Digital PaintChat: Examining and Expanding Collaborative Tools for Digital Artists

by Helen Jiang

Boston College Honors Thesis
Advisor: Professor William Ames
May 8, 2012

!

!

Contents

!

!

Abstract

The digital world has revolutionized virtually every aspect of peoples' lives. Many professional illustrators have begun to use digital tools in order to simplify their drawing process and make it more efficient. There are many different software programs that artists use, each fitted to meet different needs, such as photo manipulation, painting, or animation. Although digital art is constantly evolving and expanding, and there is little research on how artists interact with digital media.

Communication is one of the areas in which technology has had the most profound change. People from anywhere in the world have the ability to contact each other at a moment's notice. This reality has lead to new, fruitful collaborations in a variety of fields. Thus far, there are no fully-functional artist tools that enable direct communication between artists. My thesis involves the planning and implementation of such a program.

I first conducted a digital arts survey to gather data on how current digital artists interact with the programs they are using, the way they use tools that are common among all digital art software programs, as well as the shortcomings of these tools and digital art in general. The survey was answered by both amateur and professional artists from online art communities, the majority of whom have been using art programs for over four years. Afterwards, I began programming a basic drawing program based on the results of the survey, and added networking capabilities.

!

1. Introduction

1.1 What is Digital Art?

Digital Art is defined as any art that was created with the aid of a computer. There are a multitude of types of digital art, such as Graphic Design and 3D Modeling. The focus of this thesis is Digital Illustration, which is defined as the use of digital tools under direct manipulation of the artist to produce 2D images, usually through a pointing device such as a mouse or a drawing tablet. In simpler terms, an artist uses a mouse or tablet to draw.

There are many applications that have been developed to aid artists in digital illustration. They contain a virtual canvas with a large amount of tools and instruments, some meant to mimic the purpose of tools that traditional artists use, and others that do not exist outside of the computer. The digital-only tools are what give digital art a distinct look and feel from traditional art. Unfortunately, there is very little academic research on digital illustration.

1.2 Drawing Tablet

A drawing tablet is an pointing input device that allows users to hand-draw images onto a computer. It usually consists of a surface to draw upon and a special drawing pen. Coordinates on the drawing surface directly correspond to coordinates on the computer screen. Thus, unlike a mouse, the pen location is not calculated relatively to the original location of the cursor. Aside from being able to transpose brush strokes easily onto a digital canvas, the drawing tablet's other most important feature is that of pen pressure detection. With pen pressure enabled, artists can simulate the effects of light versus heavy strokes on a canvas.

1.3 Digital Art Applications

graphics meant to convey information, such as maps and icons. Vector images are also not fixed in resolution, as their mathematical nature allows the artist to zoom without constraints. A well-known example of a vector program is Adobe Illustrator. The vector-based nature of Adobe Illustrator makes is difficult for it to be used as a painting program.

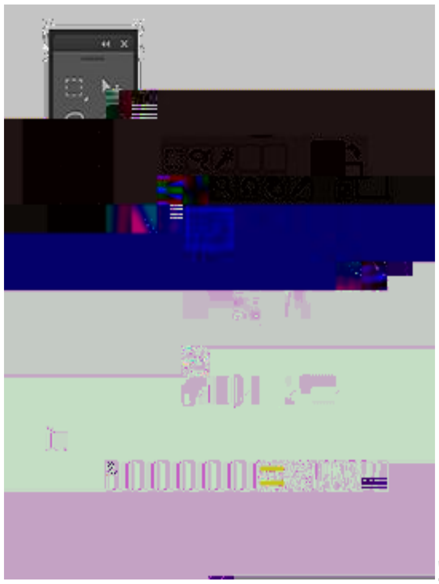
1.4 Collaboration

The idea of a online art collaboration for digital artists is not a new one. In all of its implementations, however, many crucial features are lacking. There are many collaborative projects where different artists edit the same canvas. Most of these projects do not support real-time editing where artists can work at the same time, and those that do only support the most simple tools.

The most commonly used "PaintChat" applications where artists draw together over the internet are through web browsers. Some examples of these are iScribble and Japanese application ShiPainter. Because they are browser-based tools are rather primitive and there are limitations based on browser type. None of the web-browser PaintChats by themselves support pen pressure, and the add-ons that need to be installed in order for pressure to be enabled on certain drawing boards do not universally work on all platforms and browsers.

1.5 Tools

Though each digital illustration application features different tools, there are many tools that are shared between all of them. Applications may implement the tools using different algorithms, which result in a different look and feel.


!
Figure 2: Comparison of Adobe Photoshop toolbar (left) with SAI PaintTool toolbar (right)

In the Adobe Photoshop toolbar, core tools are shown, sectioned off by the type of utility they offered. If a tool is clicked and held, a few options for variations of those tools are offered. For example, if the erase button is clicked and held, the user will be able to access different types of erase, such as the Background Eraser tool and the Magic Eraser Tool. In the SAI Paint Tool toolbar, tools are separated between select tools, manipulation of canvas view, and editing tools. The editing tools section has empty slots which allows users to create their own custom tools. These custom tools are basically the default tools with certain changed settings that are saved and renamed to a new tool.

Tools were important in the survey that I conducted as the first part of my thesis. The first section of the survey was aimed at evaluating artists' relationship with the tools that were common among all digital art programs (brush, eraser, paint bucket, zoom, etc).

1.6 Tool Customization

All digital art programs, to some degree, allow artists to customize tools. In the most simplistic programs with tool customization, such as Microsoft Paint, the user is able to

!

manipulate the size of the brush tool. In the more complex programs that I am

!

2. The Survey

'!

!

Figure 4

(!

!

- A popular category of suggestions involved color, and tools to help artists choose color.

!

Many times artists listed specific features of tools such as brushes, selection tools, and layers that they used the most often. This presented a good perspective on how wide the scope of each of these tools should be.

Other notable tools were listed as common used by multiple artists:

- The color wheel was once again frequently mentioned, as artists work with color often and want to be able to switch between and save colors palettes easily
- Move and Zoom tools, in addition to the afore-mentioned flip and rotate, were required for flexibility in the way the artist viewed their canvas.

2.2 Customizability

In this section, there was just one question to rank the importance of customizability from seven commonly customized tools: Brush, Eraser, Color Palette, History, Fill Bucket, Layer and Zoom.

The results of this section were very simple; of the seven tools, five of them (Brush, Eraser, Color Palette, Layer, and Zoom) scored above three on a scale of one to five. This is especially interesting in the case of the Zoom tool, which usually is not highly customizable in digital art programs.

Figure 5: With an average score of 3, there was no consensus on the importance of customizability for the History tool.

The two that fell under three on the scale were History and Fill Bucket. I had considered making history more customizable as a new tool to introduce to artist, but this result left me with

" "!

" "!

!

3. Programming SimplePaint

3.1 Java

I chose to program the application in Java for its ability to run on any operating system. Many programs were limited to one operating system, which many artists expressed frustration with in the Digital Art Tools survey. An example of this would be SAI PaintTool, which was confirmed by developers to only be available on the Windows operating system. In response to SAI PaintTool, another company developed Clip PaintLab, which was made to work only on Mac OS. In addition to this consideration, Java was also the programming language that I was most familiar with. It has extensive graphics and networking capabilities in its libraries already.

3.2 Program Organization

called, the three layers are all drawn, in the order of backgroundLayer, baseLayer, and strokeLayer on the very top.

In my Program I added a PenListener from the JPen library. Although at first I also added a MouseListener, I eventually used the penButtonEvent method from the PenListener Interface to detect and differentiate mouse clicks from pen clicks. Because input from the mouse only included coordinates while input from the pen also included pressure and tilt, I had to program them separately. The penLevelEvent method from the PenListener Interface detects and runs when the pen is in close contact with the surface of the drawing tablet. This includes when the pen is hovering slightly above the surface as well as when the pen is touching the surface. To

3.4 Pen Pressure

Pen pressure enables artists to create more realistic strokes with their drawing tablets. One of the biggest complaints of current "paintchat" networking applications is the incompatibility with pen pressure. I searched for a java library that would help detect pen pressure from common tablets such as the Wacom Intuos 5 Tablet that I used. On the website called SourceForge.com, I found a library called JPen that could access drawing tablets and pointing devices using Java 5. It includes event and listener architecture. Device access is implemented through providers and conains providers for Linux, Windows, Mac OS X, and the java system mouse.

tilt, I made sure to make an extra constructor that would store those initial values. The instance of the operation is initialized when the pen first touches the tablet surface (within penButtonEvent method from the PenListener Interface). When the pen is dragged, the x-coordinate, y-coordinate, pen pressure, and tilt are continuously passed into the instance of the operation (within the penLevelEvent method from the PenListener Interface). A method within the operation is then called to continuously draw and repaint, which achieves the effect of varying pen pressure. For a brush stroke, pressure was multiplied against a base size and opacity. Since pressure was a double from 0 to 1, the base size and opacity were in fact the maximum size and opacity available and could only be achieved if the user pressed down with maximum pressure. Though this was not coded into the user interface, it is possible to make either or both size and opacity static if the user wishes to do so.

!

3.5 Operations

Features were divided into two categories those that would affect the image and

3.6 Compositing

In order to achieve some of the effects I desired in this program, I had to sometimes manipulate the compositing of the baseLayer and the strokeLayer, especially in manipulating the alpha values to change opacity. I will first give a brief introduction into how compositing works.

Composites define how two inputs are blended together mathematically. In Java, the AlphaComposite class supports standard Porter-Duff compositing rules, which were fully developed by Porter and Duff in a 1984 paper.

Figure 6: Illustrations the over, in, out, atop, and xor operations outlined by Porter & Duff in the 1984 paper

Because I had many specific blending properties in mind, there were times when the

Figure 7: A diagram illustrating Bresenham's line algorithm

ape of the brush is drawn there and the locations are stored to support the vector-based backing of the program.

At first, the stroke was drawn directly onto the BufferedImage that contained the rest of the image. The effect that this created was undesirable, however   because of the default SRC_OVER compositing mode, if the stroke was not at full opaqueness (alpha = 1) areas of intersection would visibly show the overlap. This poses a problem for artists because the effect gives artists less control over the opacity of the image.

Figure 8: Self-Intersecting BrushStroke when drawn on baseLayer BufferedImage of SRC_OVER composite

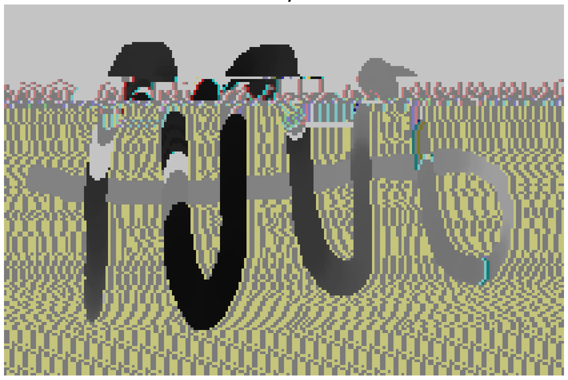
For example, if an artist attempted to cover a larger amount of area with a single stroke

!

achieve this effect, I had to create a custom composite called MaxAlphaComposite. MaxAlphaComposite is identical to the SRC Composite in all ways except for how the alpha value is blended. Instead of automatically taking the alpha of the SRC, it compares alpha of the SRC to the alpha of the DST_IN and takes the larger one. MaxAlphaComposite achieved the desired effect, as shown in the figure below.

!"#$%&#'()*+(,-,.(/0"12'()*+(3,!"#$45'()*+(67,

,



!"#$%&'()*'+',"-"./%',0%12&'01'03/0'14'!"#$%&'56'7%/89'19'03&',0%12&:/;&%'8"03'</=+.>3/?1->,"0&',

,

At the end of the stroke, when the pen or mouse is released, the strokeLayer is drawn onto the baseLayer. Because the baseLayer has composite SRC_OVER, this preserves the overlap quality between the new stroke and the rest of the image. After the two BufferedImages are merged, the strokeLayer is cleared. The brushStroke Operation is then appended to the History array and written to the ObjectOutputStream if networking is on.

3.8 EraseStroke

The EraseStroke Operation is much simpler than the BrushStroke, in that it does not require use of the strokeLayer or any custom composites. The EraseStroke also uses

still contains shape as well as size, opacity, and tilt.

!

When the user chooses the erase tool, the composite of the baseLayer is changed from SRC_OVER to DST_IN. When pixels in the source and destination overlap in -In compositing mode, the alpha form the source is applied to the destination pixels in the overlapping area. Because this is a little bit backwards from what EraseAlpha, the opacity value is flipped before being applied so that low pressure from the pen would generate a higher alpha and thus only erase a little while high pressure from the pen would generate a lower alpha and erase more. Once another tool is chosen by the user, the baseLayer composite will be changed back to the default SRC_OVER or to whatever composite is appropriate for the next Operation.

A slight problem exists with this implementation of EraseStroke. Since the erase is happening directly to the baseLayer, and since many times the parts of the EraseStroke being drawn overlap, opacity is hard to control. Usually, even at low pen pressure, because of the amount of overlaps, the EraseStroke appears to erase to a much lower opacity then expected. An easy fix has been implemented by simply reducing the effect pen pressure has on the alpha value. A formal fix AlphaEraseStroke was attempted, but still has some bugs. As of writing this thesis, only EraseStroke is implemented.

3.9 AlphaEraseStroke

AlphaEraseStroke was an attempt to use the strokeLayer to draw out eraseStrokes rather than directly erasing onto the baseLayer, similar to the way brushStroke works. In order to do this, lengthy steps had to be taken. AlphaEraseStroke lets you choose opacity and then remains the same opacity throughout the stroke, without varying by pen pressure. Size, however, still changes with pen pressure.

Within startOperation method, which is only called when the pen first hits the surface of the tablet, several preparatory steps have to taken. First, a copy of the baseLayer is created and the opacity of the entire copy is reduced by the opacity value of the entire stroke. The purpose of

drawing out. While it is possible to show the stroke all at once after it is finished, there is no direct way to show the stroke as the pen is dragged across the canvas because the information would have to come from the baseLayer and draw upon the strokeLayer. In an attempt to work around this, a custom composite called EraseAlphaComposite was made to read in an image as a reference raster. The baseLayerCopy is set as the reference raster, and instead of outputting a blending of SRC and DST_IN, it is a combination of SRC and the reference raster. It should simply output the pixels of the reference raster at the location of the SRC drawn.

After the AlphaEraseComposite is set, the draw method operates on the strokeLayer in a similar fashion to eraseStroke and brushStroke. At the end of the stroke, the strokeLayer is merged onto the baseLayer. In order for the merge to work properly, a custom composite once

!

3.10 History & Undo

The History class contains an ArrayList of type Operation, which keeps track of all changes made to the canvas. In order to undo, the following steps are taken:

```
,       +;<=>?@A;<='?B.>CBA(<="=B*067,
        D)B(?2(EC(<067,
        F>?,0+;<=>?@A;<=6,G,
,       ,       =B.*,-,+;<=>?@A;<='HB="=B*0;67,
,       ,       =B.*'?BI>0<=?>JBA(@B?3,K(<BA(@B?67,
,       ,       .B?HB"=?>JBLEIM(<B067,
,       ,       D)B(?"=?>JBA(@B?067,
,       N,
,       ?B*(;E=067
```

The undo function essentially removes the last Operation, clears the canvas, and redraws everything by going through all the Operations in the history list. This can become slow if many steps have been taking. A typical digital painting can contain thousands of strokes. In order to speed up the process, every so often an Operation in the history list will be a keyframe, which will have a saved copy of the baseLayer(s) so that the redraws can start from that point rather than the very beginning. The user would be able to set the frequency of keyframes so that they could better customize the program to fit the needs of their computer.

3.11 Networking

The program currently requires one computer to run the server separately. The IP address of the computer must then be entered into any clients that wish to connect to the server. The server controls one socket and blocks for incoming input. The server keeps a synchronized list of all clients. Clients connected to the server write Operations to the objectOutputStream, which the server then sends to all clients other than itself to avoid redrawing over itself. When clients

!

A separate History needs to be maintained to keep track of incoming Operations versus Operations that the user has run from their client canvas. Two different layers type should also be created   one where both users can edit upon one layer, and another that is only editable by the owner of the layer (but visible on other clients).

In the future it may be possible to make the server run continuously on web and let people connect more easily through that. Currently, only computers within the same network can connect because of firewall and router issues.

!

5. References

[1] Sotiris P. Christodoulou, Georgios D. Styliaras: Digital art 2.0: art meets web 2.0 trend. DIMEA 2008: 158-165

[2] Porter, Thomas, and Tom Duff. "Compositing Digital Images." *ACM SIGGRAPH Computer Graphics* 18.3 (1984): 253-59. Print.

[3] Du, Weiming, Zhongyang Li, and Qian Gao. "Analysis of the Interaction between Digital Art and Traditional Art." *2010 International Conference on Networking and Digital Society* (2010): 441-43. Print

[4] Bresenham, J. E. (1 January 1965). "Algorithm for computer control of a digital plotter". *IBM Systems Journal* **4** (1): 25 30. doi:10.1147/sj.41.0025