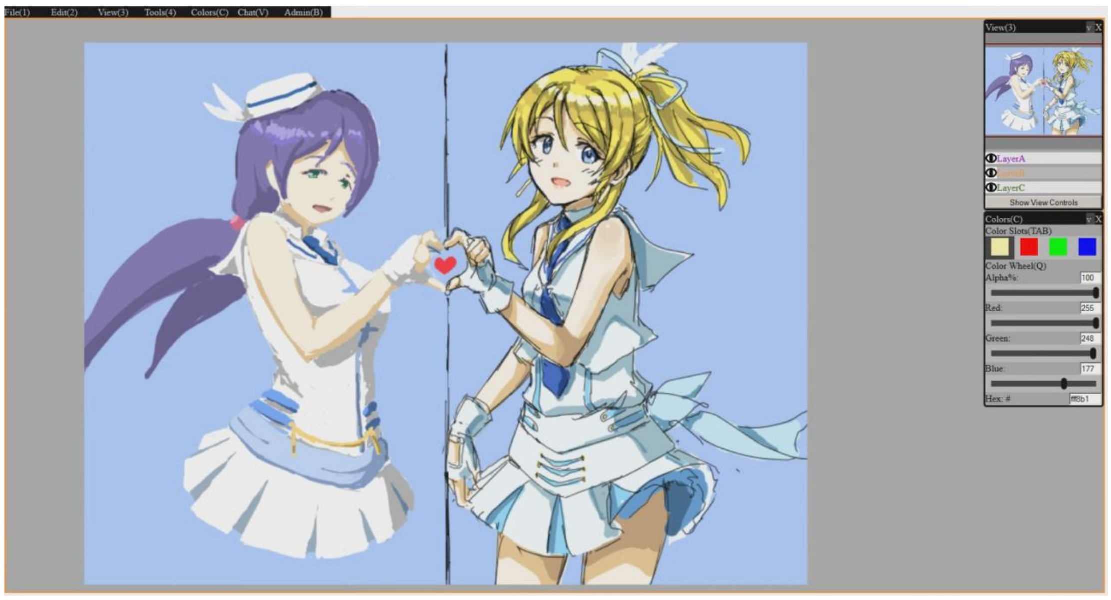
The right project artist put his shading on the second layer; under the clean line art.



Both project artists spent the rest of the time shading the drawing.

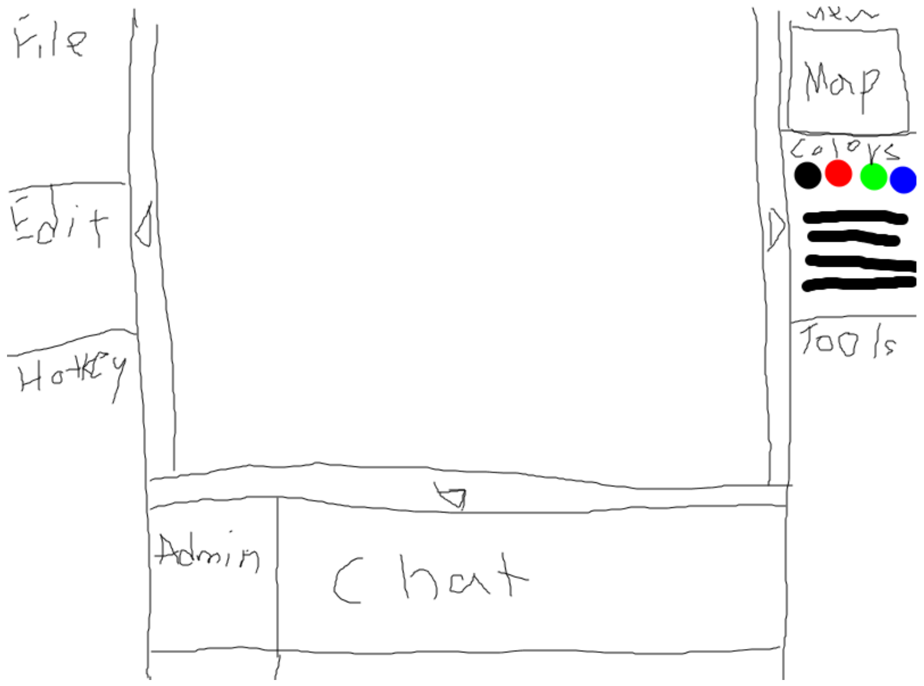
Finished drawing



Alpha%: 100
Red: 255
Green: 248
Blue: 177
Hex: # fff8b1

One issue discovered in testing was an issue with the overall interface approach. The floating panels allow much freedom and flexibility to the user's workspace, however they also takes time to setup at the beginning. A default start position for every panel can be used to reduce the time of setting up for new users but familiar users will need to close all unnecessary panels at the start, which is still a setup time cost. Unless an account system or cookies are used to save preferences, this setup time will always be there. Also the more panels exist, the more time it takes to setup. This lead to the conclusion that some panels could be merged. A good basis for merging seems to be grouping panels with similar frequency of use. For example the File, Edit, and Admin panels obviously all have low frequency of use, so they can all be built as one panel. Note, the reason that the Edit panel has low frequency of use is because users will naturally use hotkeys only. The tools and colour panel have similar frequency of use so can be merged together too. The project artist suggested that if there are only few panels then it would be better to stack them up on the sides of the window as collapsible menus. This is a similar approach to how Pixiv Chat dealt with their main menu; the whole menu can collapse and pop up.

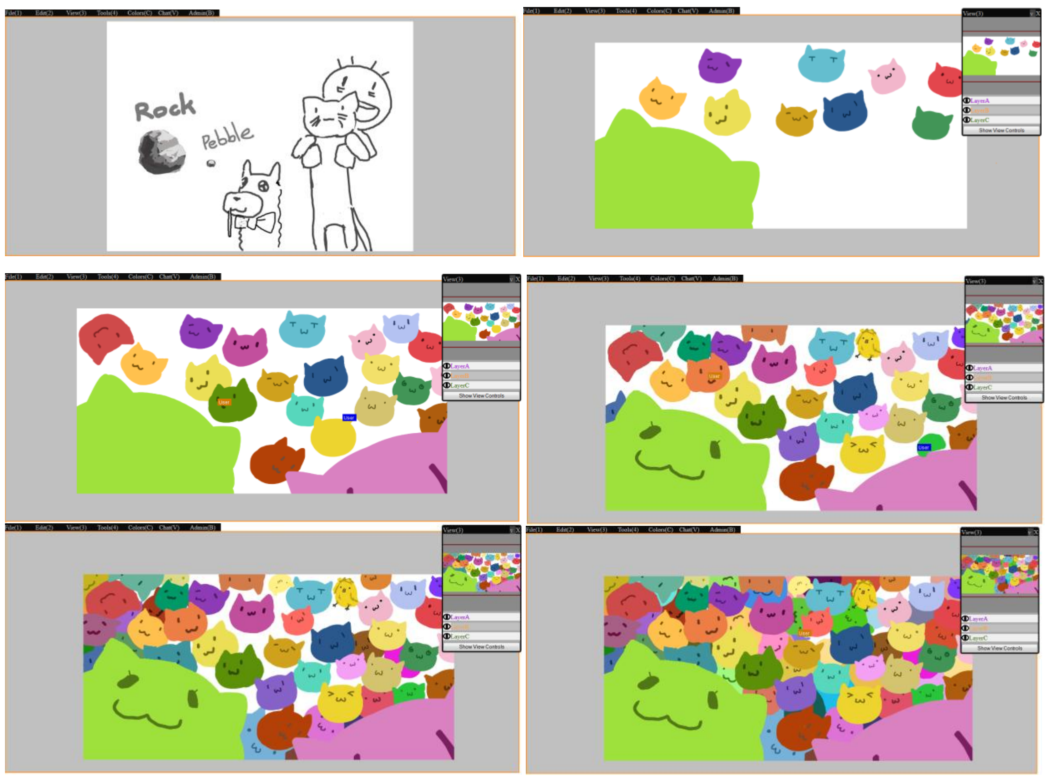
The new proposed solution to the interface is to have collapsible menus on three sides of the window. The bottom is the chat system and where the admin panel will be merged to the side of the chat system. The right hand side is where the most frequently used panels will be stacked; these are the mini-map, the colour selection, and the tools panel. The right hand side will be where all the least used panels are stacked; the File panel and the Edit panel. The least used area should also have a complete list of hotkeys available. By having this setup, all areas can be collapsed with one hotkey or individually with three. This reduces the amount of setup time to maximum of pressing one key to collapse all menus compared to the original of closing individual panels or moving each around. Of course like many HCI scenarios, user interface could risk coming down to purely a matter of preference rather than functionality. Because of this risk, this problem was not addressed after being discovered from testing.

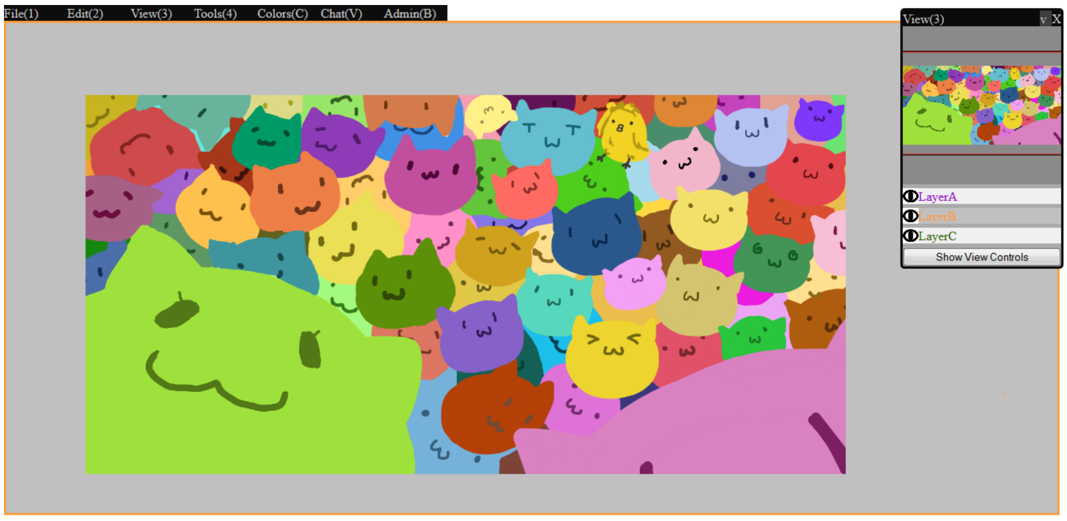


Proposed new interface layout/system drawn from the prototype by one of the project artists.

A minor issue with the current system is the stroke function of the HTML5 Canvas element. The given stroke function has an anti-alias effect by default and does not have an option to remove it. This became an issue to the artist because additional colours and blurriness are added at all edges. The only way to achieve a proper binary stroke is to create a custom brush engine. This could risk increasing the performance cost of the system but will be a necessary action.

Below is a time lapse screen recording of another larger test session. Task given: Draw a collaborative image of a cat/cats. Duration 20 minutes.




File(1)
Edit(2)
View(3)
Tools(4)
Colors(C)
Chat(V)
Admin(B)
View(3)
v
X
LayerA
LayerB
LayerC
Show View Controls

Chapter 11: Future Considerations

The result showed great potential for this application but it is still an early prototype. There are other notable features that could not be implemented with the given time. As mentioned in the user testing with project artists, there are issues that will require further experiments before any public release. This project could only focus on the drawing experience of online collaborative drawing, but as demonstrated in existing applications, drawing is only part of the overall experience. Collaborations rely heavily on social interaction and social services; the more functions available to make certain social interactions more convenient, the better the experience overall.

From the study of existing applications, many important social oriented features that were identified will have to be considered before public release. One feature that exists in many online collaborative drawing applications is the history playback. As evaluated, the structure of this prototype allows implementation of this feature being added. In order to implement this feature, a custom HTML5 Canvas-base player that can simulate instructions sent from the server will need to be built. However this will only achieve one directional playback. To support other time controls for playback, one proposal is to have the server save anchor points and images of a room periodically. This would allow playbacks to jump to different points of time by sending the raster image of that anchor and start streaming all packets from that time. This feature could also be further used as a roll-back function for rooms. As demonstrated in /f/lockdraw, the administrator could request roll-backs to reverse any unwanted events on the canvas.

A lobby system or room display system will need to be considered. This means the Node.js server needs to be modified to manage multiple rooms. This can be achieved by remembering which clients are connected to which rooms, so broadcasts are made to the right group of clients along with managing shared and property variables for each room. Display and interaction of the lobby system will be an important consideration. A robust system will need to show a considerable amount of information; canvas preview, tags, rules, descriptions, users…etc. Providing different room services will naturally be expected by users; public and private rooms, allow spectators or spectator chat; administrative right controls or moderations…etc. To achieve the best experience those functions would require a lot of large public sample experiments and feedback.

If this service is to be taken to public mass, considerations of network performance and maintenance will be an important issue. The current prototype uses encoded strings separated by commas to communicate. This is because it is easier to debug. However this means there are a lot of comma characters sent that are only used as separators. As pointed out by Niko’s Paint Chat community, network bandwidth usage is a serious matter and is expensive with real-time network applications. It is important to reduce the amount of data transferred to keep the cost low. The prototype communication method can be easily improved by defining a communication protocol and sending binary data. Many Instruction Headers, ID Numbers, or Tool Indices can be represented in a byte and co-ordinates can be represented in few more. This should cut the current prototype bandwidth usage by half.

Chapter 12: Conclusion

HTML5 is still a young technology but with many great potentials and possibilities. Its cross-platform and no-install nature will always make it preferable to other solutions. This project took the opportunity provided by the HTML5 Canvas element and Websocket to improve on online collaborative drawing experience. With help from two experienced artists, problems were accurately identified from an end-user perspective. The problems with current web-based collaborative drawing applications mainly came down to the need to bridge the core interaction gap with stand-alone drawing software. Functions like undo, free canvas transformation, or hotkeys did not really exist. Many functions could not previously be achieved due to various difficulties of networking applications and browser limitations. But with new opportunities given by the HTML5 Canvas element and Websockets, this project has strived to achieve these functions. The biggest achievement in this project is accomplishing a robust system that can achieve network synchronization and good performance with all the implemented features. Limited steps of undo was achieved by creating a vector-raster hybrid buffer system. The buffer system made use of advantages from both vector and raster graphics system; objects could be modified and re-drawn like in vector graphics, yet old objects would be rasterized to save processing time. As a result the Buffer system could perform efficiently in a web-browser, provide limited steps of undo, and could be expanded to support multiple interactive drawing layers. Network latency was hidden by implementing network prediction techniques with re-synchronization capabilities from a authoritative server.

This research used an incremental development methodology with the two project artists throughout the research; for both requirements and implementation, discussion and evaluation sessions were held to validate them. This relationship was particularly important in helping to make decisions where compromise between functionality and performance was required. With suggestions and examples given by the project artists, better hotkeys and interface interactions were developed in contrast to the basic buttons that most collaborative drawing applications provide. The prototype has floating panel interface, free canvas transformation, basic tools, two colour selection tools, a chat system, and a room-administrator system. The combination of all these features removed many of usage barriers that may have prevented new artists from participating in online drawing and provided new efficiency in the drawing experience.

While much was achieved on the technical aspects of the collaborative drawing experience, some socially focused functions were only lightly addressed with the given time of the project. As pointed out, HTML5 has not been fully finalised; different web browsers still have different implementations for each feature. This cause some performance issues in certain browsers; however there are browsers that can run this prototype with smooth performance. Hopefully as HTML5 becomes more standardized and computers become more powerful, all browsers will have a more satisfactory performance and result. Modifications and fixes will be required before public release, but the project artists were very pleased with the outcome of the project and hope to see it continue. The availability of functions that were originally thought to be impossible because none had existed in previous online collaborative drawing applications for a good past 4 years. The existence of numerous examples in online collaborative drawing applications clearly shows a demand and interest in this area of service. This project have demonstrated new possibilities and showed how to advance in collaborative web applications more than just small experiments. The knowledge and solutions provided hope to help future web-based collaborative drawing applications and HTML5 interactive networking applications to provide better functions and interactivity to the world.

The prototype is available at: http://drawtooltest-yh184.rhcloud.com/

This website will remain operational for at least one year. Use Firefox version 28 or above for best result.

References

AlRamahi, M., & Gramoll, K. (2004). Online Collaborative Drawing Board for Real-time Student-Instructor Interaction and Lecture Creation. *ASEE Conference.* Salt Lake City: The University of Oklahoma.

CELSYS Inc. (2014). *CLIP STUDIO PAINT*. Retrieved 04 09, 2014, from Clip Studio: http://www.clipstudio.net/en/

Chen, D., & Sun, C. (2001). Undoing Any Operation in Collaborative Graphics Editing. Brisbane, Australia: Griffith University.

CoSketch. (2014). *cosketch*. Retrieved 04 09, 2014, from cosketch: http://cosketch.com/

Google. (2014). *Chrome Experiments*. Retrieved 04 09, 2014, from Chrome Experiments: http://www.chromeexperiments.com/

Gutwin, C., Lippold, M., & Graham, T. C. (2011). Real-Time Groupware in the Browser: Testing the Performance of Web-Based Networking. *CSCW.* Saskatoon & Kingston: University of Saskatchewan & Queen's University.

iScribble. (2014). *iScribble*. Retrieved 04 09, 2014, from iScribble: http://www.iscribble.net/

Jones, K. (2011). *Chrome Experiments- "Paint With Me" by Kyle Jones*. Retrieved 04 09, 2014, from Chrome Experiments: http://www.chromeexperiments.com/detail/paint-with-me/?f=

Joynet. (2009). *node.js*. Retrieved 04 09, 2014, from node.js: http://nodejs.org/

Khronos Group. (2011). *WebGL Specification*. Retrieved 04 09, 2014, from Khronos Group: https://www.khronos.org/registry/webgl/specs/1.0/

Marion, C., & Jomier, J. (2012). Real-time Collaborative Scientific WebGL Visualization with WebSocket. *Web3D 2012.* Los Angeles, CA: Association for Computing Machinery, Inc.

Mixart New Media LLC. (2010). *Rate My Drawings - DrawChat*. Retrieved 04 09, 2014, from Rate My Drawings: http://www.ratemydrawings.com/chat/

Moelker, R. R., & Wijbrandi, W. E. (2012). HTML5 data visualization capabilities of mobile devices. *SC@RUG* (pp. 23-28). University of Groningen.

Mr.doob. (2010). *Chrome Experiments- "Multiuser Sketchpad" by Mr.doob*. Retrieved 04 09, 2014, from http://www.chromeexperiments.com/detail/multiuser-sketchpad/?f=

Niko. (2009). *Nikos Paintchat*. Retrieved 04 09, 2014, from Nikos Paintchat: http://pchat.mine.nu/

Pixiv. (2009, 12 05). *Pixiv Chat*. Retrieved 04 09, 2014, from Pixiv Chat: http://chat.pixiv.net/

Queeky. (2010). *Queeky's MultiDraw HTML5*. Retrieved 04 09, 2014, from Queeky: http://www.queeky.com/multidraw

Red Hat Enterprise. (2013). *Open Shift*. Retrieved 04 09, 2014, from Open Shift: https://www.openshift.com/

Sangiogi, U. B., Beuvens, F., & Vanderdonckt, J. (2012). User Interface Design by Collaborative Sketching. *In the Wild.* Newcastle, UK: Louvain Interaction Laboratory, Université catholique de Louvain.

Skycow. (2011). */f/lowckdraw*. Retrieved 04 09, 2014, from /f/lowckdraw: http://skycow.us/whiteboard.swf

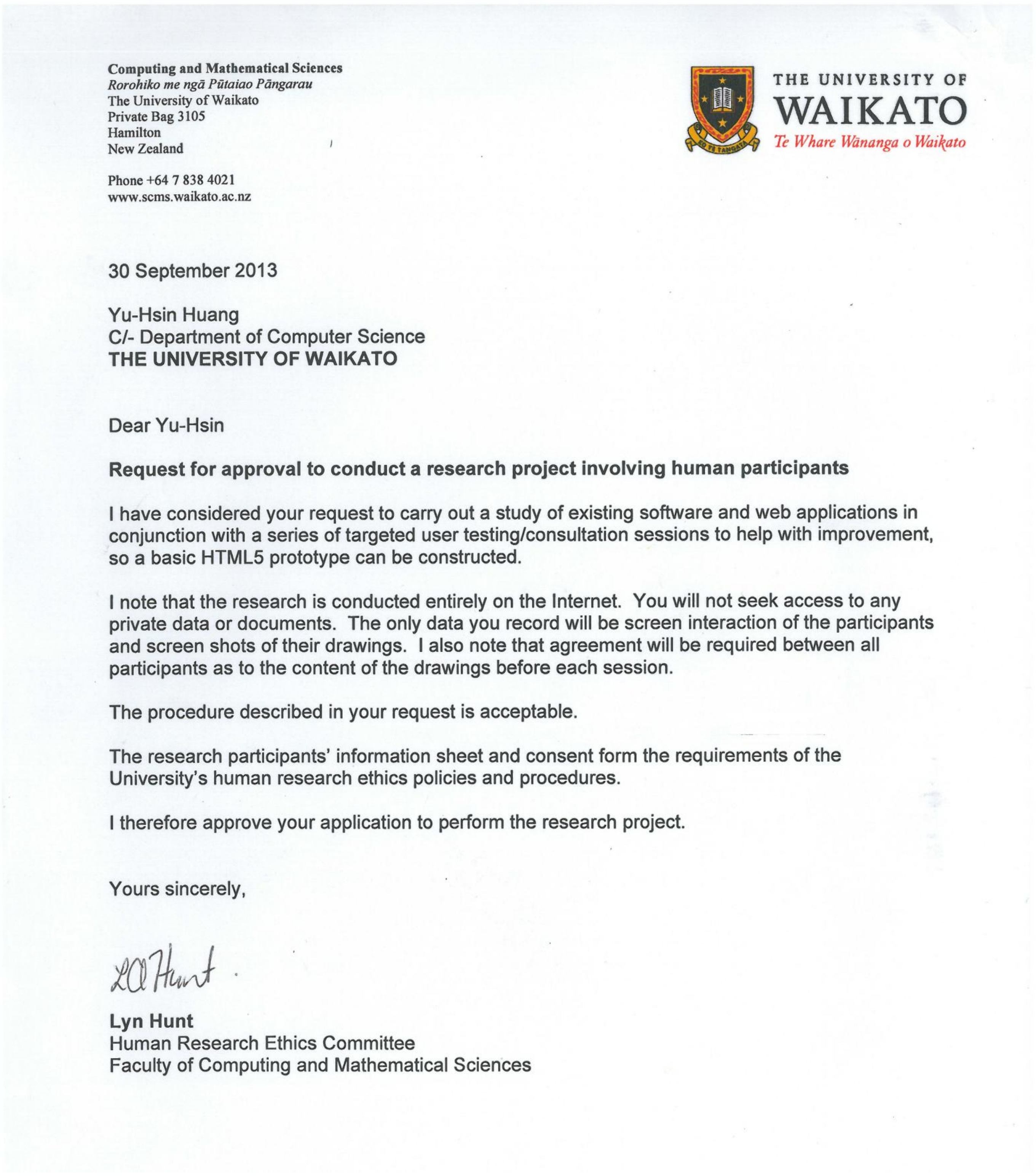
W3C. (2008). *HTML5*. Retrieved 04 09, 2014, from W3C: http://www.w3.org/TR/html5/

Wang, X., Bu, J., & Chen, C. (2002). Achieving Undo in Bitmap-based Collaborative Graphics. *CSCW.* New Orleans, Louisiana, USA: Zhejiang University, Hangzhou, Zhejiang, China.

WDF. (2010). *DoodleToo*. Retrieved 04 09, 2014, from DoodleToo: http://www.doodletoo.com/

Appendix A

University of Waikato Ethics Consent Approval



Computing and Mathematical Sciences
Rorohiko me ngā Pūtaiao Pāngarau
The University of Waikato
Private Bag 3105
Hamilton
New Zealand

Phone +64 7 838 4021
www.scms.waikato.ac.nz

THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato

30 September 2013

Yu-Hsin Huang
C/- Department of Computer Science
THE UNIVERSITY OF WAIKATO

Dear Yu-Hsin

Request for approval to conduct a research project involving human participants

I have considered your request to carry out a study of existing software and web applications in conjunction with a series of targeted user testing/consultation sessions to help with improvement, so a basic HTML5 prototype can be constructed.

I note that the research is conducted entirely on the Internet. You will not seek access to any private data or documents. The only data you record will be screen interaction of the participants and screen shots of their drawings. I also note that agreement will be required between all participants as to the content of the drawings before each session.

The procedure described in your request is acceptable.

The research participants' information sheet and consent form the requirements of the University's human research ethics policies and procedures.

I therefore approve your application to perform the research project.

Yours sincerely,

[illegible]

Lyn Hunt
Human Research Ethics Committee
Faculty of Computing and Mathematical Sciences