The Visual Computer manuscript No.
(will be inserted by the editor)

Brush Up Your Painting Skills

Realistic Brush Design for Interactive Painting Applications

Tom Van Laerhoven, Frank Van Reeth

Hasselt University - Expertise Centre for Digital Media
transnationale Universiteit Limburg
Wetenschapspark 2, BE-3590 Diepenbeek, Belgium
e-mail: {tom.vanlaerhoven, frank.vanreeth}@uhasselt.be

Abstract Most present-day interactive paint applications lack the means of adequately capturing a user's gestures and translating them into realistic and predictable strokes, despite the importance of such a mechanism. We present a novel brush design that adopts constrained energy optimization to deform the brush tuft according to the user's input movement. It incorporates bidirectional paint transfer and an anisotropic friction model. The main advantage of our method is its ability to handle a wide range of brush tuft shapes that are animated using a freeform deformation lattice, which is associated with the tuft's geometry. This way, almost no conditions or limitations are placed upon the appearance of the brush. Examples range from round brushes modeled as polygon meshes, to flat brushes with individual bristles. Less common deformable tools that are used to apply or remove paint on the canvas, like sponges, can be created as well. The model is integrated in our interactive painting system for creating images with watery paint.

Key words I.3.4 [Graphics Utilities]: Paint systems I.3.5 [Computational Geometry and Object Modeling]: Physically based modeling G.1.6 [Optimization]: Constrained optimization

1 Introduction

The paint brush is the medium that communicates an artist's intentions onto the canvas. Its importance is pointed out by an experienced artist by stating that *"most of the materials involved in painting are expendable. Just about the only tools worth protecting are your paint brushes."* [24].

In digital paint applications the brush is equally important. As a subset of the "Artistic Rendering" research domain, which focusses on the creation of "pleasing" artwork, the interactive paint applications emphasize the role of the user in creating painted images. In consequence, a 3D virtual counterpart of a real paint brush must be a able to capture a user's gestures and translate them to predictable strokes, while preserving a natural look.

Literature has already brought forward several models that target these issues, for both Oriental and Western paint styles. Most currently available commercial paint programs, however, essentially ignore the nuances in brush motion and still produce uniform, analytical marks. Given the numerous advantages of a digital equivalent of the real painting process, it is clear that an application that combines a canvas model that can capture complex paint behavior with an equally capable brush model would be a valuable tool for even the experienced traditional artist.

This paper presents a new brush model that complements our previously introduced canvas model [23]. It features the following characteristics:

- An efficient and general applicable method to deform the brush in real-time, using free-form deformation;
- Anisotropic friction;
- Bi-directional pigment and water transfer;
- Complex footprint generation;

– Integration in a real-time physically-based simulation framework for painting with watery paint media.

Unlike previously proposed solutions, our method does not use a skinning-based technique that concentrates on a single tuft shape, but relies on deformation of the tuft's local coordinate system. This way, almost any geometry can be used to define its appearance, as shown in the example section.

The rest of this paper is organized as follows: after discussing existing work on digital paint brushes in the next section, we first look at important characteristics of a real brush before introducing our own design. Different brush constructions with accompanying stroke examples are shown in section 4, followed by conclusions and suggestions for future work in section 5.

2 Background

In this section we look at how virtual paint brushes have evolved from being simple automated rubber stamp procedures, to versatile designs that generate realistic paint strokes, and which genuinely respond to an artist's gestures. For extensive surveys on interactive paint applications, and non-photorealistic rendering research in general, we refer to literature [11,19,2,8].

The work of both Greene [12] and Strassmann [22] can be considered as the starting point of a long line of attempts to create a more realistic means of input for paint applications. Greene describes an input device for turning a physical draw action into a digital stroke, commonly referred to as *Greene's drawing prism* [12]. Rather than trying to create a new brush model in software, this special device processes input from real brushes.

Strassman is the first to present a physical model of brush movement on a canvas with the purpose of creating traditional Japanese artwork using black ink [22]. The brush is modeled as a one-dimensional array of idealized bristles, each carrying an amount of ink. The actual creation of a stroke involves the input of a number of control points with position and pressure information from which the final stroke is rendered.

Virtual brushes were considerably improved since the drawing prism and Strassman's one-dimensional version. The first physically-based 3D brush model was given by Lee, who adopted Hooke's law to model a collection of elastic bristles [13]. This is one of many advanced brush models proposed in literature, all intended for either producing paintings with Oriental ink or creating Chinese calligraphy.

In fact, only the work of Baxter *et al.* explores the use of virtual brushes in Western painting [4,1]. The deformable brush integrated in their dAb painting system is the first to provide haptic feedback, enabling a user to actually feel how the brush deforms and therefore enhancing the sense of realism. A linear spring between the brush head and the canvas generates the forces that form the input for a PHANToM haptic feedback device. The dynamics of the brush head itself is handled by a semi-implicit method that integrates linear spring forces. These kind of time-stepping integration techniques, however, are less suitable to be used in a heavily damped system that is required to simulate the stiff bristles. Further limitations include the inability to handle bristle splitting.

Saito *et al.* introduce a more appropriate technique based on energy optimization [17,18]. The function that has to be minimized captures the total amount of energy in the system, a summation of bend energy from joints, potential and kinetic energy from the tuft mass, and frictional energy. The result is a very stiff dynamical system where the static equilibrium is found almost instantly, which is a good approximation of real bristle behavior. The brush geometry is constructed by a single spine that is traced by a circular disc.

Several authors extended this technique. Chu *et al.* added anisotropic friction, lateral spine nodes to control brush flattening and a bristle spreading technique based on a static alpha map [6]. Their system also takes into account "pore resistance", which occurs when bristles get stuck in irregularities of the canvas. Plasticity accounts for shape deformations by internal friction of the wet tuft, and is modeled by adjusting the target angle with a small value. The brush surface is again determined by an elliptical cross section that is traced along the spine. Therefore, this brush model is only suitable for the round brushes found in Oriental ink painting. Lateron, they improve the splitting procedure by enabling the tuft to generate smaller child tufts along its spine [7].

In more recent work of Baxter *et al.*, a similar approach is generalized to a multi-spine architecture able to also model brushes used in Western painting [3]. This method, however, relies on two different implementations depending on the geometry: a subdivision surface modeling an explicit surface, or thin polygonal strips representing individual bristles.

A completely different approach is taken by Xu *et al.*, who design the brush as a set of independent "writing primitives" described by NURBS surfaces, each representing a collection of bristles [25].

Finally, Corel's state of the art application Painter X recently introduced the "RealBristle" technology, which mimics individual bristle behavior for a wide range of brush types [9]. Nevertheless, the resulting strokes still look sterile and artificial due to simplified underlying brush and canvas models.

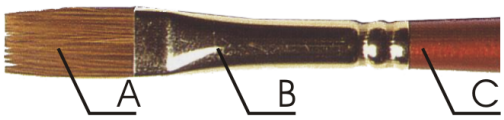


Figure 1 The three components of a paint brush: A. the tuft, B. the ferrule, C. the handle. Copyright ©1998 Smith [20].

Figure 2 Several commonly used real brushes, along with a representative stroke. (a) Chinese calligraphy brush; (b) flat brush; (c) round brush; (d) rigger brush; (e) fan brush; (f) mop brush. Copyright ©1998 Smith [20].

3 Method

3.1 Analysis of a Real Paint Brush

Figure 1 shows the main components of a real paint brush. Besides the handle and the ferrule, the most vital brush component is the tuft, which carries and applies the paint on the canvas. Its material, natural hair, bristle or synthetic fiber, determines the quality (and value) of the brush. Apart from its material, the effects a brush can produce are also influenced by its shape and size attributes, as depicted in figure 2. The figure lists a small selection out of the wide range of available brush shapes, each accompanied with a sample stroke.

Careful observation of the behavior of a round brush in the hands of an artist reveals that movement always occurs in a pull-motion. The bristles are almost never pushed along the canvas, except for applying small details like dots. The "snappy" bristle behavior makes that the brush almost instantaneously regains its shape when it is lifted from the canvas. Furthermore, it is clear that the brush is an extremely versatile tool that is able to create shapes ranging from thin detailed lines to broad strokes. Keeping these observations in mind, we outline both behavior and appearance of the virtual paint brush in the next sections.

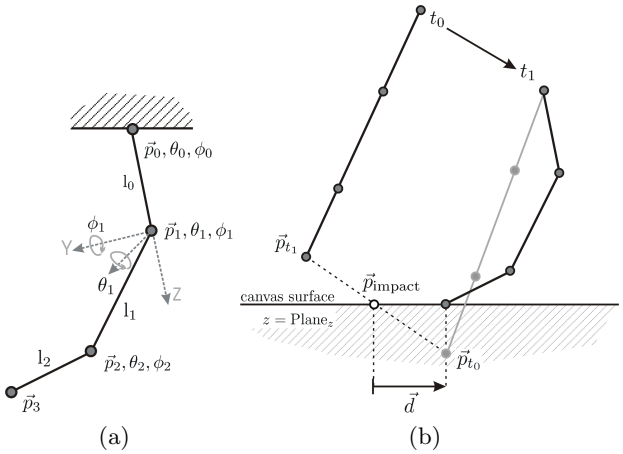


Figure 3 (a) kinematic representation of a bristle; (b) finding the drag vector in a 2D scenario where the bristle tip touches the canvas somewhere in the time interval $[t_0, t_1]$. The start point of the drag vector is calculated by interpolation, while the end point is approximated by projecting the joint position at t_1 on the canvas surface.

3.2 Brush Dynamics

In our system, the dynamics that govern the behavior of a single bristle match the approaches found in the work of other authors, like Saito, Chu and Baxter [17,6,7,3], using an energy optimization framework to compute the static equilibrium of the system. This approach results in very "snappy" bristle behavior we observed in the previous section. We will briefly revisit our interpretation of the technique in this section.

Before deriving the kinematic equations we first briefly describe the bristle representation. The notation used in this section is derived from the work of Baxter *et al.* [2].

A single bristle is represented as a kinematic chain, shown schematically in figure 3(a). Each segment in the chain has a predefined length and two angles, θ and ϕ, which determine the segment's orientation. We express the orientation in fixed XYZ angles representation, because it is intuitive to work with and because of its compact form. We assume that the final rotation around the global Z axis is always zero, which corresponds to a bristle with zero twist.

With this representation, we can transform a vector ${}^{i}\mathbf{v} = (\mathrm{x}, \mathrm{y}, \mathrm{z})$ in the coordinate frame of segment i to its parent coordinate frame $i-1$, using the combined XY rotation matrix ${}^{i-1}\mathbf{R}_i$:

$$
\begin{aligned}
{}^{i-1}\mathbf{v} &= {}^{i-1}\mathbf{R}_i \, {}^{i}\mathbf{v} \\
&= \mathbf{R}_{\mathrm{Y}}(\phi) \, \mathbf{R}_{\mathrm{X}}(\theta) \, {}^{i}\mathbf{v} \\
&= \begin{bmatrix} \cos\phi & \sin\phi\sin\theta & \sin\phi\cos\theta \\ 0 & \cos\theta & -\sin\theta \\ -\sin\phi & \cos\phi\sin\theta & \cos\phi\cos\theta \end{bmatrix} \begin{bmatrix} \mathrm{x} \\ \mathrm{y} \\ \mathrm{z} \end{bmatrix}
\end{aligned} \qquad (1)
$$

In this formula we adopted a notation that is used by most kinematic literature [10,16].

The actual bend angle β, the angle between two adjacent segments, is now determined by $\beta = \cos^{-1}(\cos\theta\cos\phi)$.

Energy Analysis We take on a rather "unphysical" approach to simulate the dynamics of a deformable bristle, by optimizing a behavior function $\mathbf{C}$ that captures the total energy in the system. The result is a bristle that immediately regains its shape, which actually closely mimics the behavior of a real bristle as observed in section 3.1.

In comparison, the physically-based method would convert the behavior function to a force equation that models a generalized spring, "pulling" the system in the desired state. Some explicit or implicit time-stepping algorithm would be used to integrate the equations, but most likely produce either inaccurate or unstable behavior [4].

The total energy in a system containing a single bristle is the sum of deformation energy (the potential energy stored in the angular springs) and frictional energy:

$$\mathbf{C} = \mathbf{E}_{\text{total}} = \sum_{\text{joints}} (\mathbf{E}_{\text{spring}}) + \mathbf{E}_{\text{friction}}. \tag{2}$$

The angular spring pulls the bend angle β between two adjacent segments towards a rest angle. When assuming straight bristles, this rest angle is always 180°. An angular spring, modeled as a scalar potential energy function, is based on Hooke's law, $\mathbf{E}_{\text{spring}} = \frac{\text{k}}{2}(180° - \beta)^2$, where k is the spring constant for that particular spring.

The friction energy term captures the energy caused by the bristle being dragged on the rough canvas surface. Similar to the work of both Chu and Baxter, a simple but efficient Coulomb friction model is used for this purpose:

$$\mathbf{E}_{\text{friction}} = \mu \sum_{\substack{\text{contact} \\ \text{joints}}} |\mathbf{N}|\|\mathbf{d}\|, \tag{3}$$

with μ the kinetic friction coefficient, N the force normal to the contact surface, and $\mathbf{d}$ the drag direction of the joint projected onto the surface (figure 3(b)), accumulated for every joint in contact with the surface.

Adding an *anisotropic friction* component to equation 3 accounts for the fact that the direction of minimal resistance is the "pull" direction of the bristle, as opposed to sideway dragging or bristle pushing (when the bristle becomes stuck in the canvas pores). An approach inspired by the Blinn-Phong formula for calculating the intensity of a specular highlight provides the desired result [3]:

$$\mathbf{E}_{\text{friction}} = \mu \sum_{\substack{\text{contact} \\ \text{joints}}} (1 - \eta)|\mathbf{N}|\|\mathbf{d}\|, \tag{4}$$

with $\eta = C_\eta \max\left(0, \mathbf{d}_p \cdot \frac{\mathbf{d}}{\|\mathbf{d}\|}\right)^k$, and d_p the preferred drag direction. The anisotropic constants $0 \leq C_\eta \leq 1$ and k determine the shape of the anisotropic cone. As also $0 \leqslant \eta \leqslant 1$, this addition to the equation effectively scales the friction energy in favor of the preferred direction, in which the anisotropic component removes all friction.

An important advantage of equation 4 is that it has $C1$ continuity, which is a requirement for the optimization method described in the next section.

One assumption we make is that the normal force $\mathbf{N}$ is a constant, and approximated at each time step. This simplification is done because calculating the actual normal force based on the configuration of all springs is tedious and it does not noticeably improve results.

Constraints To model the impenetrable canvas surface, an inequality constraint, for each joint, suffices: $\text{Plane}_z - (\mathbf{p})_z \geq 0$. The constraints force each joint $\mathbf{p}$ to stay above the surface described by $z = \text{Plane}_z$, with the Z-axis pointing into the canvas.

In practice, we replace the constraint with an equality constraint for every joint that violates the non-penetration constraint during the current time step: $\text{Plane}_z - (\mathbf{p})_z = 0$. This avoids the scenario where the optimizer decides that lifting the joint from the canvas is a solution that requires less energy than undergoing the larger frictional energy. In that case, the bristle would jump across the canvas.

Energy Optimization Having modeled the system's energy, we have enough information to find its equilibrium: the state of balance in which all the forces acting on the bristle are balanced. This state is defined at the energy minimum.

For this purpose we rely on the "donlp2" optimization framework [21]. This software package minimizes a (in general nonlinear) differentiable real function f, which is subject to (in general nonlinear) inequality and equality constraints g, h (equation 5). It accomplishes this by using sequential quadratic programming (SQP), an effective numerical method for nonlinearly constrained optimization [14].

$$\begin{aligned} f(x) &= \min_{\text{x}\in\text{S}} \\ S &= \{x \in \mathbb{R}^n : h(x) = 0, g(x) \geq 0\}. \end{aligned} \tag{5}$$

Mapping these equations to our model, f is the energy function $\mathbf{E}_{\text{total}}$, and the constraint functions were described in the previous section. Both the energy function and the constraint functions are non-linear in variables θ and ϕ.

3.3 Brush Geometry

Simulating every single bristle of a brush with the technique described above is not feasible, as a real paint

brush can contain hundreds of bristles. The established modus operandi in this case is to simulate just a few bristles that dictate the tuft behavior, traditionally using a skinning-based technique.

Regarding this issue, Baxter *et al.* choose a Butterfly subdivision scheme, and in later work a Catmull-Clark surface to tie a monolithic geometric model to a single spine. This requires a converter application that determines the subdivision control vertices. The brush geometry in the work of Chu *et al.* is obtained by sweeping an elliptical shape along a single spine.

We design a polygon mesh of a brush tuft in an undeformed state with an arbitrary 3D modeling application, in our case the freely available open-source tool *Blender* [5]. While painting, the tuft geometry is first embedded in a tightly enclosed free-form deformation (FFD) lattice described by a number of control points [16]. The parallelepiped-shaped lattice imposes a local coordinate system on the object, which is then deformed by manipulating the control points based on movement of the kinematic chain from the previous section.

Free-form deformation can be a computationally expensive task when used in combination with a detailed polygon model. For this reason, a vertex shader was created in NVidia's Cg language that deforms each polygon vertex using programmable graphics hardware [15].

Single spine A first brush design uses a single kinematic chain that serves as the tuft's *spine*, which maps its movement on the lattice control points on all four vertical sides of the parallelepiped. This setup is depicted in figure 4. Note that the control points of the deformation volume are allowed to penetrate the canvas. This is necessary because the brush hovers slightly above the canvas surface, and only a small amount of tilt pushes the control points below the surface. The polygon mesh of the thin round brush design in figure 4 is defined closely around the spine, so the small amount of penetration is hardly noticeable.

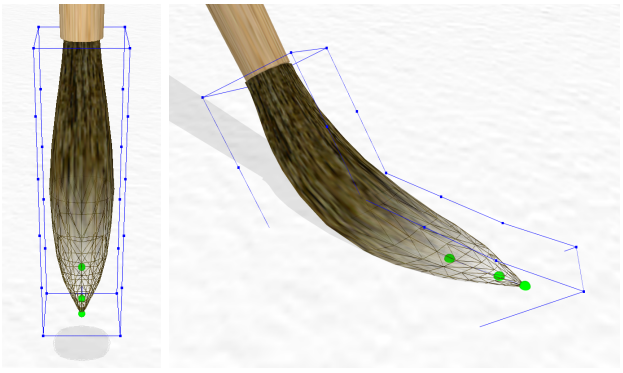


Figure 4 A single-spine round brush model. A free-form deformation grid, shown in blue, is associated with the tuft's polygon mesh (partially textured). The grid is manipulated by the kinematic chain formed by the green joints.

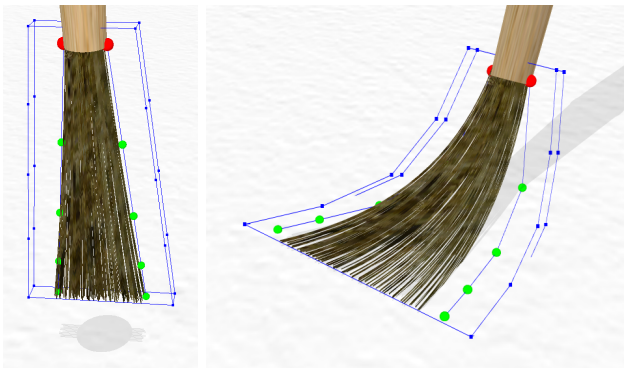


Figure 5 A flat brush model with two spines. The spines each manipulate one side of the free-form deformation grid. The brush geometry itself consists of a few hundred polylines.

Although bristle spreading can not be achieved with this single-spine design, it is versatile enough to allow for various tuft shapes, like the deformable sponge from figure 7(f).

Multiple spines The concept of associating a deformable spine, represented by a kinematic chain, with a FFD lattice can be extended to work with multiple spines. Figure 5 shows a flat brush design with two spines, each coupled to one side of the deformation lattice. If the user applies pressure to brush, the spines will spread the tuft's geometry.

In this particular example the brush was modeled by means of a procedurally generated polyline mesh, rendered using a few hundred OpenGL line strips. Unfortunately, the width of an OpenGL line strip can solely be specified in screen space, as a (floating-point) number of pixels. Bristle width will only be important when creating the brush's footprint on the canvas, however, as this directly influences the simulation (section 3.4). Based on the dimensions of the orthographic view volume used in this process, it is possible to calculate the necessary image space width given a bristle width in object space. The appearance of the brush as perceived by the user in the painting environment (using a perspective view) is of less importance, so in this case drawing the bristles in image space does not disturb the simulation.

Continuing this approach, more spines could be added and the resolution of deformation grid could be increased. Multiple spines can easily be simulated without noticeable performance drop. For the results in section 4, no more than two spines were used, however, because that particular configuration provides the desired spreading and scratching effect. Adding more spines did not result in noticeably better looking results.

Texture purpose	Content
Active pigment set 1	$[p_1, p_2, p_3, p_4]$
Active pigment set 2	$[p_5, p_6, p_7, p_8]$
Water quantities	$[w$, capacity, unused, unused$]$
Footprint	[on/off, t_x, t_y, unused]

Table 1 Three floating point texture objects are used to embody the brush's paint reservoir. A fourth texture stores the tuft footprint.

3.4 Bi-directional Paint Transfer

Paint consists of a mixture of pigment and water, and is being transfered back and forward between the brush and the canvas during painting.

Our canvas model for the simulation of complex watery paint media, like watercolor, gouache and Oriental ink, was introduced in previous work [23]. It consists of three layers, each modeled as a 2D grid of cells in which different rules and algorithms govern paint behavior. The *fluid layer* describes the movement of water and pigment on top of the canvas using fluid dynamics algorithms. The *surface layer* captures pigment particles that settle in the canvas surface irregularities, while the *capillary layer* embodies the internal canvas structure. The data of each layer is stored in floating point texture objects. Furthermore, the model features the Kubelka-Munk diffuse reflectance model to composite each layer of paint, and produce the final image. Every step of the simulation relies entirely on programmable graphics hardware to enable an interactive simulation rate. For detailed information on this topic, we refer to literature [23].

To represent the paint quantities inside the brush, floating point texture objects are used, similar to the representation of paint quantities on the canvas. Together they form the brush's *paint reservoir*. For this purpose we use three texture objects, storing a maximum of eight active pigment quantities $p_{1\ldots8}$, the amount of water w, and the reservoir's capacity at that point, as shown in table 1. A fourth texture keeps the most recent footprint of the brush tuft on the canvas.

Transferring paint, between the brush and the canvas now requires four steps:

1. Determine tuft footprint.
2. Determine paint transfer between brush and canvas.
3. Update paint quantities on the canvas.
4. Update paint quantities in the reservoir.

Determining the area on the canvas surface that is touched by the brush is straightforward. Rendering the tuft on the stencil buffer from the canvas' point-of-view, using an orthographic projection results in the desired footprint. The front clipping plane is set just below the canvas surface (we allow the brush to slightly penetrate the surface), while the back clipping plane is placed just above the surface. All geometry that is contained in this viewing volume contributes to the footprint. A fragment shader is used to retain the original 2D texture coordinates (t_x, t_y) from the tuft's reservoir in the footprint. These identify the origin of a footprint cell, which is necessary for the final update step.

```
for each contacting (reservoirCell, canvasCell) pair:
  rq = reservoirCell.quantity();
  cq = canvasCell.quantity();
  toCanvas = downRate*rq;
  toReservoir = upRate*cq;

  // verify available space in canvas cell
  if (toCanvas + cq > cellCapacity)
    toCanvas = cellCapacity - cq;

  // verify available space in reservoir cell
  if (toReservoir + rq > reservoirCapacity)
    toReservoir = reservoirCapacity - rq;

  canvasCell.add(toCanvas - toReservoir);
  reservoirCell.add(toReservoir - toCanvas);
end;
```

Table 2 Pseudo-code for determining bi-directional pigment and water transfer between reservoir and canvas cells.

Table 2 outlines the procedure for transferring pigment and water amounts between brush reservoir and the fluid layer of the canvas, while ensuring mass conservation. Two parameters, `downRate` and `upRate`, govern the net transfer rate.

The two last statements in table 2 execute steps three and four respectively. The footprint resides in the reservoir's projected space, which equals canvas space. To update the reservoir quantities, however, each cell of the footprint has to be projected backwards on the reservoir texture. This operation is performed using a vertex shader that relies on the retained texture coordinate.

A stroke on the canvas can now be created by repeatedly executing these four paint transfer steps on each position interpolated between coordinates sampled from the input device.

4 Results

The techniques outlined in this work are implemented in C++ and NVidia's Cg on a Pentium D, 2.8 GHz, equipped with a NVidia GeForce 6800. They were integrated in our previously introduced system for interactively creating images with watery paint, like watercolor, gouache and Oriental ink [23]. The setup incudes a Wacom tablet interface that provides 5DOF and enables a user to control the position, pressure and tilt of the brush in an intuitive way.

The resulting application performs at an interactive frame rate. It features an intuitive user interface that,

(a) (b)

Figure 6 A 3D view on the brush and canvas (a), and a palette for interactive color mixing (b).

while painting, enables the choice between three different views on the canvas: a 3D perspective view (figure 6(a)), a 2D orthographic view, and a brush-following view that attaches the camera to the brush itself. The latter is less useful during painting and mainly used for demonstration purposes. A separate palette dialog enables mixing paint and brush loading (figure 6(b)).

The efficient balance between CPU load, performing energy optimization, and GPU load, deforming and rendering the brush's geometry, as well as the canvas simulation, ensures an overall real-time painting simulation.

The sample strokes in figure 7 show the results of interactive brush movement with various kinds of brushes, using both watercolor paint and Oriental black ink. The geometry of each brush consists of either a polygon mesh modeled with the open-source tool Blender [5] and exported to the 3ds file format, or a procedurally generated OpenGL mesh of hundreds of polylines as individual bristles.

In both flat and round brush models, the spring constants between consecutive segments decrease near the tip. Each kinematic chain consists of at least four segments with decreasing lengths towards the tip. This makes the tuft tip more flexible than the top, which is stiffer because the bristles are tightly packed in the ferrule, and enables a brush to draw fine strokes (figure 7(k)).

A fan-shaped flat brush was used to create the scratchy strokes in images 7(d), 7(e) and 7(g). Finally, the sponge features large reservoir capacity and `upRate` values, making it possible to soak up wet paint from the canvas.

The process of painting with the virtual brushes was evaluated by various users, among which several experienced artists. The close resemblance to the behavior of a real brush made that almost no instructions were needed, and within seconds natural looking strokes appeared on the canvas, in accordance with the user's intentions. The sponge especially was rated very positive for its ability to fill large areas quickly.

5 Conclusions and Future Work

We outlined the construction of a 3D deformable brush that combines well with our previously introduced canvas model, supplying complex tuft footprints and bi-directional paint transfer. The brush's deformation depends on one or more kinematic chains, which participate in an optimization framework that computes the system's static equilibrium. The geometry of the brush consists of a 3D model description, either polygons or polylines. A free-form deformation lattice, which encloses the geometry and which is deformed using the kinematic chains, ensures the geometry of the brush inherits the deformation. Additionally, canvas friction is taken into account to enhance the realism of brush movement. Several brush types were created using this approach, ranging from a round and flat brush to a sponge. Strokes created with these brushes show that, in combination with a 5DOF tablet interface, complex prints can be produced. Furthermore, evaluation by several users indicated that its expressiveness exceeds that of existing systems.

There is some room for improvement of this model. The effects of plasticity and pore resistance are ignored at the moment. For both these issues, however, adequate solutions that can easily be incorporated in our model already exist in literature.

Acknowledgements We gratefully express our gratitude to the European Fund for Regional Development (EFRD), the Flemish Government and the Flemish Interdisciplinary institute for Broadband Technology (IBBT), which are kindly funding part of the research at the Expertise Centre for Digital Media. Many thanks also go to Marie-Anne Bonneterre and Josee Xavier for their artistic contribution.

References

1. Baxter, W.V.: Notes on brush simulation with optimization. Technical report, University of North Carolina at Chapel Hill, Department of Computer Science (2004)
2. Baxter, W.V.: Physically-based modeling techniques for interactive digital painting. Ph.D. thesis, University of North Carolina at Chapel Hill, Department of Computer Science (2004)
3. Baxter, W.V., Lin, M.C.: A versatile interactive 3D brush model. In: Proceedings of the 12th Pacific Conference on Computer Graphics and Applications, pp. 319–328. IEEE Computer Society Press, Seoul, Korea (2004)
4. Baxter, W.V., Scheib, V., Lin, M.C., Manocha, D.: dAb: interactive haptic painting with 3D virtual brushes. In: E. Fiume (ed.) Proceedings of ACM SIGGRAPH 2001, pp. 461–468. ACM Press, NY, USA (2001). DOI http://doi.acm.org/10.1145/383259.383313
5. Blender Foundation: Blender v2.41 (Software package). Available at http://www.blender.org. Blender Foundation (2006)
6. Chu, N.S., Tai, C.L.: An efficient brush model for physically-based 3D painting. In: Proceedings of the 10th

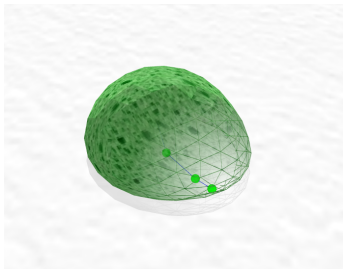
(a) Smearing paint with a clean round brush.

(b) Smearing paint with a clean round brush.

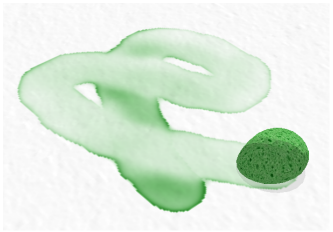
(c) Scratchy brush strokes.

(d) Scratchy brush strokes.

(e) Scratchy brush strokes.



(f) A deformable sponge, using a single spine.

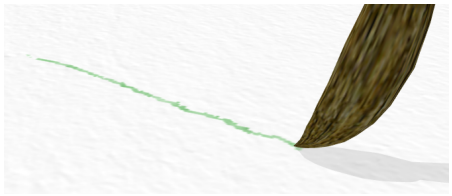


(g) Using a sponge to deposit paint.

(h) Mixing paint with a round brush.

(i) Using a flat brush.

(j) Bristle spreading with black ink.



(k) Drawing a very fine stroke with the flexible tip of a round brush.

Figure 7 Computer-generated sample strokes.

Pacific Conference on Computer Graphics and Applications, p. 413. IEEE Computer Society (2002)

7. Chu, N.S.H., Tai, C.L.: Real-time Painting with an Expressive Virtual Chinese Brush. IEEE Computer Graphics and Applications **24**(5), 76–85 (2004)
8. Colomosse, J.P.: Higher Level Techniques for the Artistic Rendering of Images and Video. Ph.D. thesis, University of Bath (2004)
9. Corel Painter X (Software package), http://www.corel.com/painterx. Corel (2006)
10. Craig, J.C.: Robotics. Addison-Wesley, New York (1989)
11. Gooch, A., Gooch, B.: Non-photorealistic rendering. A K Peters, Ltd. (2001)
12. Greene, R.: The drawing prism: a versatile graphic input device. In: Proceedings of the 12th annual conference on Computer graphics and interactive techniques, pp. 103–110. ACM Press, NY, USA (1985). DOI http://doi.acm.org/10.1145/325334.325202
13. Lee, J.: Physically-based modeling of brush painting. In: Proceedings of the fifth international conference on computational graphics and visualization techniques on Visualization and graphics on the World Wide Web, pp. 1571–1576. Elsevier Science Inc. (1997). DOI http://dx.doi.org/10.1016/S0169-7552(97)00073-1
14. Nocedal, J., Wright, S.J.: Numerical optimization. Springer Science+Business Media (1999)
15. NVIDIA: Cg toolkit user's manual, v1.4.1 (2006)
16. Parent, R.: Computer animation – algorithms and techniques. Morgan Kaufmann, San Fransisco (2002)
17. Saito, S., Nakajima, M.: Physics-based brush model for painting. In: Conference Abstracts and Applications of ACM SIGGRAPH 1999, p. 226 (1999)

18. Saito, S., Nakajima, M.: 3D Physics-based brush model for interactive painting (in Japanese). Jyouhou-Shori Gakkai Ronbushi (Japanese journal) **41**(3), 608–615 (2000)
19. Smith, A.R.: Digital paint systems: An anecdotal and historical overview. IEEE Annals of the History of Computing **23**(2), 4–30 (2001)
20. Smith, S.: The complete watercolour course, second edn. Collins & Brown (1998)
21. Spellucci, P.: DONLP2 Users guide. Technical University at Darmstadt, Germany (2004)
22. Strassmann, S.: Hairy brushes. In: Proceedings of the 13th annual conference on Computer graphics and interactive techniques, pp. 225–232. ACM Press, NY, USA (1986). DOI http://doi.acm.org/10.1145/15922.15911
23. Van Laerhoven, T., Van Reeth, F.: Real-time simulation of watery paint. In: Journal of Computer Animation and Virtual Worlds (Special Issue CASA 2005), vol. 16:3–4, pp. 429–439. J. Wiley & Sons, Ltd. (2005)
24. Wenz-Denise, S.: The invaluable paintbrush. World Wide Web, http://www.passionforpaint.com (2001)
25. Xu, S., Tang, M., Lau, F.M., Pan, Y.: A solid model based virtual hairy brush. In: Proceedings of Computer Graphics Forum, vol. 21 (2002)

dr. Tom Van Laerhoven
Tom Van Laerhoven is a senior researcher in computer science at Hasselt University (UHasselt) in Diepenbeek, Belgium. He obtained a MS in computer science in 2000 at UHasselt (formerly LUC). In 2006, he finished his PhD entitled "An Extensible Simulation Framework Supporting Physically-based Interactive Painting" at the Expertise centre for Digital Media (EDM), a research institute of Hasselt University. His research activities are concerned with computer animation, physically-based modeling and animation, non-photorealistic rendering and parallel and distributed algorithms.

Prof. dr. Frank Van Reeth
Frank Van Reeth is Professor of computer science at Hasselt University (UHasselt) in Diepenbeek, Belgium. He is deputy managing director of the Expertise centre for Digital Media (EDM) at UHasselt. He obtained a MS in computer science in 1987 at the Free University of Brussels, and a PhD in computer science at UHasselt (formerly LUC) in 1993. His research interests include computer graphics, computer animation, networked virtual environments, human computer interaction and multimedia technology. He published over 100 scientific papers in the above domains. He is a member of ACM, the Computer Graphics Society (CGS), Eurographics and IEEE.