# Simulating Chinese Brush Painting:

# The Parametric Hairy Brush

Ross B. Girshick

Senior Honors Thesis

for the

Department of Computer Science

Brandeis University

2004

# Contents

# List of Figures

# Abstract

This paper proposes a new geometric model for simulating the structure and behavior of brushes used in traditional Chinese painting. The geometric model uses a novel parametric representation of the internal structure of a real brush coupled with an intuitive approximation of soft brush dynamics suitable for use in a real-time painting system. The goal of this model is to allow a digital artist to produce realistic Chinese brush painting by capturing the minute details of a real brush.

# 1 Introduction

In Chinese brush painting each stroke exists on its own as a piece of art. The distinct characteristics of a brush stroke come from the combination of a painter's skill with the unique physical properties of the paper, ink, and brush. As an artist paints the brush bristles deposit ink and water on the paper, the ink and water diffuse through the paper, and the bristles deform due to external forces.

Digitizing the painting process requires a solution to three distinct problems: modeling ink and water diffusion in paper, modeling soft brush dynamics, and creating a stroke input method. Recent research demonstrates that solutions to the latter two problems are more fundamental for capturing the style of Chinese brush painting [CT02]. Until recently, however, nearly all research has focused exclusively on the first problem. The Parametric Hairy Brush (PHB), described in this paper, is a new brush model designed to realistically simulate the Chinese brush. The two key properties that distinguish the PHB from previous brush models are a simple and flexible geometric representation of the internal structure of a real brush and the ability to simulate a wide range of stylistic effects "out of the box" without manual configuration of esoteric parameters.

Before introducing the PHB design this paper offers a brief survey of the most important research on simulating brush painting. The survey provides a guide through previous attempts to solve the aforementioned problems and motivates the development of the PHB. Following the survey, the PHB design is described in detail along with a discussion of major design decisions and implementation issues. Finally, the PHB is compared against the two other leading Chinese brush painting systems in terms of aesthetic results and brush model properties.

# 2 Related Work

This section, through the presentation of keystone research, shows the evolution of brush painting systems. The significant findings of each work are demonstrated to illustrate how they shaped the direction of future research and ultimately motivated the PHB. It also describes in detail the three problems encountered in brush painting simulation: modeling the brush, modeling ink diffusion, and handling user input.

## 2.1 Hairy Brushes [Str86]

In 1986, Steve Strassmann of the MIT Media Laboratory published the seminal work "Hairy Brushes." This paper, based on his masters' thesis, reports on a novel system designed for simulating Japanese *sumi-e* ink and brush painting. Characterized by its simplicity, *sumi-e* paintings consist of a few well placed ink strokes on rice paper. This simplicity made it an ideal choice for simulation on the CPU performance bound machines of the mid 1980's.

Strassmann's solution uses a collection of abstractions – the brush, the stroke, the dip, and the paper – to divide the painting system into logically separate components. The abstractions were chosen to provide a flexible framework for experimentation. For instance, instead of painting directly to pixels on the frame buffer, the paper object could composite an ink value with paper texture map value before rendering the final pixel to the frame buffer. The high degree of flexibility and correspondingly rich experimental environment made Strassmann's work a model for future painting systems.

In order to employ the abstractions in a painting system, Strassmann implemented the objects in Zetalisp on one of the Media Lab's LISP machines. In this implementation painting is a two step, non-interactive process: first user input is collected to define a brush stroke and then the program renders the stroke.

A stroke object is created by using a mouse to specify control points that define a

cubic spline. When a control point is selected the user specifies the brush pressure to be applied at that point using keyboard bindings. After defining a spline, it is approximated as a series of line segments. At each joining of two line segments a new perpendicular line segment is defined. It is bisected by stroke path and its length is proportional to the pressure value interpolated at the joint. These lateral segments are connected to each other at their end points – generating a series of quadrilaterals along the spline. These quadrilaterals define the stroke body and are stored in the stroke object for later use.

After preparing the stroke object the user interaction phase is over and it is ready to be stroked by the brush object. The brush is represented as a simple linear array of bristles, each connected to an independent ink supply. Using heuristic update rules, the position of each bristle and the state of its ink supply evolve over the length of the stoke.

Each bristle deposits ink onto the paper object if the bristle's ink supply is not depleted and the bristle contacts the paper with sufficient pressure. Ink deposition is accomplished by sending a message from the brush object to the paper object specifying which position is affected as well as how much ink to deposit. The paper object can then apply any desired preprocessing, such as composition with a texture map, before writing the final pixel data to the frame-buffer.

In order to create strokes with differing stylistic effects, the dip object is used to control which bristles are supplied with ink and with how much they are supplied. For example, to create an effect such as a stroke that becomes drier over time, the dip object can apportion a relatively small quantity of ink to a large percentage of the bristles.

Strassmann's was able to account for many of the interesting characteristics seen in simple *sumi-e* style paintings. The overall process of painting, however, is far from what a real artist experiences. Each stroke has to be created off-line as a series of specified points and pressures. After specifying these properties a typical stroke required one to two minutes to render. This performance issue prevented implementing a real-time painting system and or a more natural method of painting strokes.

The lasting importance of Strassmann's work is the architecture he defined for developing brush painting systems. Virtually every paper on brush painting since the publication of "Hairy Brushes" cites his work and employs an architecture heavily influenced by Strassmann's abstractions.

## 2.2 The Introduction of a 3D Brush [Lee01, Lee99]

After Strassmann's "Hairy Brushes" there was a long pause in research on realistic brush models. During this pause, non-photorealistic rendering research encompassed other problems, the most germane of which was simulated ink diffusion. Thirteen years after "Hairy Brushes," Jintae Lee of the University of Aizu, Japan, connected the new research in ink diffusion with a Strassmann architecture painting system.

The brush model Lee proposed was the first evolutionary step away from Strassmann's inherently two dimensional model towards a dynamics three dimension model of a soft brush. The argument behind a 3D brush model is threefold: (1) it provides the artist with visual feedback that is similar to the artist's real workspace, (2) it is possible to capture the unique characteristics of brush painting with a model that closely mimics the physical behavior of a soft brush, and (3) it provides a more flexible framework for creating different types of brushes by changing physical parameters such a bristle stiffness, length, and absorption.

The model that Lee proposed in 1999 uses a sparse array of bristles (about 10), represented as polylines, attached to a cylindrical brush handle. Each joint in a polyline bristle represents a hinge. The hinges are incorporated into a spring-mass physics model that is solved using numerical integration in real time as the brush is moved by user input with a mouse. The number of bristles is limited because the time consuming numerical integration prevents the use of a more realistic number of bristles.

The limited number of bristles introduces a difficulty in mapping the output of the

brush model to a stroke. To overcome the sparse bristle issue strokes are defined using one of two available modes: outline definition and bristle foot printing. The outline definition mode is very similar to Strassmann's stroke area definition model. The last bristles in contact with the paper on each side of the array are used to generate a bounding area on the paper as the brush moves over time. The interior of the stroke is then rendered using a sophisticated algorithm for modeling ink diffusion. In the bristle foot printing mode, the area of the paper directly contacted by each bristle is used to decide where to place ink. Since the bristles are sparsely distributed this mode is not used for painting in general, but rather only for achieving specialized unrealistic affects.

Lee chose not to address the sparse bristle problem because his main interest lay in applying the substantial body of ink diffusion research to his improved brush model. Building upon the fiber mesh ink diffusion model proposed in 1991 by Guo and Kunii, Lee developed a complete and realistic ink diffusion simulator that allows for the flexible specification of different paper types and ink properties. The idea behind the fiber mesh model is to divide the paper into paper units called papels. A random mesh of fibers is composited on top of the papel units to define the underlying fiber structure of the paper. The fibers, represented as splines, provide pathways for ink particles to diffuse from one papel to its neighboring papels. The computation can be viewed as evolving a 2D cellular automaton over time – the diffusion dynamics provide the update rules and the state of each cell is a tuple defining its current ink properties and it connectivity via fiber pathways to its neighboring cells. In order to incorporate this diffusion model into the painting system, Lee considered that diffusion occurs outward from the boundary edges of a stroke.

Using this diffusion model, Lee was able to produce highly realistic images of ink spots. When combined with his brush model, however, strokes look overly regular and lack many of of the stylistic effects seen in brush painting, such as white paper exposed in the body of a stroke. The resultant painting system produced acceptable results for

simulating *sumi-e* painting, but was not versatile enough for simulating the more diverse strokes found in Chinese brush painting. A typical Chinese brush painting contains hundreds or thousands of strokes in a variety of styles which simply cannot be captured by Lee's system. The failure of Lee's highly advanced diffusion model combined with a slightly more advanced brush model to capture the nuances of Chinese brush painting pushed further research towards developing more realistic brush models.

## 2.3 The Virtual Chinese Brush (VCB) [CT02]

Lee's research lead brush painting into the realm of 3D modeling and demonstrated that a high degree of physical detail needs to be encoded into the brush model in order to achieve realistic results. Upon this rationale Nelson Chu and Chiew-Lan Tai of Hong Kong University devised a more physically accurate 3D brush model. Focusing first on modeling the brush and handling user input, while leaving ink diffusion for later, Chu and Tai demonstrated that among the three problems in this field, diffusion is actually the least significant factor in producing realistic results. Diffusion is in fact so insignificant to brush painting simulation that Chu and Tai were able to produce fantastic results in spite of implementing a simple, *ad hoc* diffusion model.

Aside from Chu and Tai's significant finding regarding ink diffusion, they also thrust brush modeling to a new level. Inspired by Bill Baxter's work at UNC on modeling western style brushes using implicit integration of a spring-mass particle system, Chu and Tai set out to adapt and extend this approach to model the Chinese brush [BSLM01]. The key characteristic that Baxter's brushes lacked and that Chu and Tai endeavored to integrate into their model was the ability for the brush to spread and split while painting. Spreading and splitting, they argue, is necessary for producing many of the characteristic effects that appear in Chinese brush painting.

Chu and Tai's model represents the brush as a collection of bristle tufts. The number of tufts is predefined before a stroke is painted and can vary from one to approximately

five depending on available computing power. The anatomy of a tuft is a skeleton defined by a polyline spine with lateral segments emanating from the joints. The lateral segments are held parallel to the paper surface, though each can rotate within its own plane. To generate the brush body, a polygon mesh is tessellated around the skeleton.

The dynamics that control the deformation of the tufts are both the key to the VCB's realistic results and the system's major drawback. As the system runs in real-time, the state of a tuft's skeleton is updated by solving a constrained energy minimization problem using quadratic sequential programming. The constrained energy minimization problem is formulated by assigning a bend and twist energy to each node in the system (nodes on the polylines and lateral segments) as well as specifying that no node may penetrate the paper and that the lateral nodes are to remain parallel to the paper plane. The intuitive idea behind this approach is that at each time step the skeleton will want to achieve its minimum energy state given the satisfaction of its constraints. The dynamics used by their system produced highly realistic brush deformations. The numeric solution to their energy minimization problem, however, is very costly. The result of this high cost is a low limit on the number of tufts that their system can model while still operating in real time. In practice they found on commodity hardware they could use up to five tufts at a time. This inherently limits the resolution of detail that their brush model can achieve. In order to provide a partial solution to this problem they apply an alpha map to each tuft that attempts to make its tip look more like a collection of hairs. The alpha map, in theory could dynamically evolve over time, however in their system it is static. The system produces a rendered stroke by clipping all of the constituent tuft's polygon meshes against the paper plane and rendering the result taking into account the alpha map. The alpha composition produces the illusion of limited diffusion without explicitly developing a diffusion model.

Another major contribution of Chu and Tai's system is the creation of a novel input device. They fitted a real brush handle with a gyroscope and an ultrasound buzzer. By

using three ultrasound receivers placed on a computer monitor they can calculate the 3D position of the brush, while the gyroscope provides tilt information. This system allows a painter to use his or her tool to more naturally move the virtual brush. They also enabled their system to use a regular mouse or a graphics tablet that reports tilt and pressure information to the system.

Within a narrow range of painting styles Chu and Tai's system produces excellent results. The narrow range of painting styles is dictated by the fact that the brush can only break up into small number of tufts and that number has to be chosen before a stroke is painted. Most of the strokes generated by Chu and Tai's system look like wet brush strokes of varying opacity. Highly texture strokes achieved by dry brushes that fragment into hundreds of bristle bundles cannot be achieved. The novel input device, while a great advancement has the problem of depriving the real brush of its characteristic light and spontaneous feel. The gyroscope is heavy and its need to be physically wired to the computer is cumbersome.

## 2.4 A Solid Modeling Approach [XTLP, XLTP03]

Chu and Tai's virtual Chinese brush was a monumental improvement over Lee's brush model, however its stylistic limits left a clear path for improvement. In order to make a brush model that is expressive enough to capture a wide range of painting styles Lau, et al of The University of Hong Kong devised a new brush model based on solid modeling principles and an intuitive geometric deformation system rather than a computationally expensive physics based approach.

The motivating idea behind switching from a more accurate physical simulation to an approximate geometric simulation lies in the conjecture that shifting computational resources from modeling deformation to modeling the brush in greater detail is ultimately a beneficial trade-off. Lau's first approach, published in mid 2002, created an expansive framework for brush painting including a sophisticated brush model, an ink diffusion

model, and an automated machine learning system for tuning brush model configuration parameters to better mimic sample strokes. Although performance issues are not mentioned in Lau's 2002 paper, a followup publication in late 2003 presents a modified brush model that addresses performance problems.

Lau's original brush is modeled as a solid NURBS object generated by using a circle at the base of the brush, an ellipse in the middle of the brush, and a line at the end of the brush as profiles for the general solid modeling sweeping operation. These geometric primitives are connected along a spline called the control axis. In response to pressure and internal stress a single NURBS solid, referred to as a writing primitive in Lau's nomenclature, can subdivide into two NURBS objects. The new writing primitives share the same control circle (fixed at the base of the brush handle), but have their own control ellipses and control lines and can thus move independently. The writing primitives deform based on rules that adjust the position and shape of the control axis, control ellipse, and control line (the control circle is always fixed to the brush handle). By using a computationally inexpensive dynamics system the brush should ideally be able to split into hundreds of writing primitives.

Similar to the virtual Chinese brush, the total brush foot print in Lau's model is the sum of all of the writing primitives clipped against the paper plane. Interestingly, in Lau's paper there is no mention of a strict constraint for keeping the writing primitives from penetrating below the paper plane. After clipping, ink is distributed into each of the writing primitives' foot print using either a radial or vector gradient based on ink values specified on the control points of the writing primitives control axis.

Lau's 2003 paper presents a modified model in which a collection of approximately ten archetypal writing primitives are used to model the brush in a hierarchical fashion. The brush starts as a whole and splits in response to pressure, just like in the original model. However, instead of forming independent writing primitives the brush divides into classes of writing primitives, each defined by one of the archetypes. Deformation

effects are calculated explicitly for each of the archetypal writing primitives in use and then this deformation, possibly altered, is applied as an affine transformation to the entire class of writing primitives. This hierarchical approach allows the brush to split into thousands of writing primitives before the system can no longer perform in real time. Another performance alteration put forth in this paper is the use of a precomputed database of deformation hints. This allows the on-line deformation calculations to be relatively simple while relying on the offline information to correct oversimplifications and deformation errors.

For input Lau designed the system to work with wheel mice and graphics tablets. Using a wheel mouse the movement controls positioning on the paper plane while the wheel controls vertical height. A combination of key bindings are used to control the other degrees of freedom. With a graphics tablet input is more natural because tilt and height can be control by moving the digital pen rather than scrolling a wheel and pressing keys.

Lau's systems achieves amazingly realistic results. He juxtaposes examples of real strokes next to simulated imitations – the differences can only be seen with careful examination. The system, however, is not a perfect solution to problem of simulating Chinese brush painting. One limitation of the system is the inability for writing primitives to merge back together once they split, instead the brush must be completely reset with a dip action. The system also requires careful tuning to achieve its highly realistic results. This tuning is aided with a machine learning system that automates much of the process, but it nevertheless inserts another step before painting can begin (unless the defaults are acceptable).

# 3 The Parametric Hairy Brush Design

The goal of the Parametric Hairy Brush (PHB) design is to continue the development of brush painting systems in order to address existing limitations, produce realistic results, and do so in a manner that mimics the non-digital painting process. Due to the implications of Chu and Tai's research, PHB research is focused on the brush model, rather than ink diffusion. This section provides a detailed specification for the PHB model. Its design is influenced by Lau's work but uses a novel parameterized geometric approach to efficiently represent the smallest units of the brush. The result is much less complex geometrically and is also computationally more efficient.

## 3.1 Introduction to the PHB Model

The PHB is built in a two level hierarchy composed of a control volume and its children, the bristle bundles. A bristle bundle is the smallest unit of representation in the model. Depending on the resolution of the rendering, a bristle bundle can be thought of as an individual bristle or a closely packed neighborhood of bristles. For instance, when rendering on a 512 pixel square paper a brush with 350 bristle bundles is used.

Internally a bristle bundle is a Bézier curve with control points that are parameterized on the control volume. The control volume provides a high level, approximate description of how the bristle bundles move collectively. During simulation the control volume is deformed, thus moving the control points of each bristle bundle. This provides a "best guess" of how the individual bristles will move. At the second level of the hierarchy, deformation is accomplished by adjusting the bristle bundle's control point parameters according to a set of rules. These adjustments model the small scale changes that take place between bristles inside of the brush, such as bristles clumping or spreading out evenly.

## 3.2 The Control Volume

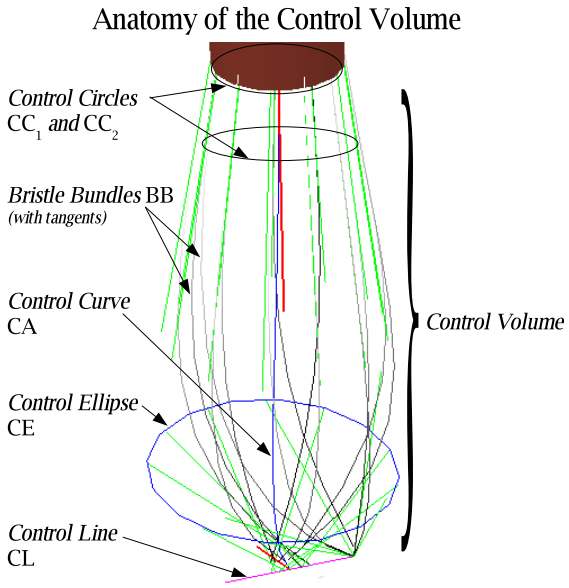


Figure 1: PHB rendered with sparse bristles to reveal its anatomy

The control volume is heavily influenced by Lau's solid model. It is composed of a control curve CA defining the central axis of the volume, two control circles $CC_1$ and $CC_2$, a control ellipse CE, and a control line CL. The control circles are used for defining the first two control points of each bristle bundle and the control ellipse and control line are used for defining the latter two control points of each bristle bundle (Figure 1).

The control curve (Figure 2) is a Bézier curve, with control points $CA_1$, $CA_2$, $CA_3$ and $CA_4$, originating from the center of the bottom of the brush handle. The first two control points are fixed relative to each other and lie on an imaginary line protruding from the bottom of the brush handle. The other two control points are initially very close together and lie on a line parallel to the paper plane. During deformation the magnitude of the vector $\overrightarrow{CA_4CA_3}$ between the second pair of control points is expanded or contracted as necessary in order to maintain a control curve with an approximately constant arc length.

$CC_1$ is fixed at the bottom of the brush handle and never moves from this position.

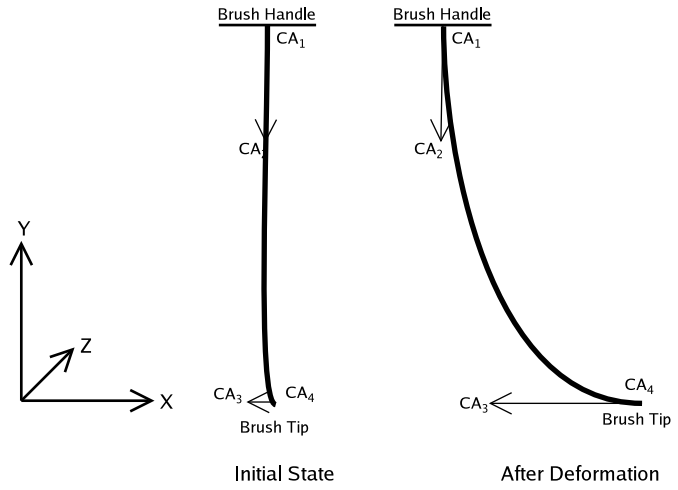


Figure 2: The control curve CA with control points and tangent vectors

$\mathrm{CC}_2$ is centered on (and perpendicular to) the control curve at a fixed location $\frac{1}{8}$ of the arc length along the control curve. A control circle is mathematically represented as a series of matrix operations (in homogeneous coordinates) on a parameterized circle:

$$M(x_0, y_0, z_0, r, \theta, \phi, \mathrm{CC}_p, \mathrm{CC}_r) = T(x_0, y_0, z_0) R_y(\theta) R_x(\phi) \mathrm{Circle}(r, \mathrm{CC}_p, \mathrm{CC}_r)$$

$$= \begin{bmatrix} 1 & 0 & 0 & x_0 \\ 0 & 1 & 0 & y_0 \\ 0 & 0 & 1 & z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) & 0 \\ 0 & \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathrm{CC}_r r \cos(\mathrm{CC}_p) \\ 0 \\ \mathrm{CC}_r r \sin(\mathrm{CC}_p) \\ 1 \end{bmatrix}$$

Performing the multiplications above yields the following parameterization (in 3D coordinates) of the interior and boundary points of a control circle for a fixed $r$, $x_0$, $y_0$, $z_0$, $\phi$, and $\theta$:

$$\mathrm{CC}(\mathrm{CC}_r, \mathrm{CC}_p) = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x_0 + \mathrm{CC}_r r(\cos(\theta)\cos(\mathrm{CC}_p) + \cos(\phi)\sin(\theta)\sin(\mathrm{CC}_p)) \\ y_0 + \mathrm{CC}_r(-r\sin(\phi)\sin(\mathrm{CC}_p)) \\ z_0 + \mathrm{CC}_r r(\sin(\theta)\cos(\mathrm{CC}_p) + \cos(\phi)\cos(\theta)\sin(\mathrm{CC}_p)) \end{bmatrix}$$

$\mathrm{CC}_p$ is in the interval $[0, 2\pi)$ and parameterizes the perimeter of the circle, $\mathrm{CC}_r$ is in the interval $[0, 1]$ and parameterizes the circle's radius, and $r$ is the radius of the control circle. The angle $\phi$ is rotation around the $X$-axis and $\theta$ is rotation around the $Y$-axis. In this coordinate system the paper is on the $XZ$-plane and up vector points along the $Y$-axis. For $\mathrm{CC}_1$, $\theta$ and $\phi$ are adjusted to match the tilt of the brush and for $\mathrm{CC}_2$ $\theta$ and $\phi$ are chosen to make $\mathrm{CC}_2$ perpendicular to CA. All points inside of a control circle correspond to a unique pair ($\mathrm{CC}_p$, $\mathrm{CC}_r$).

The control ellipse CE is centered on the control curve $\frac{3}{4}$ of the arc length along the control curve. Similar to the control circles, the interior points of the control ellipse are parameterized in two variables $\mathrm{CE}_p$ and $\mathrm{CE}_r$ using the following equation (derived in the same way as above):

$$M(x_0, y_0, z_0, a, b, \theta, \phi, \mathrm{CC}_p, \mathrm{CC}_r) = T(x_0, y_0, z_0) R_y(\theta) R_x(\phi) \mathrm{Ellipse}(a, b, \mathrm{CC}_p, \mathrm{CC}_r)$$

$$\mathrm{CE}(\mathrm{CE}_r, \mathrm{CE}_p) = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x_0 + \mathrm{CE}_r(a\cos(\theta)\cos(\mathrm{CE}_p) + b\cos(\phi)\sin(\theta)\sin(\mathrm{CE}_p)) \\ y_0 + \mathrm{CE}_r(-b\sin(\phi)\sin(\mathrm{CE}_p)) \\ z_0 + \mathrm{CE}_r(-a\sin(\theta)\cos(\mathrm{CE}_p) + b\cos(\phi)\cos(\theta)\sin(\mathrm{CE}_p)) \end{bmatrix}$$

$\mathrm{CE}_p$ is in the interval $[0, 2\pi)$ and parameterizes the perimeter of the ellipse, $\mathrm{CE}_r$ is in the interval [0, 1] and parameterizes the proportional distance to the center of the ellipse, $a$ is the length of the major axis, $b$ is the length of the minor axis, $\theta$ is rotation about the $Y$-axis, and $\phi$ is rotation about CE's major-axis. Values for $\theta$ and $\phi$ are chosen to match the orientation vector of the control line (described below) and make CE's major-axis perpendicular to CA. All points inside of the control ellipse correspond to a unique pair ($\mathrm{CE}_p$, $\mathrm{CE}_r$).

The control line CL is centered at the end point of the control curve and is always parallel to the paper plane. It is parameterized in one variable $\mathrm{CL}_l$, which interpolates

between two points $P_0$ and $P_1$, using the following equation:

$$\mathrm{CL}(\mathrm{CL}_l) = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} (1-\mathrm{CL}_l)P_{0x} + \mathrm{CL}_l P_{1x} \\ (1-\mathrm{CL}_l)P_{0y} + \mathrm{CL}_l P_{1y} \\ (1-\mathrm{CL}_l)P_{0z} + \mathrm{CL}_l P_{1z} \end{bmatrix}$$

where

$$P_0 = \begin{bmatrix} \mathrm{CA}_{4x} - V_x \frac{L}{2} \\ \mathrm{CA}_{4y} - V_y \frac{L}{2} \\ \mathrm{CA}_{4z} - V_z \frac{L}{2} \end{bmatrix} \quad \text{and} \quad P_1 = \begin{bmatrix} \mathrm{CA}_{4x} + V_x \frac{L}{2} \\ \mathrm{CA}_{4y} + V_y \frac{L}{2} \\ \mathrm{CA}_{4z} + V_z \frac{L}{2} \end{bmatrix}$$

Or in vector notation:

$$\mathrm{CL}(\mathrm{CL}_l) = (1-\mathrm{CL}_l)(\mathrm{CA}_4 - \vec{V}\frac{L}{2}) + \mathrm{CL}_l(\mathrm{CA}_4 + \vec{V}\frac{L}{2})$$

$\mathrm{CL}_l$ is in the interval $[0, 1]$ and parameterizes the length of the line, $\mathrm{P}_0$ and $\mathrm{P}_0$ are the end points of the control line, $\vec{V}$ is the vector pointing in the direction of the line, and $L$ is the length of the line. All points on the control line correspond to a unique parameter $\mathrm{CL}_l$.

The control volume can be thought of intuitively as the hull defined by sweeping along the control curve through the sweeping profiles defined by the previous control geometries.

## 3.3 Bristle Bundles

A bristle bundle is a Bézier curve with control points $\mathrm{BB}_1$, $\mathrm{BB}_2$, $\mathrm{BB}_3$, and $\mathrm{BB}_4$. For a given bristle bundle, $\mathrm{BB}_1$ and $\mathrm{BB}_2$ are each assigned a pair of parameters specifying their location in the interior of $\mathrm{CC}_1$ and $\mathrm{CC}_2$. These parameters are constant and can thought of as defining the fixed root of a brush hair. $\mathrm{BB}_3$ is assigned a pair of parameters specifying an interior point of the control ellipse and, similarly, $\mathrm{BB}_4$ is assigned a parameter

specifying a point on the control line. A bristle bundle is defined as:

$$\mathrm{BB} = (\mathrm{BB}_1, \mathrm{BB}_2, \mathrm{BB}_3, \mathrm{BB}_4)$$
$$= (\mathrm{CC}_1(\mathrm{BB}_1{}_{\mathrm{CC}_p}, \mathrm{BB}_1{}_{\mathrm{CC}_r}), \mathrm{CC}_2(\mathrm{BB}_2{}_{\mathrm{CC}_p}, \mathrm{BB}_2{}_{\mathrm{CC}_r}), \mathrm{CE}(\mathrm{BB}_3{}_{\mathrm{CE}_p}, \mathrm{BB}_3{}_{\mathrm{CE}_r}), \mathrm{CL}(\mathrm{CL}_l{}_{\mathrm{BB}_4}))$$

Geometrically, this defines a curve starting at the base of the brush handle, passing through a point in the control ellipse, and ending on the control line. As the control volume changes $\mathrm{BB}_1$, $\mathrm{BB}_2$, $\mathrm{BB}_3$, and $\mathrm{BB}_4$ are updated by evaluating their corresponding parameters in the appropriate equation.

## 3.4 The Simulation Loop

The simulation loop executes as follows: (1) Sample the input device (A mouse, tablet, etc.), (2) Calculate the new virtual brush position and velocity, (3) Apply control volume deformations, (4) Adjust bristle bundle parameters, (5) Render and (6) Repeat.

Input device sampling requires acquiring a new set of coordinates from the input device. Depending on available information, the new coordinates may express three to five degrees of freedoms (ignoring the $6^{\mathrm{th}}$ DOF: twists around the $Y$-axis). The minimum required is a new $X$, $Y$, $Z$ coordinate set specifying the center of $\mathrm{CC}_1$. After acquiring the new input coordinates, they are feed into an inertial predictor. The inertial predictor is implemented as described in Lau's 2002 paper [XTLP]. It calculates a new position for the virtual brush based on a weighted average of the brush's previous four positions and the brush's current velocity. After the brush's new position and velocity are calculated, these values are used to deform the control volume and the bristle bundle parameters as described in the next section. The scene is rendered and then the simulation repeats. In order to provide interactive real time painting, the simulation loop must repeat at a regular interval that is frequent enough to smoothly respond to user input.

## 3.5 Deforming the Control Volume

In order to minimize the computational cost of deforming the control volume, a simple geometric approximation is used instead of a costly physical simulator. The cost of this deformation must be minimal in order to save computational resources for other aspects of the simulation such as bristle bundle deformation and ink diffusion.

The geometric approximation is based on the intuition that when a brush is pressed against a piece of paper and then displaced by a vector $D$ the section of the brush attached to the handle will move the full distance, but the tip will move only a fraction of that distance (Figure 3). The fraction of the distance is proportional to the pressure exerted on the paper by the brush. Pressure is approximated by assuming that it is proportional to the product of the brush's velocity, elasticity, curvature, and the paper's roughness. Using this simple principle it is possible to model the dynamics of the brush with surprising grace.

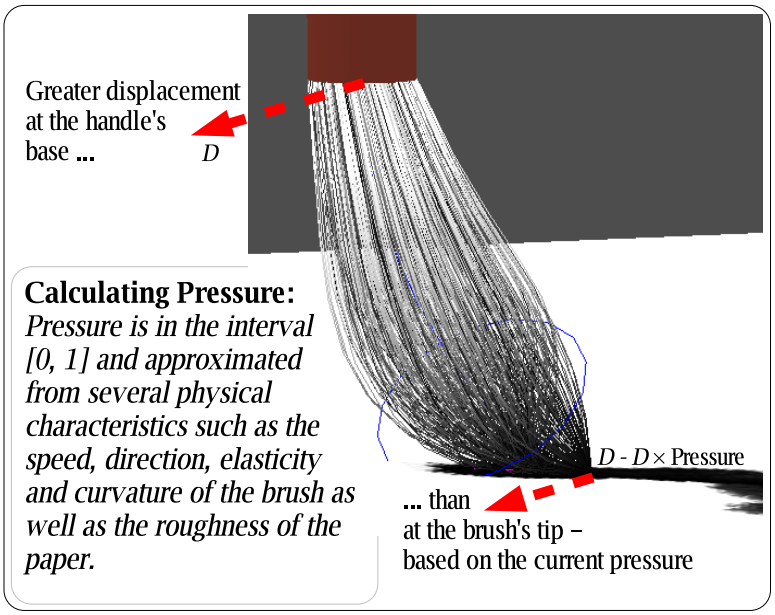


Figure 3: The base and tip may move at different rates

Deformation of the control volume begins by displacing CA's control points. The control points representing where the brush is fixed to the handle, $CA_1$ and $CA_2$, are

translated by the full displacement vector $D$. Before the other two control points, representing the brush tip, are translated several physical quantities at the current time step $t$ are approximated: brush curvature, speed, and direction – and using these: pressure. Pressure is measured as a relative quantity in the interval $[0, 1]$. In order to achieve this, all physical approximations are normalized in some fashion to the interval $[0, 1]$.

The curvature ($\kappa_t$) is calculated as the difference between the CA's arc length ($\alpha_t$) and the length of the chord drawn between $\mathrm{CA}_1$ and $\mathrm{CA}_4$ normalized with respect to the arc length:

$$\kappa_t = \frac{\alpha_t - |\mathrm{CA}_4 - \mathrm{CA}_1|}{\alpha_t}$$

Curvature has the desired properties of being minimal, $\kappa_t = 0$, when the chord length is maximal: $|\mathrm{CA}_4 - \mathrm{CA}_1| = \alpha_t$. When the chord length is minimal, $|\mathrm{CA}_4 - \mathrm{CA}_1| = 0$, then the curvature is maximal, $\kappa_t = 1$. It is important to note that $\kappa_t$ is never allowed to reach zero because it would "crowd out" the other physical parameters and freeze the brush in its current configuration. This is enforced by keeping the length of the vector from $\mathrm{CA}_4$ to $\mathrm{CA}_3$ non-zero and keeping it parallel to the paper plane.

Instead of using the absolute speed of the current displacement vector $D_t$, speed ($\sigma_t$) is calculated relative to the speed at the previous time step:

$$\sigma_t = \frac{1}{1 + \frac{1}{||D_t| - |D_{t-1}||}}$$

Relative Speed, $\sigma_t$, has the property of being sensitive to small changes, but plateauing towards 1 when speed changes are great. This is a highly desirably property because $\sigma_t$ is responsive to subtle changes in brush, but provides numeric stability during sudden, extreme changes.

The direction of motion ($\delta_t$) is calculated based on the observation that pushing a brush towards its tip should increase pressure more than dragging the brush away from

its tip. This is realized with the formula:

$$\delta_t = \frac{\text{angle}(u, v)}{\pi} = \frac{\arccos(u \cdot v)}{\pi}$$

Where $u$ is a the unit vector from $CA_4$ to $CA_3$ and $v$ is the unit vector in the direction of $D_t$. When $u$ and $v$ are aligned (the brush is moving away from the tip) $\delta_t$ is 0. As the angle between $u$ and $v$ increased $\delta_t$ grows toward 1.

The physical quantities $\kappa_t$, $\sigma_t$, and $\delta_t$ are combined with paper roughness $\rho$ and brush elasticity $\epsilon$ to calculate the pressure $\psi_t$. The square root is taken as a correction factor based on empirical experimentation with the model.

$$\psi_t = \sqrt{\kappa_t \times \sigma_t \times \delta_t \times \epsilon \times \rho}$$

Using $\psi_t$ the control point $CA_4$ is translated by $D_t - D_t \times \psi_t$ and $CA_3$ is placed on a vector starting at $CA_4$ and pointing in the direction of motion averaged over the last 4 time steps. The exact placement of $CA_3$ along this vector is not very important since it will be changed when the arc length of CA is constrained.

We can verify that this matches our intuition of how the brush should move by examining how $CA_4$ is moved for various values of $\kappa_t$, $\sigma_t$, and $\delta_t$. For instance, if the brush is fully extended, i.e. it has a low curvature, then $\kappa_t$ approaches 0 and the tip of the brush moves by nearly the full displacement $D_t$. In this situation, if $CA_4$ does not move by nearly the full displacement, the result will be an unrealistic stretching of the brush tip. In another situation (Figure 4) imagine that the is brush has very low curvature and is pushed in the direction of its tip. This movement will maximize $\delta_t$ and cause $CA_4$ to move less than the full distance $D_t$. Over a few time steps the curvature will build causing $\psi_t$ to increase until $CA_4$ does not move at all. When $CA_4$ is momentarily stuck in place the brush "folds" over itself and the vector $\overrightarrow{CA_4CA_3}$ flips direction.

After deforming CA, the direction of motion and pressure are used to deform CE and

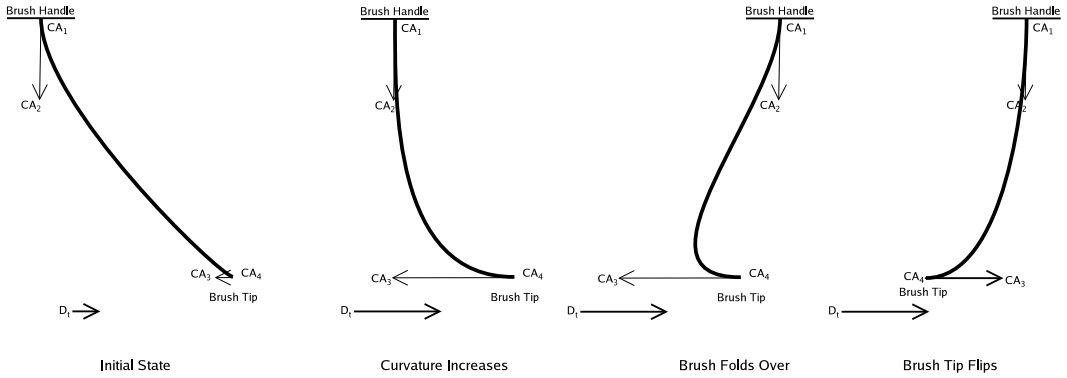


Figure 4: Complex brush dynamics – the tip "folds over" itself.

CL. At each time step CL's orientation and length are adjusted. Orientation is adjusted so that as the brush twists CL is roughly perpendicular to the direction of motion, but then as motion in that direction continues CL aligns itself with that direction. The effect of this strategy – when holding all other pressure parameters constant – is a brush foot print that initially thickens and then tapers off as the brush continues to move in the same direction. The update rule for $V$, CL's orientation vector, at time $t$ is:

$$\text{Let} \quad dv = D_t - D_t \times \psi_t \quad \text{and} \quad V_t^{'} = \left( P_1 + dv \times \left( 1 - \left| V_{t-1} \cdot \frac{dv}{|dv|} \right| \right) \right) - (P_0 + dv)$$

$$\text{Then set} \quad V_t = \begin{bmatrix} \frac{V_x{}'}{|V_t'|} \\ \frac{V_y{}'}{|V_t'|} \\ \frac{V_z{}'}{|V_t'|} \end{bmatrix}$$

$P_0$ and $P_1$ are the end points of CL from the previous time step. The length of CL is chosen between the predefined minimum and maximum lengths $L_{max}$ and $L_{min}$ as $L_t = L_{min} + (L_{max} - L_{min}) \times \psi_t$. CE's orientation $\theta$ is then chosen to match CL's orientation. CE's tilt $\phi$ is chosen as the slope at CE's point of attachment to CA.

In brief, the algorithm for updating the control volume is:

1. **Use the inertial predictor to determine a displacement vector $D_t$**
2. **Translate $CA_1$ and $CA_2$ by $D_t$ to the new position of the brush**
3. **If $CA_4$ does not penetrate the paper:**

**4.** **Translate $CA_3$ and $CA_4$ by $D_t$**

**5.** **Else:**

**6.** **Calculate the current pressure $\psi_t$**

**7.** **Translate $CA_4$ by $D_t - D_t \times \psi_t$**

**8.** **Place $CA_3$ the average velocity vector starting at $CA_4$**

**9.** **If $CA_3$ or $CA_4$ are still below the paper:**

**10.** **Constrain them to be on the paper plane**

**11.** **End If**

**12.** **Adjust the magnitude of the vector between $CA_4$ and $CA_3$ in order maintain CA's arc length**

**13.** **Rotate CL**

**14.** **Extend or contract CL**

**15.** **Rotate $\theta$ and $\phi$ of CE**

**16.** **Extend or contract CE's major and minor axes**

**17.** **End If**

**18.** **Using the new control curve position, update $CC_1$, $CC_2$, CE, and CL**

## 3.6 Adjusting Bristle Bundle Parameter

The second level of the PHB hierarchy allows for deformation of individual bristle bundles. A bristle bundle is deformed by changing any of the parameters that define how it relates to the control volume. Since the bristles are glued into the brush handle the parameters that define the first two control points never change. The parameters that define how a bristle passes through the control ellipse can be altered, but this is not needed in order to produce a nice tip splitting effect. The only parameter left to adjust is the control line parameter. Figure 5 illustrates how adjusting only the bristle bundle parameters along CL can completely split the brush tip.

The parameters along the control line should change in one of two orthogonal ways. When the brush is moving in the direction of the CL's orientation vector then the parameters should cluster toward the end of CL opposite to the direction of motion, but

when the brush is moving perpendicular to CL's orientation vector then the parameters should spread back out throughout the brush.

The ability to control the relative position of each bristle bundle by manipulating only one number is one of the PHB's great strengths. This flexible and simple framework for deforming bristle bundles provides a great workbench for experimentation. Future research should focus on developing more realistic strategies for manipulating the bristle bundles.

Figure 5: $CL_l$ is either 0 or 1 for all bristle bundles creating a split tip.

# 4 Significant PHB Design Decisions

The goal motivating the PHB design is the simulation of individual hairs in real time. In order to accomplish this, computational resources have to be carefully divided amongst the functions that comprise the simulation. Before Lau's work brush models generally used expensive numerical approximations of physical phenomenon in order to simulate the overall deformation of the brush body – such as numerical integration of differential equations or constrained energy minimization. These technique produce accurate soft body deformation dynamics, but at a computational cost that prohibits the detailed simulation of other aspects of the model. The PHB, like Lau's model, uses an intuitive

geometric approximation for the deformation of the brush in order to save computational resources for simulating other aspects of the brush, such as individual bristle movements and ink diffusion.

## 4.1 Hierarchical Deformation

The hierarchical construction of PHB is based on the observation that the individual movement of bristles within the brush is closely correlated to the overall movement of the brush. In particular, the correlation is strongest where the bristles are fixed at the base of the brush and weakens monotonically as you move toward the brush's tip. This correlation implies that the deformation of the control volume will model the exact behavior of the bristle bundles at the base of the handle, but only approximate the behavior of the bristle bundles as you move away from the handle. Additionally, the error in the approximation grows monotonically as you approach the tip.

The deformation of the control volume acts as a mapping from the previous set of bristle positions to a good estimate of the set of bristle positions at the next time step. After applying this mapping, the bristle positions can be tweaked by making small changes to the bristle parameters. The parameterization of the bristles is invariant under transformations to the control volume. Neighboring bristles are guaranteed to still be neighbors after a transformation. Thus it is possible to intelligently adjust these parameters without evaluating and analyzing the Bézier curves of each bristle after deforming the control volume. Although the model allows for adjusting the parameters at any of the control geometries, the parameters only need to be tweaked at the tip of the brush and possibly at the control ellipse. Since the tip of the brush is only parameterized by one variable it is computationally inexpensive to make intelligent changes to the parameters. The computational complexity of adjusting the parameters for the ellipse is higher, due to an additional parameterization dimension. Empirical results indicate that making adjustments at the control ellipse is unnecessary (as seen in Figure 5).

## 4.2 Strict vs. Loose Constraints

It is tempting to apply strict, physically accurate constraints to the brush model. Such as absolutely preventing the brush from penetrating the paper, guaranteeing that bristles do not intersect, and maintaining constant bristle lengths. However, the computational cost of strictly enforcing these constraints is not matched by a great gain in realism (or even a small gain). The approach taken here is to enforce constraints as much as needed, but no more. The constraints implemented in PHB are: the control curve may not penetrate the paper and each bristle's arc length is maintained within 95% to 105% of its original length.

The first constraint is easily enforced due to the choice of using Bézier curves. Since a Bézier curve is bounded by the convex hull defined by its control points, keeping these control points on or above the paper satisfies this constraint. This, however, does not imply that each bristle will be entirely above the paper. In the case that the control ellipse intersects the paper it is possible for a bristle to partially intersect the paper. However, due to the placement of the ellipse on the control curve, it tends to stay mostly above the paper. The small amount of bristle-paper intersection was empirically determined to be acceptable.

Maintaining each bristle's arc length is crucial to the realism of the brush model. A high degree of accuracy, however, is not critical. Empirical tests show that as long as the each bristle is forced to within $\pm 5\%$ of its original length the model produces essentially the same results. This is an important finding because it is relatively easy to enforce this loose constraint, but very difficult to enforce a strictly constant arc length.

## 4.3 Bézier Curves

While the convex hull property of Bézier curves was a primary consideration when deciding on a curve type, it was not the deciding factor. Several types of curves exhibit this

property, however, the simplicity of Bézier curves makes the convex hull easy to define and change. In addition to the convex hull property, Bézier curves are attractive because fast and easy to implement algorithms exist for evaluating them. This is critical due to the vast number of Bézier curves used in the PHB model.

# 5 Significant PHB Model Implementation Details

## 5.1 Implementation Platform

The PHB was implemented on Linux in C using GNOME, GTK+ and OpenGL to provide user interface and graphics rendering functionality. Extensive use was made of GLib's convenience data structures and callback scheduling functionality. All of these libraries are cross platform compatible allowing PHB to be compiled for other target operating systems such as Windows and OS X.

## 5.2 Geometric Algorithms

GNU gprof was used to collect profiling information about the runtime characteristics of PHB. As expected these data show that the majority of function calls are related to manipulating Bézier curves. Thus an efficient implementation of Bézier curves is crucial to performance. OpenGL provides the ability to create and render Bézier curves natively. Once an OpenGL Bézier curve is created it can only be manipulated with OpenGL's built-in transformation functions. This does not provide enough flexibility for PHB since hundreds of Bézier curves need to be frequently adjusted by defining new control point positions. The overhead and complexity of using OpenGL's model view matrix operations to accomplish these transformations prohibited this approach.

To meet the needs of PHB a custom Bézier curve library was implemented. Each curve has an internally stored algebraic definition as well as a polyline geometric ap-

proximation. After the algebraic form is defined the Bézier curve object waits in the *unprepared* state until an evaluation is requested. The curve is evaluated using forward differencing over a globally defined number (8) of equally spaced points on the interval $[0, 1]$ [Hof95]. The results of an evaluation are cached in the object and its state is set to prepared. Caching is very important since the curve will need to be evaluated again without being changed in order to render it as a polyline. Additionally, at the next time step the Bézier curve might not be altered, allowing the cache to be used once more. If the Bézier curve is transformed, then its state needs to be set to *unprepared* to force cache regeneration.

Efficient evaluation of Bézier curves is also crucial due to the loose constraint of keeping each curve's arc length within $\pm 5\%$ of its original length. This is accomplished by approximating arc length as the cumulative length of the component segments in the curve's polyline approximation. If the curve is too long, then the vector from $BB_4$ to $BB_3$, in the case of a bristle bundle, is iteratively contracted until a maximum iteration count (40) is reached or the desired arc length is achieved. The case when a curve is too short is handled with a similar extension of the vector. Empirical results show that under most conditions this approach terminates after fewer than 10 iterations. In some cases it reaches the max iteration count on the first attempt to resize, but is then restored to an appropriate length during the next time step.

A sample of some profiling data used in tuning these functions:

```
  %   cumulative   self              self     total
 time   seconds   seconds    calls  ms/call  ms/call  name
 48.03     14.14    14.14 20915424     0.00     0.00  bezier_evaluate
 29.11     22.71     8.57  1822720     0.00     0.01  bezier_arc_length
  9.61     25.54     2.83 16404983     0.00     0.00  distance_squared
  5.77     27.24     1.70  2323936     0.00     0.00  bezier_prepare_evaluation
  2.24     27.90     0.66   500500     0.00     0.01  bristle_render
  1.26     28.27     0.37   250403     0.00     0.10  bezier_match_length
  1.22     28.63     0.36   750904     0.00     0.00  bezier_reset_evaluation
  ....
```

```
0.00     29.44     0.00      715     0.00    35.30  brush_update_current_position
....
```

The function brush_update_current_position executes on average at 35.30 ms/call. This yields a frame rate of nearly 30 frames per second.

## 5.3 Rendering Algorithms

Most of the rendering is accomplished in straightforward OpenGL. The one exception is the generation of the painting itself. The painting is modeled as a dynamically changing texture mapped to the paper object. The texture is generated using the following algorithm:

**1. Initialize a white filled texture and map it to the paper object**

**Loop:**

**2. Update the brush position and simulate deformations**

**3. Switch to an orthographic projection looking down on the paper object**

**4. Set the near and far clipping planes to just above and significantly below the paper and turn off depth testing**

**5. Render the brush first followed by the paper**

**6. Copy a small region around the brush from the frame-buffer to the corresponding region of the paper texture used at the last time step**

**7. Reset to the previous perspective view, enable depth testing, and repeat the loop**

The output of this algorithm is the cumulative sequential rendering of the brush's footprint over time. The clipping plane is extended below the paper so that bristles penetrating the paper surface can still contribute to the brush's footprint. Turning off depth testing and rendering the brush first assures us that the bristle segments below the paper will be visible.

Since this is a pixel precision algorithm, it is possible to use various OpenGL effects when rendering the bristles. This provides a fast and easy method for implementing simple diffusion models. Although the focus of this research is entirely on a brush model, alpha blending is used to provide a simplistic impression of diffusion and ink depletion.

The disadvantage of using a pixel precision algorithm is that the width of each bristle is at least one pixel. This implies that only a limited number of bristles can be added to a brush before the effect of additional bristles is no longer visible. For this reason bristle bundles can be thought of as individual bristles at high resolution or as clusters of bristles at lower resolutions. Empirically tests led to a choice of 350 bristle bundles when the paper is a 512 pixel square texture map. Making the rendering algorithm object precision would eliminate this problem at the cost of extra computation and programming complexity.

## 5.4 Real Time Programming

The entire program executes in a real time loop using a GLib timer callback function. This callback executes as frequently as possible and starts the brush_update_current_position function seen in the profiling data above. The real time loop executes the following steps:

1. **Sample the input device for new brush coordinates**
2. **Use the inertial predictor to generate a displacement vector**
3. **Move the brush and apply deformations as needed**
4. **Update the paper texture**
5. **Update the perspective view of the scene**
6. **Repeat**

## 5.5 Input Devices

The current system can be used with either a mouse or a graphics tablet such as the Wacom Intuos 2. When using a mouse the cursor provides the horizontal coordinates and the scroll wheel provides the vertical displacement. If a scroll wheel is not available then the 'j' and 'k' keys move the brush down and up. The graphics tablet provides much more natural support for controlling the brush. A tablet such as the Intuos 2 relays pressure and tilt information. Pressure is mapped to brush height and the tilt data is used to control the 3D orientation of the brush handle.

While nice strokes can still be simulated using only a mouse, if the system is to be used for its true purpose, painting, then a natural brush like input device is crucial. Giovanni Motta [1] proposed using two inexpensive USB cameras to track LEDs embedded in an actual brush. This method could provide the ultimate experience in simulated painting by allowing the digital artist to use the real tool. Other attempts at making natural brush input devices, such as the input device used in VCB, are suboptimal because they obscure the feel of a real brush by attaching wires as well as heavy and bulky objects to the brush.

# 6 Conclusion

A brush model alone cannot mimic real brush painting. In order to develop a system that achieves that level of realism the brush model must be coupled to a realistic ink diffusion and a natural user input system. The metric for judging the success of a new brush model – on its own – is therefore unavoidably complex. The simple comparison of simulated paintings to real paintings cannot be used as the only measure. Instead properties of the brush model, such as performance, complexity, ease of experimentation, and versatility of painting style must be analyzed.

## 6.1 Aesthetic Results

Figure 6 demonstrates the quality of the results produced by the PHB. Simulated strokes are interlaced with photographs of real strokes in order to provide a basis for judging. To remove bias introduced by the background the simulated strokes were composited onto a texture map created by photographing rice paper. The PHB produces stylistically richer results than the VCB. This can be seen in (A) and (G) where the brush exposes the white paper underneath to produce a stroke with varied width and texture. The results

[1] In personal communications.

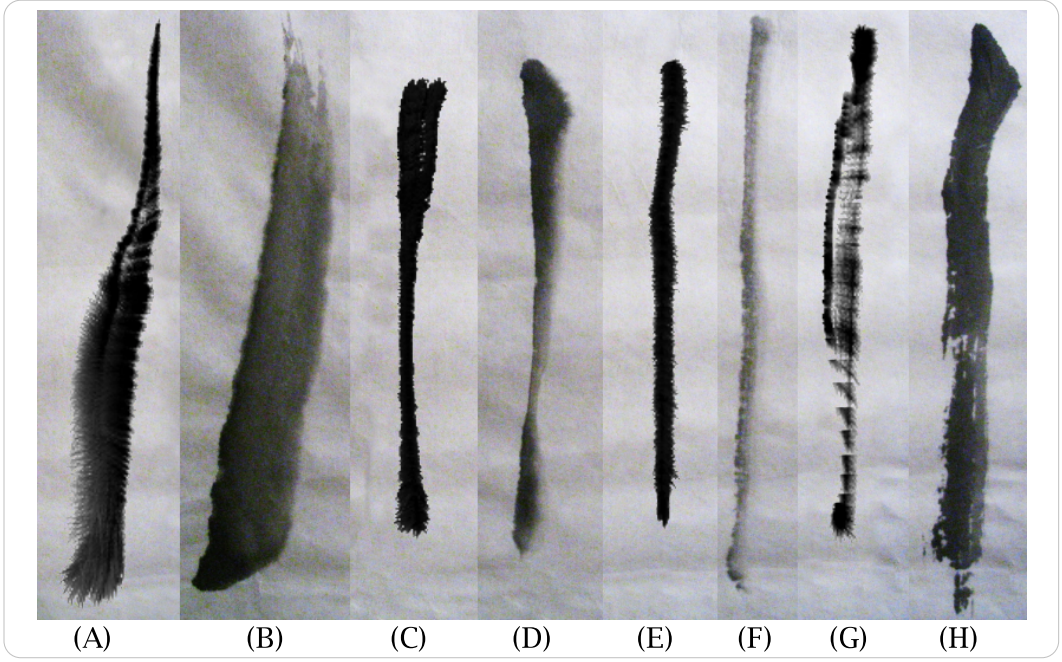


Figure 6: Real vs. simulated strokes: (A), (C), (E) and (G) are simulated. (B), (D), (F) and (H) are real.

published with the VCB do not indicate that it can produce strokes exhibiting these stylistic effects.

In comparison to Lau's painting system, the PHB produces nearly comparable results in terms of stylistic flexibility, but the output does not mimic reality as closely. It is worth noting that the actual rendering of the brush tip (Figure 3) produced by PHB is more realistic than the brush tip rendered by Lau's system, which often looks oddly chaotic. This reflection of reality gives the digital artist a stronger intuition about how the brush will behave since he or she can draw from experience with real brushes. Since Lau's system employs a complex machine learning subsystem for better simulating training samples as well as a complete ink diffusion model it is impossible to know whether the differences are inherent in the brush models or the result of the more complete painting system.

The aesthetic results of the PHB are in league with Lau's painting system and the

VCB. Therefore the brush models employed in these systems provide a good basis for ranking the model properties of the PHB.

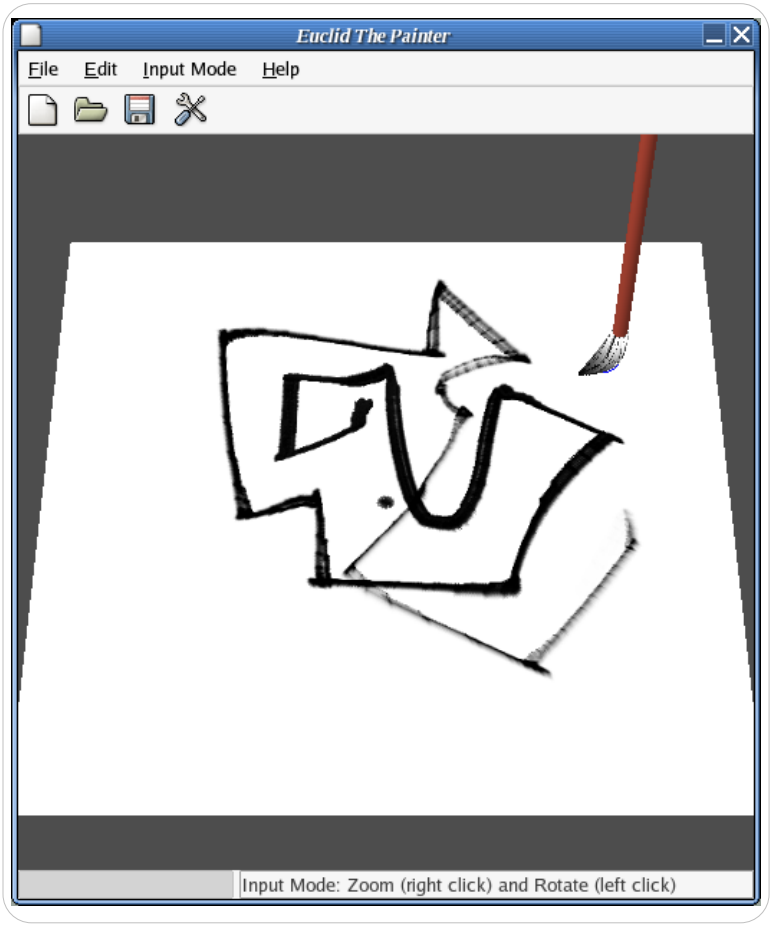


Figure 7: A very long brush stroke exhibiting various stylistic effects.

## 6.2 Complexity

The PHB excels in the simplicity of its design. It is geometrically and algorithmically less complex than Lau's model. The additional geometric complexity of Lau's model is incurred in the use of a hierarchy of NURBS objects based on primitives similar to those used in the PHB's control volume. The PHB uses a collection of Bézier curves which are orders of magnitudes less complex than NURBS solids. The original algorithm for deforming Lau's brush, used in his 2002 paper, and the PHB brush are similar in origin.

They are both based on approximating various physical parameters in order to move the tip of the brush at a different rate than the base of the brush. Lau's 2003 system simplifies the base algorithm further and introduces a database of common errors and their corresponding resolutions in order to supplement the base deformation algorithm. Lau implemented the errors and corrections database to reduce the computational complexity of the deformation algorithm – which is already fairly low. This suggests that the addition of the database fits with the theme of the 2003 paper: optimizing performance at the cost of implementation and design complexity.

## 6.3 Flexibility

Lau's system has a tremendous amount of flexibility in terms of configuration parameters that can be tweaked by hand or by the machine learning subsystem. This form of flexibility, however, is counter productive to mimicking the traditional method of painting. Ideally configuration should be accomplished in entirety when the brush is dipped or reloaded with ink. The PHB does not yet support a sophisticated system for dipping the brush, but when it does it will be in better shape for matching the natural process of painting precisely because the system does not have any configuration parameters that need to be set before painting commences. This is possible because the PHB model encodes the real physical structure of a brush in a very natural and easily manipulated framework making it possible to simulate the detailed movements of bristles within the brush.

# References

[BSLM01] William Baxter, Vincent Scheib, Ming Lin, and Dinesh Manocha. Dab: Interactive haptic painting with 3d virtual brushes. In *Proc. of ACM SIGGRAPH*, pages 461–468, 2001.

[CT02] Nelson S.-H. Chu and C.-L. Tai. An efficient brush model for physically-based 3d painting. In *Proc. of Pacific Graphics*. Beijing, China, IEEE Press., Oct. 2002.

[FvDFH96] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics — Principles and Practice*. The Systems Programming Series. Addison-Wesley, second edition in c edition, 1996.

[Hof95] Kenny Hoff. Derivation of incremental foward-difference algorithm for cubic bezier curves using the taylor series expansion for polynomial approximation. 1995.

[HWS03] Sheng-Wen Huang, Der-Lor Way, and Zen-Chung Shih. Physical-based model of ink diffusion in chinese ink paintings. In V. Skala, editor, *Journal of WSCG*, volume 11, February 2003.

[iAUK92] Ken ichi Anjyo, Yoshiaki Usami, and Tsuneya Kurihara. A simple method for extracting the natural beauty of hair. volume 26, pages 111–120, July 1992.

[Lee99] Jintae Lee. Simulating oriental black-ink painting. *IEEE Computer Graphics and Applications*, 19(3):74–81, May/June 1999.

[Lee01] Jintae Lee. Diffusion rendering of black ink paintings using new paper and ink models. *Computers and Graphics*, 25(2):295–308, April 2001.

[Str86] Steve Strassmann. Hairy brushes. In *SIGGRAPH '86 Proceedings*, pages 225–232, 1986.

[SXSC02] S.L. Su, Y.-Q. Xu, H.-Y. Shum, and F. Chen. Simulating artistic brushstrokes using interval splines. In *CGIM*, August 2002.

[WHCS01] Der-Lor Way, Chih-Wei Hsu, Hsin-Yi Chiu, and Zen-Chung Shih. Computer-generated chinese painting for landscapes and portraits. In V. Skala, editor, *WSCG 2001 Conference Proceedings*, 2001.

[WLS02] Der-Lor Way, Yu-Ru Lin, and Zen-Chung Shih. The synthesis of trees in chinese landscape painting using silhouette and texture strokes. In V. Skala, editor, *Journal of WSCG*, volume 10, 2002.

[WNDS01] Mason Woo, Jackie Neider, Tom Davis, and Dave Shreiner. *OpenGL programming guide*. Addison-Wesley, 2001. Woo.

[XLTP03] Songhau Xu, Francis C.M. Lau, F. Tang, and Yunhe Pan. Advanced design for a realistic virtual brush. volume 22, pages 533–542, September 2003.

[XTLP] Songhua Xu, Min Tang, Francis Lau, and Yunhe Pan. A solid model based virtual hairy brush. pages 299–308.

[YL03] Young Jung Yu and Do Hoon Lee. Interactive rendering technique for realistic oriental. In *Journal of WSCGi, Vol. 11, No. 1*, 2003.

[ZSyT+99] Qing Zhang, Youetsu Sato, Jun ya Takahashi, Kazunobu Muraoka, and Norishige Chiba. Simple cellular automaton-based simulation of ink behaviour and its application to suibokuga-link 3d rendering of tress. In *The Journal of Visualization and Computer Animation 10*, pages 27–37, 1999.