Industrial-Strength Painting with a Virtual Bristle Brush

Stephen DiVerdi[*] Aravind Krishnaswamy[†] Sunil Hadap[‡]
Adobe Systems Incorporated

Abstract

Research in natural media painting has produced impressive images, but those results have not been adopted by commercial applications to date because of the heavy demands of industrial painting workflows. In this paper, we present a new 3D brush model with associated algorithms for stroke generation and bidirectional paint transfer that is suitable for professional use. Our model can reproduce arbitrary brush tip shapes and can be used to generate raster or vector output, none of which was possible in previous simulations. This is achieved by an efficient formulation of bristle behaviors as strand dynamics in a non-inertial reference frame. To demonstrate the robustness and flexibility of our approach, we have integrated our model into major commercial painting and vector editing applications and given it to professional artists to evaluate.

CR Categories: I.3.4 [Computer Graphics]: Graphics Utilities–Paint systems— [I.6.8]: Simulation and Modeling—Types of Simulation–Animation

Keywords: natural media painting, virtual brush, bristle dynamics

1 Introduction

Commercial digital painting has generally been limited to the repeated application of 2D bitmaps along a path (“stamping”), as pioneered by systems such as SuperPaint [Smith 2001]. In general, Gaussian blobs are used, resulting in smooth and relatively textureless compositions, but more interesting bitmaps can be employed to create paint-like strokes. However, more sophisticated stroke generation techniques [Baxter and Govindaraju 2010, Chu and Tai 2004, Laerhoven and Reeth 2007, Xu et al. 2004] have remained squarely in the domain of research systems.

We believe this is because of the hard constraints put on commercial features that research demonstrations do not have to satisfy. Users of industrial-strength painting programs have strict requirements to work at high resolutions (often print-quality of 300dpi or greater) on many different document types (8-, 16-, and 32-bits per channel, RGB, CMYK, and other color formats). Support for these documents must provide interactive performance on a wide range of hardware, including low-end GPUs and CPUs. Stroke quality is of the utmost importance as well – consistently drawing smooth lines with predictable, stable, and robust behavior is absolutely necessary to avoid user frustration. The main advantage of existing commercial digital paint tools is that they satisfy all these requirements with their simplified 2D stamp model. Conversely, research prototypes tend to only work at screen resolution, in a single color space and bit depth, and using specific high-end hardware. They are adequate as proofs of concept, but not as a part of industrial workflows.

In this paper, we present a 3D virtual brush simulation with associated stroke generation and bidirectional paint transfer algorithms that is both high quality and high performance. Our virtual brush model represents individual bristles, and therefore is capable of the full range of configurations of an arbitrary real brush, unlike existing brush simulation research. The focus of this research is in satisfying the requirements of industrial-strength painting workflows, and towards that goal, we have integrated our engine into a major commercial digital painting product, which has since been used by a large number of professional artists to create an impressive array of different natural media painting styles. These artists have played a critical role in our work, directly influencing our design decisions and the final interface, for maximum usability. Another advantage of our brush model is that it enables the output of vector stroke data, and so we have also integrated it into a major commercial vector editing application, with similar success. Professional illustrators have used our vector brushes to create an organic style previously unavailable in vector artwork.

The key contributions of this paper are

- a fast physical formulation of individual brush bristles,
- stroke generation with raster or vector output, and
- artist directed design and results from an industrial setting.

2 Related Work

Most commercial digital painting applications such as Adobe Photoshop [2008] use simple 2D bitmaps stamped along a path to create relatively flat strokes, but there are a few applications that make more significant attempts at natural media painting. Ambient Design ArtRage [2009] provides a set of brush-like tools that work for fun, casual painting. Corel Painter [2010] includes a “RealBristle” feature that creates textured strokes based on a set of brush-like parameters. While both of these products can be used to add a natural media feel to digital compositions, the experience is still not as dynamic or expressive as a real brush.

Three basic methods have been pursued to simulate the physics of brush shapes: forward dynamics, energy minimization, and data-driven approaches. Baxter et al.’s original work on brush simulation [2001] computed forward dynamics of a mass and spring system, which was used to deform a mesh. The mass and spring system was fundamentally limiting to the fidelity of brush behavior that could be reproduced, and required clever modeling for different types of brushes.

Later work by Baxter and Lin [2004], as well as Chu and Tai [2004] and Van Laerhoven and Van Reeth [2007], has focused on energy minimization as the method to simulate brush dynamics. While energy minimization provides good stiff behavior, it has not been used to directly simulate a large number of individual bristles for performance reasons.

Most recently, data-driven approaches have been explored. Xu et al. [2004] created bristle “macros”, where a clump of bristles was represented by a single macro that deformed according to data from off-line simulations. Baxter and Govindaraju [2010] collected deformation data from videos of real brushes, which they used to com-

[*]e-mail: steved@adobe.com
[†]e-mail: aravin@adobe.com
[‡]e-mail: hadap@adobe.com

VRST 2010, Hong Kong, November 22 – 24, 2010.
© 2010 ACM 978-1-4503-0441-2/10/0010 $10.00

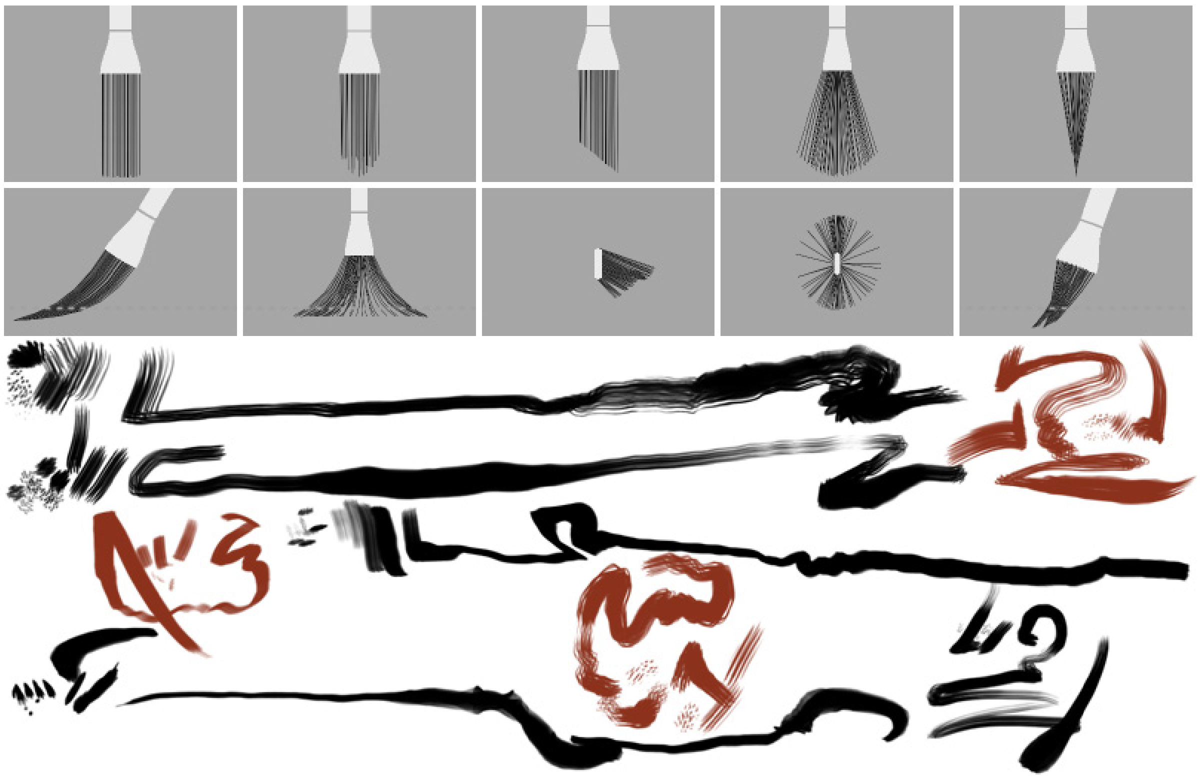


Figure 1: *The top row shows some of the different brush shapes we have modeled in our system. The middle row has typical deformations of those brushes, in average and extreme conditions, including arbitrary splitting of bristles. The bottom row contains exemplary strokes made with each of the brush types.*

pute the shape of a few splines that deform a brush mesh. Data-driven techniques are very fast, as they replace complex math with table lookups, but they are limited in the range of shapes they can reproduce, and adding additional behaviors either requires significantly more data collection (and larger associated tables), or complex algorithms for incorporating physics.

There have been two different brush shape representations so far as well: surface meshes and interpolated bristles. Mesh-based brushes – Baxter et al.'s, [2001], Chu and Tai's [2004], and Baxter and Govindaraju's [2010] – treat the brush as a single bulk surface, which is deformed by the underlying physical computations. Deformed meshes are very fast to render on modern graphics hardware, but the possible range of shapes that are representable by a mesh's constant-topology is very limited. These brushes can make smooth wet-paint strokes, but depend on texture mapping for dry or scratchy strokes, and do not support split or distressed brush stokes. Chu and Tai [2004] extended their mesh with hierarchical split tufts to recreate a limited form of some of these behaviors.

Rather than use a single brush mesh, interpolating approaches – Baxter and Lin's [2004] and Van Laerhoven and Van Reeth's [2007] – use brush physics to deform a set of pieces of geometry that represent individual brushes. Interpolation is much faster than simulation, enabling the appearance of a brush with many bristles. However, since the shape is dictated by a small number of spines, the possible range of shapes is still small. Furthermore, actual splits in the shape of the brush tip are not possible, as bristles will always be smoothly interpolated between spines, even when the real bristles may have formed multiple distinct clusters.

3 Brush Simulation

The first step in reproducing a brush stroke is computing the tip's changing shape during the stroke. We use direct numerical simulation of the dynamics of individual bristles as the underlying model. As far as we are able to determine, this is a first in brush simulation and it is the cornerstone of the quality of results and overall success of our painting tool. However, we were faced with seemingly incompatible simulation tradeoffs – accurate physical simulation of bristles versus real-time constraints from the usability point of view. The final details of our model are the result of many iterations of experimentation and feedback from professional artists at each stage of the design.

3.1 Dynamics of Individual Bristles

There are a number of very effective methods developed for the simulation of strand-like 1D primitives over the last decade. The various models roughly fall into three categories: a) a strand as a serial chain of rigid rods connected by spherical joints [2006], b) a strand as piecewise higher-order dynamic primitives (helices) [2006], or c) a strand as particles connected by constraints [2008]. Considering the robustness criteria of industrial strength brush simulation, all the three methods have potential to integrate the dynamics in a fully-implicit manner. However, the possibility of extremely efficient numerical simulation coupled with the relatively straightforward collision response model that can be used, led us to alternative (c). Specifically, we implemented the Discrete Elastic Rods paradigm from Bergou et al. [2008] as shown in Figure 2. This model is very effective in capturing all the nuances of bristle dynamics – non-straight rest shape, stiff bending and torsional elasticity, and inextensibility. In addition, the method efficiently eliminates the unnecessary modes of torsional dynamics by using a quasi-static assumption – at each time step, the torsion along the curve geometry is computed as an energy minimization problem.

In the Discrete Elastic Rods model, the total elastic energy of the bristle is comprised of [2008]

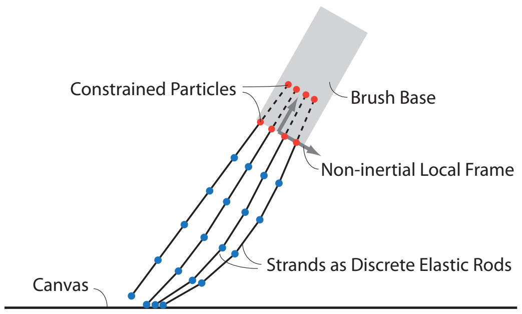


Figure 2: *A brush is modeled as a set of discrete elastic rods, simulated in a non-inertial local frame.*

$$\begin{aligned} E(\Gamma) &= E_{bend}(\Gamma) + E_{twist}(\Gamma) \\ E_{bend}(\Gamma) &= \frac{1}{2}\int (\boldsymbol{\omega} - \overline{\boldsymbol{\omega}})\, \overline{\boldsymbol{B}}\, (\boldsymbol{\omega} - \overline{\boldsymbol{\omega}})^T ds \\ E_{twist}(\Gamma) &= \frac{1}{2}\int \beta m^2 ds \end{aligned} \quad (1)$$

Here, Γ is the shape of the bristle, $\boldsymbol{\omega}$ is the curvature vector of the deformed bristle expressed in the material frame, $\overline{\boldsymbol{\omega}}$ is the curvature vector corresponding to the rest configuration of the bristle, m is the twist along the bristle, $\overline{\boldsymbol{B}}$ is the bending stiffness matrix, and β is the torsional stiffness.

We use the fully-implicit integration scheme of backward-Euler. Consider the state space of a single n-vertex bristle $\boldsymbol{x} \in \Re^{3n}, \boldsymbol{v} \in \Re^{3n}$. The implicit velocity integration is given by

$$\begin{aligned} \mathbf{v}_{n+1} - \mathbf{v}_n &= (\mathbf{I} - h\mathbf{M}^{-1}\frac{\partial \mathbf{f}}{\partial \mathbf{v}} - h^2\mathbf{M}^{-1}\frac{\partial \mathbf{f}}{\partial \mathbf{x}})d\mathbf{v} \\ &= h\mathbf{M}^{-1}(\mathbf{f}_n + h\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\mathbf{f}_n) \quad (2) \\ \mathbf{f} &= \nabla E(\Gamma) \quad (3) \end{aligned}$$

This results in a linear system that has a nice block tri-diagonal structure with a block size of 3×3, for our straight rest shape case. We extended the direct sparse linear solver for tri-diagonal systems from Davis [2006] to solve our block tri-diagonal problem. The direct sparse linear solver has a significant advantage over iterative solvers in terms of lower computational cost and memory bandwidth. No matter how "stiff" the linear system is, the direct sparse linear solver takes only one iteration for the solution, accessing the state space only once. The extremely efficient solver for the sparse linear system, along with carefully optimized constraint computations and their Jacobians, is the basis of real-time performance of bristle dynamics.

3.2 Collision Response

As the bristles only interact with planar geometry of the paper, collision detection is fairly straightforward. We perform continuous collision detection for each particle and the plane. The collision response and friction model is due to Bridson et al. [2002, 2003]. For the inextensibility constraint, we could have used linear springs with very high stiffness constant. The backward-Euler integration scheme is stable but not monotone, which means that we can get spurious oscillations in the case of extremely high forces. This is particularly true in the scenario where the bristles are smashed down vertically and buckle under collision response. Ultimately, instead of using stiff linear springs, we use the position and velocity projection methods of Provot [1997] and Goldenthal et al. [2007], along with the collision response model in an unified way as follows [2003].

$$\begin{aligned} \mathbf{v}^* &= \mathbf{v}^n + \frac{h}{2}\mathbf{M}^{-1}\mathbf{f}^{n+1} && \text{a) implicit half step} \\ d\mathbf{v}_n^* &= -\mathbf{v}_n^* && \text{b) plastic collision} \\ d\mathbf{v}_t^* &= \mathbf{v}_t^* - max(0, 1 - \mu\frac{d\mathbf{v}_n^*}{\mid \mathbf{v}_t^* \mid})\mathbf{v}_t^* && \text{c) friction} \\ \mathbf{v}^* &= \mathbf{v}^* + d\mathbf{v}_n^* + d\mathbf{v}_t^* && \text{d) velocity impulse} \\ \mathbf{x}^{n+1} &= \mathbf{x}^n + h\mathbf{v}^* && \text{e) explicit full step} \\ \mathbf{x}^{n+1} &= \mathbf{x}^{n+1} + \mathbf{x}_{corr} && \text{f) position correction} \\ \mathbf{v}^* &= \mathbf{v}^* + \frac{\mathbf{x}_{corr}}{h} && \text{g) velocity projection} \\ \mathbf{v}^{n+1} &= \mathbf{v}^* + \frac{h}{2}\mathbf{M}^{-1}\mathbf{f}^{n+1} && \text{h) implicit half step} \end{aligned}$$

We first estimate the velocities at the half time step $\mathbf{v}^*$ using implicit backward-Euler integration (step a). We then apply the velocity impulses due to collision response and friction (step b-d). Then we compute the candidate position update $\mathbf{x}_{n+1}$ from the corrected mid-point velocities (step e). The candidate positions typically violate the inextensibility and non-penetration constraints. We use an iterative inverse kinematics-like procedure [1997] to correct the positions in order to satisfy the constraints (step f). We reflect the change in positions into the half-step velocity (step g), and the remaining half implicit step is performed to compute the final velocities (step h).

3.3 Bristle Plasticity

Real brushes typically have bristles that have slightly wavy (non-straight) geometry, for which a straight rest shape of the strand would suffice. However, to mimic the plastic shape deformation of a wet brush volume, predominately resulting from the bristles-bristle adhesion, we considered using the bending elasticity of the non-straight rest shape that evolves using an artificial plasticity. To tryout the plasticity ideas, we implemented the fully-implicit bending energy formulation of non-straight rest shapes, as described in Bergou et al.'s later work [2010], in our individual strand simulation testbed.

For artificial plasticity, for each time step, if the local bending energy density is above certain threshold, we successively "yield" the rest configuration curvature $\overline{\boldsymbol{\omega}}$ towards the current curvature $\boldsymbol{\omega}$ according to the following formula.

$$\begin{aligned} \overline{\boldsymbol{\omega}} &= (1-\alpha)\,\overline{\boldsymbol{\omega}} + \alpha\,\boldsymbol{\omega}, \\ \exists &\quad (\boldsymbol{\omega} - \overline{\boldsymbol{\omega}})\,\overline{\boldsymbol{B}}\,(\boldsymbol{\omega} - \overline{\boldsymbol{\omega}})^T \geq \epsilon \end{aligned} \quad (4)$$

Here, α is the yield rate and ϵ is the yield threshold. In our initial experimentation, we found that this simple artificial plasticity model is quite effective in capturing plastic deformation of the bristle volume. However, brush plasticity poses unique workflow issues, such as whether and when to reset the bristles to their original shape, which artists thought would be more confusing than helpful. Ultimately, we opted for the increased performance and simpler dynamics of straight rest shape bristles for the current implementation. For the case of straight rest shape, the dynamics of individual bristle reduces to a totally torsion free formulation, resulting into very efficient implementation.

3.4 Local Dynamics

In order to attach the bristles to the base of the brush tip, we extend the bristles at the root by the addition of extra particles as shown

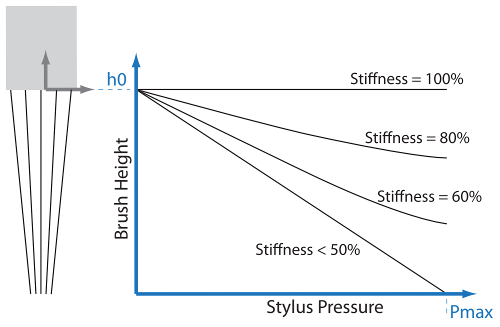


Figure 3: *The mapping between stylus pressure and brush height changes with bristle stiffness.*

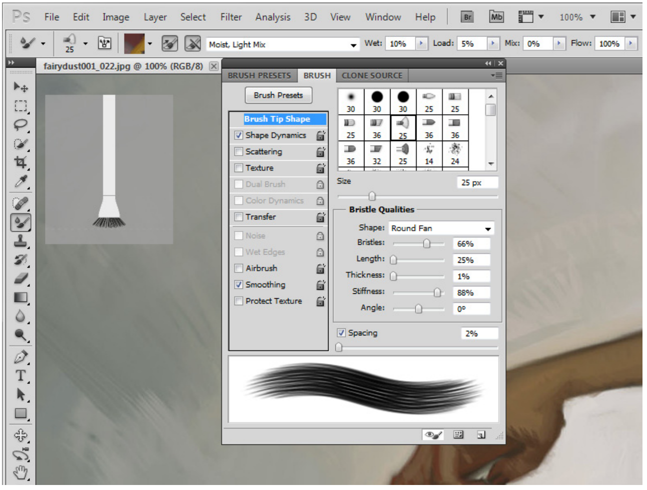
in Figure 2. The two particles at the base are constrained to the motion of the brush. If we carry out the dynamics in the global inertial frame, there will be very large forces exerted on "free" particles of the bristles due to the inextensibility constraints and bending/torsional stiffness. Even though the system could handle the "stiff" underlying dynamics due to its fully-implicit formulation for bending/torsional dynamics and projective method for inextensibility condition, we effectively avoid the situation altogether by computing forces in a non-inertial frame. This is similar to previous acceleration-less energy minimization methods [2004, 2004] and addresses one of the main limitations of forward dynamics for bristle simulation, achieving robust, stable behavior even under very brisk motion.

We transform the global body forces due to gravity, acceleration, and velocity (linear and angular) of the brush and position and orientation of collision geometry (the canvas) into the local frame attached to the base of the brush tip. We then carry out the bristle dynamics calculations in the non-inertial local frame. This way we can effectively control the extent of external forces that are applied to the bristles. In fact, we completely remove the acceleration (linear and angular) of the brush from the dynamics; in reality bristles are too stiff to visibly react to the brisk motion of the brush during normal brush strokes. We also remove the centripetal and Coriolis acceleration on each particle due to angular velocity of the brush. However, we include brush velocity (linear and angular) in the dynamics to capture the frictional dynamics with the paper.

3.5 Bristle-bristle Interaction

A significant factor in the bulk behavior of brush tips is the interaction among individual bristles due to collisions and stiction. Previous brush simulation work has attempted to recreate the behaviors through the explicit introduction of tip spreading [2004, 2010]. Unfortunately, in a brush tip model that simulates individual bristles, bristle-bristle interaction comes at a steep computational cost. Contacts grow quadratically with the number of bristles and even linear techniques such as continuum dynamics [2001] still add significant complexity. However, contrary to these results, in our initial brush simulation experiments (using a bristle as a serial chain of rigid rods model) we included bristle-bristle collision response and did not find a significant, perceptible difference in the quality of brush strokes.

Our key observation is that, when brushes are pressed into the canvas causing significant bristle-bristle interaction, the artist's intent is less precise, whereas for careful, fine strokes, bristle-bristle interaction is not a dominant effect. This seems counter-intuitive, as without collisions, bristles will cross through each other creating non-physical configurations. However, the brush tip's shape is not the goal here, but the resulting stroke, which looks comparable in the two scenarios. Therefore, because of the order of magnitude performance gain, we chose to simulate each bristle independently



Moist, Light Mix Wet: 10% Load: 5% Mix: 0% Flow: 100%

Size 25 px

Shape: Round Fan
Bristles: 66%
Length: 25%
Thickness: 1%
Stiffness: 88%
Angle: 0°

Spacing 2%

Figure 4: *A screenshot of the brush tip and paint mixing parameters as presented to the user in our commercial digital painting program.*

and forego bristle-bristle interaction.

3.6 User Interface

For simplicity and intuitiveness of the interface, the user is provided with a set of basic brush tip shapes to choose among: point, blunt, curve, angle, and fan, in either flat or round cross-section variants. These shapes were created in an offline modeling process that allowed for arbitrary bristle configurations. The user can then select settings for brush size (a uniform scale of the shape), brush length (scale along each bristle's axis), number of bristles, and bristle stiffness (see Figure 4). This set of exposed control parameters was chosen based on artist feedback. Bristles are generated using Poisson-disk Distribution inside the scaled shape to create the final brush tip.

To afford users intuitive control of the brush pose during a stroke, we directly map the position and orientation of a tablet stylus to the position and orientation of the virtual brush. This reproduces the full range of dynamic shape changes that users expect from working with real brushes. However, it is unclear how to best map the stylus pressure to the brush height with respect to the canvas. Artists pointed out that our original, simple one-to-one mapping made the bristle stiffness parameter useless because the same pressure applied to the stylus always resulted in the same deformation of the bristles, whereas a brush with very stiff bristles should be very difficult to get to deform at all. Therefore, we changed our pressure to height mapping to depend on the bristle stiffness, so stiffer bristles are deformed less at maximum stylus pressure (see Figure 3).

4 Paint Deposition

After the motion of the bristles has been computed via our physical model, the dynamic shape is used to determine the contact area on the canvas and to compute the bidirectional transfer of paint between the canvas and the brush.

4.1 Stroke Generation

While our physical bristle simulation is able to compute the dynamics of many more bristles in real time than previous methods, it is still not able to achieve the up to one thousand bristles found in a real brush. Additionally, different computer hardware has different maximum numbers of bristles that can be simulated, based on processor speed. However, more important than the exact number of bristles in recreating a brush stroke's appearance is the area of coverage by those bristles, so we maintain the coverage area by increasing bristle thickness as the number of bristles decreases. This way, one hundred bristle, fifty bristle, and ten bristle brushes all create the same bulk stroke effects, with a difference only in the quality of fine texture features. This allows the number of bristles to be treated as a tradeoff between simulation performance and brush stroke fidelity.

Conversely, previous work uses interpolated bristles to create a densely populated brush with low simulation cost – up to 200 in Baxter and Lin [2004]. The downside to interpolated bristles is twofold. First, they prevent proper brush tip splitting behavior, by enforcing a smooth distribution of bristles regardless of configuration. Real brush tips exhibit non-uniform bristle distribution through clumping. Second, a large number of interpolated bristles has a high rasterization cost which necessitates the use of the GPU. However, because of the difficulty in robust support for a diverse array of consumer GPUs, we cannot depend on hardware accelerated rasterization and must design for the CPU only case. Instead of many interpolated bristles, thick bristles are able to achieve similar bulk brush stroke quality and allow arbitrary brush tip splitting, with lower computational cost.

Users may also want to explicitly reduce the thickness of bristles without increasing their number, to create the appearance of dry, scratchy strokes. Since thick bristles yield strokes with full coverage, they have the appearance of very wet paint. Therefore, we leave relative thickness as a user parameter – see Figures 4 and 5.

Once the 3D positions of the bristle vertices are known, the canvas contact area is determined by rendering orthographically projected quad strips, as in Baxter and Lin [2004]. While many contact area stamps are necessary along a stroke to create a smooth appearance, physical simulation steps can be taken at a lower rate and bristle geometry can be interpolated for intermediate stamps. This creates piece-wise linear strokes similar to previous approaches. However, artists require smooth strokes (C2 continuity), normally achieved by bezier interpolation of input positions, rather than linear. This suggests bezier interpolation should be used for the bristle geometry to achieve sufficient quality, but our observation is that placing linearly interpolated bristle geometry along bezier interpolated input positions is acceptable and fast.

An additional advantage of the individual bristle model is that it enables vector output through bristle sweeping. A brush stroke can be represented as a set of transparent filled vector outlines, called “sweeps”, one per-bristle. A sweep is generated by computing the envelope around the set of instantaneous 2D poses of a bristle over the stroke. A bristle’s instantaneous 2D pose is an ellipse oriented so its furthest points align with the two extreme points of the bristle’s contact with the canvas and the minor axis is the bristle diameter. The envelope around these poses is then computed using Pudet’s method [1994]. Mesh-based brush models would not work with this technique because the contact area would need to be vectorized each stamp, and only a single sweep would be generated for the entire stroke resulting in flat shading rather than texture.

4.2 Bidirectional Transfer

During the brush stroke, as each instantaneous contact area is computed, the transfer of paint between the brush and canvas must be simulated. Through this mechanism, the effects of non-uniform brush paint load, brush drying, and brush dirtying are enabled. Baxter et al.’s [2001] work models paint load as reservoir and surface layers mapped on to the brush mesh. Paint on the canvas is mixed into the surface layer, which is kept full from the reservoir and then deposited at a constant rate. The downside of this approach is that regardless of how much paint gets picked up, the constant volume and constant deposition rate results in a constant length of dirty streak, whereas a real brush that picks up a large amount of paint should make a much longer stroke than a brush that picks up a small amout.

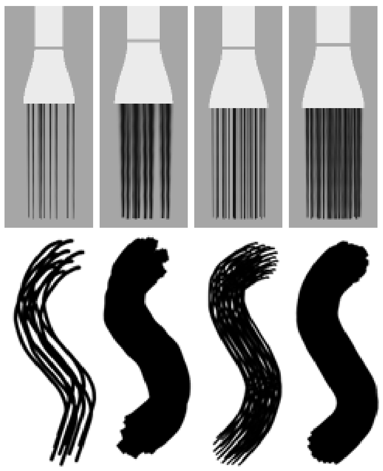


Figure 5: *Different brush tips and strokes with (from left to right) 10 bristles and 25% thickness, 10 bristles and 100% thickness, 40 bristles and 25% thickness, and 40 bristles and 100% thickness.*

To address this limitation, we replace the surface texture with a pickup buffer which starts out empty and is updated to store only the paint picked up off the canvas. Deposition is done by mixing the reservoir and pickup paint colors together just before applying to the canvas. In this way, the pickup and reservoir buffers are depleted independently, and so small amounts of picked up paint create short streaks while large amounts create longer streaks. As in the work of Chu et al. [2010], these buffers are aligned and resolution-matched with the canvas, rather than mapped to the bristles directly.

Ideally, to enable conservation of paint during and between strokes, the canvas should also be augmented with an F channel. This allows a brush stroke to pick up and put down the same paint repeatedly without changing the overall amount of paint on the canvas. However, if a canvas F channel is not available, each canvas pixel must always be treated as though it stores an infinite amount of paint.

Since our model is based on simple per-channel linear blending operations, it can work with arbitrary document colorspaces – typically RGB or CMYK, but also Lab or others, and with or without alpha. Furthermore, while blending math is carried out in 32-bit floating point, careful consideration of conversion and rounding allows canvas data of arbitrary bit-depth (generally 8, 16, or 32 bits per channel) to be manipulated without algorithmic changes. These considerations allow us to use SSE to implement the blending math, for a significant speed improvement.

To allow intuitive user control over the paint mixing process, we worked with artists to determine what paint stroke qualities it is most important to have direct control over. As a result, we expose three paint parameters: load, wetness, and mixing. Load determines the amount of paint stored in the reservoir buffer upon filling with paint, wetness specifies the amount of paint that is picked up from the canvas into the pickup buffer, and mixing represents the rate at which picked up paint is mixed with reservoir paint. See Figure 6.

5 Results

As we have worked extensively with professional artists on the integration of our brush model into commercial raster and vector painting applications, we have received many impressive pieces of artwork using the new tools. A test sheet showing a variety of different brush stroke and paint mixing effects can be seen in Figure 7. Some

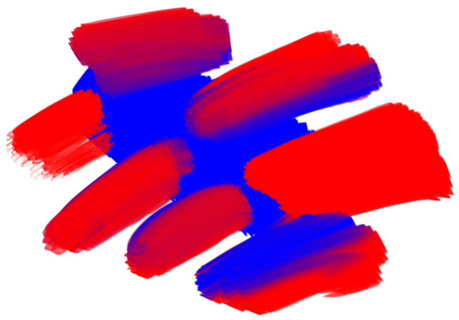


Figure 6: *A variety of red strokes with different paint mixing parameters applied on top of some blue paint, showing some of the many behaviors that are possible.*

examples of completed pieces, including both raster and vector output, can be seen in Figures 8 and 9.

5.1 Performance

The three main stages of our algorithm are physical simulation of the bristles, rasterization of the brush stamp, and compositing the stamp and canvas. Table 1 shows profile data from our raster application using a variety of different settings for a round point brush on a 300dpi RGBA canvas. Our test system is a MacBook Pro laptop with 2.6GHz Intel Core2 Duo processor running Mac OS X 10.5.8.

diam (mm)	**bristles**	**thick** (%)	**P** (ms)	**R** (ms)	**C** (ms)
2.5	50	1	0.5	0.29	0.31
5	50	1	0.51	0.44	0.84
10	50	1	0.46	0.97	2.8
10	50	50	0.54	1.8	3.2
10	50	100	0.53	2.3	3.7
10	25	1	0.25	0.78	2.8

Table 1: *Milliseconds per output stamp spent in each portion of brush stroke generation algorithm (**P**hysics, **R**asterization, **C**ompositing), on our test system for exemplary input.*

One physical simulation step of a single bristle with six vertices takes 30µs on average. That means up to 1000 bristles can be simulated on a single core at 30Hz, which is a significant improvement over existing bristle or spine simulation speeds. However, simulation steps are taken based on traversed canvas pixels, which is velocity dependent – for example, a typical quick stroke may cover five inches per second, which is 1500 pixels. At one physics step per 3 pixels, that is 500 physics steps, which means about 67 bristles can be simulated in that time. For comparison, Baxter and Lin [2004] report a four vertex spine solves in 100µs (increasing quadratically with the number of vertices). At 10µs for a six vertex bristle, Baxter and Govindaraju's [2010] data-driven approach could simulate more bristles but with a less expressive range of possible shapes.

Unfortunately, the difficulties in supporting GPU acceleration reliably over a wide variety of different graphics cards vendors, driver versions, and operating systems has ruled it out for the time being, forcing us to rely on CPU rasterization and compositing instead. The fillrate of our software rasterizer is 75Mpixels/sec. This ends up being a significant cost, despite being negligible on our GPU prototype.

Because of the high stamping rate required to get smooth strokes, the compositing cost, which largely consists of the paint transfer algorithm, also ends up being significant. For a typical circle stamp of 100x100 pixels with 7900 non-zero pixels to be composited, 1200 stamps can be composited per second on a single CPU core, or 0.83ms per stamp.

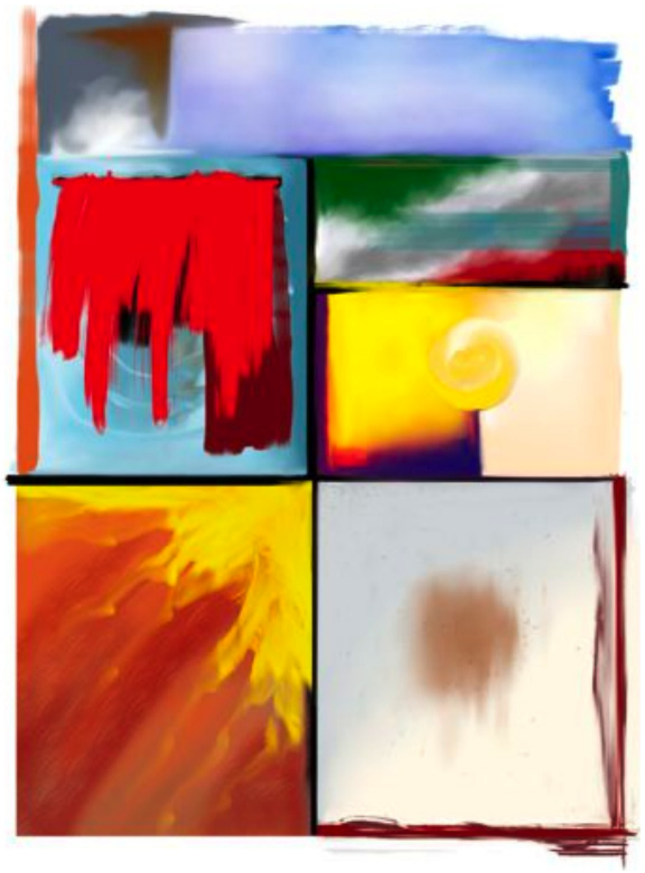


Figure 7: *Test sheet showing a variety of brush stroke and paint mixing effects.*

There are a variety of ways we are able to take advantage of multicore systems to improve our performance. We get the biggest improvement by pipelining the three algorithm stages so they can overlap one another. For a two-core machine, we put the physics on one core and the rasterization and composition on the second, which yields a nearly 2x speed improvement depending on the size of the brush and the number of bristles. For a larger number of cores, we can further pipeline rasterization with compositing, which gives another significant speedup for large brushes. Beyond pipelining, as the number of bristles continues to increase, there is a smaller benefit gained from distributing the different bristles' physical simulation across the cores. Larger stamps can also experience improved compositing cost by sending tiles of the stamp and canvas to different cores.

In general, we are able to achieve interactive painting rates for brushes of up to 100 bristles and sizes up to 25mm at 300dpi on average hardware. Newer processors such as the Intel Core i7 are much faster and can handle larger brushes.

6 Conclusions

We have presented a novel virtual 3D brush model that represents individual bristles at a faster speed and with a greater degree of flexibility than previous work. We have integrated this system into two commercial painting products, one raster and one vector, to demonstrate our model's robustness to real-world industry demands. Professional artists have successfully used these tools to create a wide variety of different styles of artwork, all with the appearance of nat-

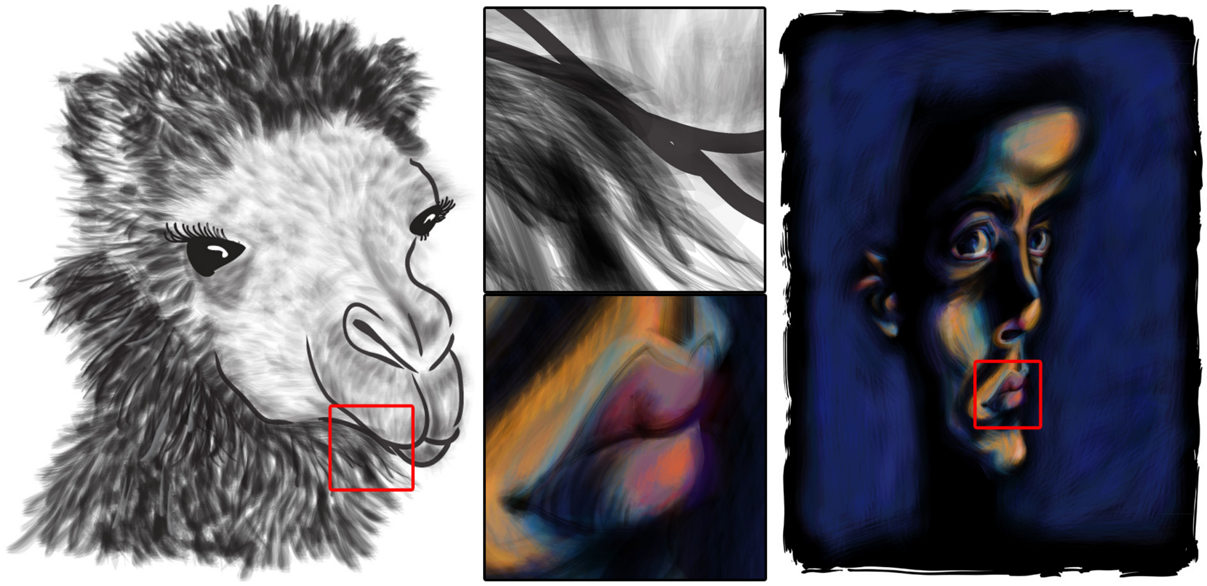


Figure 8: *Examples of vector output by professional artists. Insets are zoomed in to show resolution independence. Used with permission.*

ural media paintings.

In the future, we will continue to improve performance of our model, specifically by integrating GPU acceleration for the rasterization and compositing. That alone will enable an additional leap in performance, especially for large brushes. We also will consider the wide array of other effects seen in natural media artwork for our new tools, maintaining our emphasis on industrial-strength solutions that are usable by artists for professional work.

Acknowledgements

We would like to thank Miklos Bergou and Eitan Grinspun for their valuable support during our collaboration with University of Columbia on Discrete Elastic Rods and its implementation. We also specially thank the artists Mike Shaw, Daniel Presedo, Karen Tanner, John Derry, Greg Geisler, Craig Mullins, Olaf Giermann, and Scott Valentine for their feedback and artwork.

References

ARTRAGE, 2009. Ambient design. `http://www.artrage.com/`.

BAXTER, W., AND GOVINDARAJU, N. 2010. Simple data-driven modeling of brushes. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, 135–142.

BAXTER, W., AND LIN, M. 2004. A versatile interactive 3d brush model. In *Proceedings of the Pacific Conference on Computer Graphics and Applications*, 319–328.

BAXTER, B., SCHEIB, V., LIN, M., AND MANOCHA, D. 2001. Dab: Interactive haptic painting with 3d virtual brushes. In *Proceedings of ACM SIGGRAPH*, 461–468.

BERGOU, M., WARDETZKY, M., ROBINSON, S., AUDOLY, B., AND GRINSPUN, E. 2008. Discrete elastic rods. In *Proceedings of ACM SIGGRAPH*, 1–12.

BERGOU, M., AUDOLY, B., VOUGA, E., AND WARDETZKY, M. 2010. Discrete viscous threads. In *Proceedings of ACM SIGGRAPH*. To appear.

BERTAILS, F., AUDOLY, B., CANI, M.-P., QUERLEUX, B., LEROY, F., AND LÉVÊQUE, J.-L. 2006. Super-helices for predicting the dynamics of natural hair. In *Proceedings of ACM SIGGRAPH*, 1180–1187.

BRIDSON, R., FEDKIW, R., AND ANDERSON, J. 2002. Robust treatment of collisions, contact and friction for cloth animation. In *Proceedings of ACM SIGGRAPH*, 594–603.

BRIDSON, R., MARINO, S., AND FEDKIW, R. 2003. Simulation of clothing with folds and wrinkles. In *Proceedings of the ACM Symposium on Computer Animation*, 28–36.

CHU, N., AND TAI, C.-L. 2004. Real-time painting with an expressive virtual chinese brush. *IEEE Computer Graphics and Applications 24*, 5, 76–85.

CHU, N., BAXTER, W., WEI, L.-Y., AND GOVINDARAJU, N. 2010. Detail-preserving paint modeling for 3d brushes. In *Proceedings of the International Symposium on Non-Photorealistic Animation and Rendering*.

DAVIS, T. A. 2006. *Direct Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

GOLDENTHAL, R., HARMON, D., FATTAL, R., BERCOVIER, M., AND GRINSPUN, E. 2007. Efficient simulation of inextensible cloth. In *Proceedings of ACM SIGGRAPH*, 49.

HADAP, S., AND MAGNENAT-THALMANN, N. 2001. Modeling dynamic hair as a continuum. *Computer Graphics Forum 20*, 3.

HADAP, S. 2006. Oriented strands: dynamics of stiff multi-body system. In *Proceedings of the ACM Symposium on Computer Animation*, 91–100.

LAERHOVEN, T. V., AND REETH, F. V. 2007. Brush up your painting skills: Realistic brush design for interactive painting applications. *The Visual Computer 23*, 9, 763–771.

PAINTER, 2010. Corel. `http://www.corel.com/painter/`.

PHOTOSHOP, 2008. Adobe. `http://www.adobe.com/photoshop/`.

PROVOT, X. 1997. Collision and self-collision handling in cloth model dedicated to design garment. In *Proceedings of Graphics Interface*, 177–89.

PUDET, T. 1994. Real time fitting of hand-sketched pressure brushstrokes. *Computer Graphics Forum 13*, 3, 205–220.

SMITH, A. R. 2001. Digital paint systems: An anecdotal and historical overview. *IEEE Annals of the History of Computing 23*, 4–30.

XU, S., TANG, M., LAU, F., AND PAN, Y. 2004. Virtual hairy brush for painterly rendering. *Graphical Models 66*, 5, 263–302.

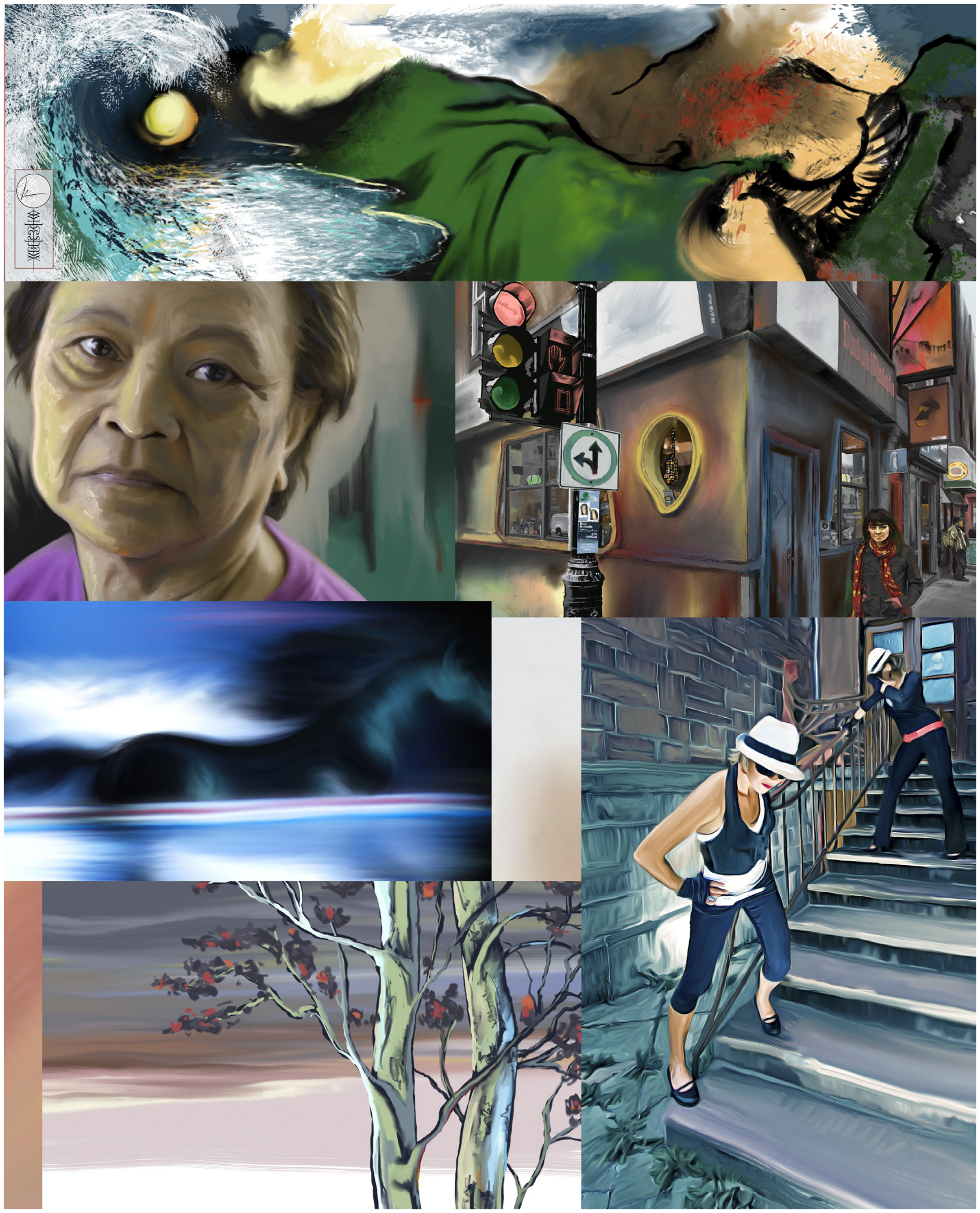


Figure 9: *Examples of raster output in a variety of styles by professional artists. Each piece was composed using different settings for the brush and mixing parameters to achieve the final appearances. Used with permission.*