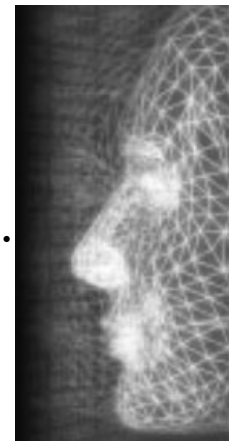
COMPUTER ANIMATION AND VIRTUAL WORLDS
Comp. Anim. Virtual Worlds 2005; **16:** 429–439
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cav.95

Natural Phenomena and Special Effects



Real-time simulation of watery paint

By Tom Van Laerhoven and Frank Van Reeth*

Existing work on applications for thin watery paint is mostly focused on automatic generation of painterly-style images from input images, ignoring the fact that painting is a process that intuitively should be interactive. Efforts to create real-time interactive systems are limited to a single paint medium and results often suffer from a trade-off between real-timeness and simulation complexity. We report on the design of a new system that allows the real-time, interactive creation of images with thin watery paint. We mainly target the simulation of watercolor, but the system is also capable of simulating gouache and Oriental black ink. The motion of paint is governed by both physically based and heuristic rules in a layered canvas design. A final image is rendered by optically composing the layers using the Kubelka–Munk diffuse reflectance model. All algorithms that participate in the dynamics phase and the rendering phase of the simulation are implemented on graphics hardware. Images made with the system contain the typical effects that can be recognized in images produced with real thin paint, like the dark-edge effect, watercolor glazing, wet-on-wet painting and the use of different pigment types. Copyright © 2005 John Wiley & Sons, Ltd.

KEY WORDS: paint systems; physically based modeling; non-photorealistic rendering

Introduction

Creating a digital equivalent of the traditional painting process has several advantages. The possibility of experimenting with various techniques and painting media with control over aspects, such as drying time, undoing mistakes, saving intermediate results, and the ability of introducing a wide range of digital tools, makes a painting system a valuable tool for both novices and experienced artists.

Due to the complexity of this process, however, it is a challenging task to simulate all this in real time. Existing work reveals that numerous problems remain in visual results, as well as in the creational process itself. In both Western and Oriental versions of digital painting quite realistic results can be obtained, but most often at the expense of a tedious, non-intuitive way of creating them. User input and rendering usually occur in separate stages of the simulation process, creating a mismatch between what the system delivers and what the artist actually had in mind. Our work targets precisely these problems.

Figure 1. A computer-generated watercolor image.

*Correspondence to: T. V. Laerhoven, Hasselt University—Expertise Centre for Digital Media and Transnationale Universiteit Limburg, Wetenschapspark 2, BE-3590 Diepenbeek, Belgium. E-mail: tom.vanlaerhoven@uhasselt.be

Contract/grant sponsor: European Research Project 'Custodiev'; contract/grant number: IST-2001-37116.

Copyright © 2005 John Wiley & Sons, Ltd.

Main Contributions

We introduce a new canvas model for the interactive simulation of thin, watery paint in real time. It is the first model that provides real-time painting experience with watercolor paint, while comprising sufficient complexity to capture its complicated behavior, its interactions with the canvas as well as its chromatic properties. We extend our previous work on a parallel implementation[1] with a new approach, suitable for graphics hardware, the Kubelka–Munk diffuse reflectance model and the capability to produce paintings with paint media related to watercolor, like gouache and Oriental black ink.

Background

Painting systems that capture the complexity, variety, and richness of painting media only recently began to appear in literature. Cockshott identifies the main problem of existing painting systems as being the lack of sparkle in the images they produce when compared to those made by traditional methods and media.[2] He claims this is caused by the shallowness of painting models and the lack of understanding the process of real painting and the behavior of paint. His own model, based on a cellular automaton, therefore includes rules accounting for surface tension, gravity, and diffusion. At the same time, Small introduces a canvas model for watercolor, again based on the cellular automaton principle.[3] Prior to their work, the actual painting process was mostly limited to the rendering of a brush imprint, with notable results by Greene's drawing prism and Strassmann's hairy brush.[4,5]

Curtis *et al*.[6] adopt a more sophisticated paper model and a complex shallow layer simulation for creating watercolor images, incorporating the work on fluid flows by Foster and Metaxas.[7] The painting consists of an ordered set of translucent glazes or washes. The individual glazes are rendered and composed using the Kubelka–Munk equations to produce the final image.[8] Their model is capable of producing a wide range of effects from both wet-in-wet and wet-on-dry painting. Fluid flow and pigment dispersion are again realized by means of a cellular automaton. In fact, cellular automata[9] and related techniques play an important role in the work of many authors.[2,6,10,11]

All of the above work, however, is more related to automatic rendering than to the interactive painting experience, mostly because the computational complexity did not allow real-time processing.

Baxter *et al*. are the first to present a fully interactive physically-based paint simulation for thick oil-like painting medium.[12,13] The IMPaSTo application is a mature painting system, exploiting graphics hardware for both the physical simulation of paint flow and the rendering with an interactive implementation of the Kubelka–Munk diffuse reflectance model. As the system is using an advection scheme to model the paint dynamics it neglects ambient behavior of the paint medium like diffusing, running, and dripping effects, which is our main objective in this paper.

The possibility to create watercolor images is also present in several commercial painting systems. Most significant is Corel Painter IX, which incorporates retouchable wet areas and lets users control the diffusion process.[14]

Although closely related in that they both use brush and water, a distinction can be made in the goals persued by Oriental black ink paintings and Western watercolor paintings. The former uses highly absorbent, thinner, and more textured paper types. Other apparent differences can be found in the brush types and paint techniques. An extensive comparison of both techniques is given by several authors.[15,16,17]

Overview and Architecture

The canvas model has a layered design, consisting of three active layers and an unlimited number of passive layers. The passive layers are considered to contain previously drawn strokes that have dried and no longer participate in the simulation, except in the final step when the canvas is rendered. The part of the simulation that handles the dynamics of pigment and water takes place in the active layers. Inspired by the three-layer canvas model of Curtis *et al*.,[6] our active layers have very similar tasks. The underlying computational model, however, is very different (Figure 2). The motivation for using this three-layer design stems from an analysis of the behavior of paint; three different states for pigment or water can be distinguished:

- Pigment and water in a shallow layer on top of the canvas.
- Pigment deposited on the canvas.
- Water absorbed by the canvas.

In what we will be referring to as the shallow fluid layer, a 2D fluid body consisting of a mixture of water and paint pigment represents a stroke on top of the canvas. The water will eventually evaporate or be

Copyright © 2005 John Wiley & Sons, Ltd.

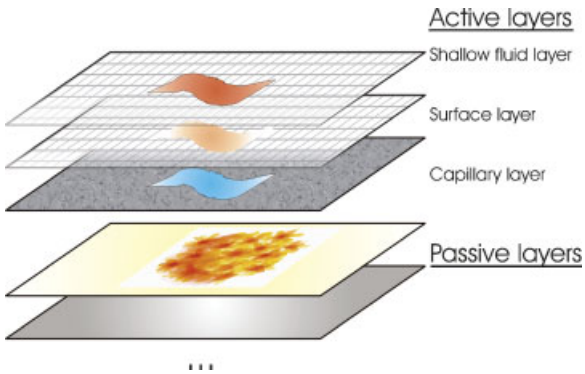


Figure 2. The canvas model, with three active layers and an unlimited number of passive layers.

absorbed into the canvas capillary layer, while pigment particles will settle within the irregularities of the canvas surface, represented by the surface layer. Figure 2 presents a schematic view of the canvas model. In subsequent sections we will elaborate on the details of each layer, starting with the active layers.

Shallow Fluid Layer

In its initial form, a stroke consists of a shallow layer of fluid that resides on top of the canvas surface. The movement of the fluid flow through time is given by the two-dimensional Navier—Stokes equation.[18] In literature we can find quite a few numerical approaches to solve this problem, but we must keep in mind some issues on this matter:

- The procedure of finding a solution must be both fast and stable.
- The fluid flow must be constrained by the stroke's boundaries.
- We wish to add constraints on the level of individual cells. For example, a cell's water content may not violate predefined upper and lower boundaries.

The canvas model from Curtis *et al.*[6] uses the algorithm proposed by Foster and Metaxas,[7] based on a finite differencing of the Navier–Stokes equation and an explicit ODE solver. This approach, however, is not suitable in our situation, as it is both slow and unstable under certain circumstances. The main problem is the explicit time solver in combination with a high diffusion factor, which is what we are dealing with in case of watery paint.

More suitable for our purpose is the work of Stam, describing a number of fast and stable procedures to simulate fluid flows using implicit solvers.[19,20] It allows us to take larger time steps and results in a faster simulation that never 'blows up.' Baxter *et al.* use a similar approach in their viscous paint application.[13] The key difference with watery paint is an extra diffusion step, as well as the necessity to model pigment and water interaction with the canvas.

The state of a vector field $\vec{v}$ defining the velocities of a fluid body at any given time and space during simulation is given by Equation (1), which is a variant of the Navier–Stokes equation.[19]

$$\frac{\partial \vec{v}}{\partial t} = -(\vec{v} \cdot \nabla)\vec{v} + \nu \nabla^2 \vec{v} \tag{1}$$

In order to adapt this procedure for the purpose of moving paint fluid, we need to introduce the following variables for this layer:

- A vector field $\vec{v}$ defining the fluid velocity. In discrete form we assign to the center of each cell i, j velocity $(v_x, v_y)_{i,j}$.
- Water quantity $w_{i,j}$ for each cell, measured in terms of height, and constrained within $[w_{\min}, w_{\max}]$.
- Amount of pigment $(p_{\text{idx}})_{i,j}$ for each cell, as a fraction of the cell surface. Each pigment type is denoted by a unique index.
- A diffusion constant ν, determined by the ratio of mass density ρ and viscosity η of the fluid.

Given these variables, we can mainly follow the method of solution described by Stam to update the state of a fluid flow. As shown in the next sections, some modifications were made that are specific to our problem.

A time step in this part of the simulation requires four operations:

1. Add water, pigment, and velocity values.
2. Update velocity field $\vec{v}$ (Equation (1) and section).
3. Update water quantities w (Equation (2) and section).
4. Update pigment quantities p_{idx} for each pigment (Equation (3) and section).

Once the velocity field is updated according to Equation (1), we can use it to update each cell's water quantity (Equation (2)) and pigment quantity for each pigment (Equation (3)):

$$\frac{\partial w}{\partial t} = -(\vec{v} \cdot \nabla)w + \nu_w \nabla^2 w \tag{2}$$

$$\frac{\partial p_{\text{idx}}}{\partial t} = -(\vec{v} \cdot \nabla)p_{\text{idx}} + \nu_p \nabla^2 p_{\text{idx}} \tag{3}$$

Copyright © 2005 John Wiley & Sons, Ltd.

updateVelocityField (v, source, dt)
addNewVelocities (v, source)
diffuseVelocities (v, dt, diff.rate)
advectVelocities (v, dt)
addHeightDifferences (v, dt)

Table I. Updating the velocity vector field

Updating the Velocity Vector Field

All steps in the update velocity routine are enumerated in Table 1. The 'addHeightDifferences' step accounts for differences in water height and will be explained in the next section.

The state of a two-dimensional fluid flow at a given instant of time can be modeled as a vector field that is sampled at the center of each cell of a 2D grid. Updating the velocity according to Equation (1) now equals resolving the two terms that appear at the right-hand side.[20]

1. self-advection $-(\vec{v}\cdot\nabla)\vec{v}$
2. diffusion $\nu\nabla^2\vec{v}$

Self-Advection

Self-advection calculates how the values in the velocity field affect the velocity field itself. In Reference.[20] an implicit method is described that assigns a particle to each cell center, conveying that cell's velocity value. Intuitively, the particles are dropped in the velocity field and re-evaluated at the resulting position.

Diffusion

Diffusion of the velocity field, caused by the second term at the right-hand side of equation 1, accounts for spreading of velocity values at a certain rate. We use the Jacobi method to solve this problem, which takes the form of a Poisson equation, in order to find $\vec{v}_{\text{new}}$.[21]

updateWaterQuantities (w, source, dt)
addWater (w, source)
diffuseWater (w, dt, diff.rate)
advectWater (w, dt)

Table 2. Updating water quantities

Updating Water Quantities

The previous section handled the computation of the velocity field for this time step. We will now use this new vector field to update the scalar field of water quantities. Again it means first adding additional water quantities w_{source} to the scalar field and then solving the two terms at the right-hand side of the equation:

- diffusion $\nu_w\nabla^2 w$
- advection $-(\vec{v}\cdot\nabla)w$

In practice, the same methods from the previous section could be used at this point. We will develop our own algorithms for diffusing and advecting the water, however, because we want to add constraints on the upper and lower boundaries of a cell's water content, and we want a mechanism to simulate the 'dark edge' effect (Figure 4(k)).

Water Diffusion

For the diffusion process of water quantities we first annotate the velocity field with additional diffusing motion by accounting for differences in water quantities between neighboring cells. This is the previously unmentioned 'addHeightDifferences' step in Table 1.

The water quantity of a cell in the shallow layer is expressed in terms of water height. During the diffusion process, water needs to reach a point at which each cell contains an equal water height. Therefore, we calculate the velocities that are necessary to obtain equal heights in all cells. The resulting vector field $\vec{v}_h$ is combined with the velocity field $\vec{v}_{\text{old}}$ we already calculated: $\vec{v}_{\text{new}} = \omega_i\vec{v}_{\text{old}} + \omega_h\vec{v}_h$, where ω_i and ω_h are weight factors and $\omega_i + \omega_h = 1$. In all examples we used $\omega_h = 0.06$. The advantage of this approach is that the movement of pigment, which depends on the velocity field, will also be affected by differences in water quantities. This way we can obtain the 'dark edge' effect, a result of the fact that at the edge of a stroke water evaporates faster and is replaced by water and pigment from the interior of the stroke. Figure 4(k) shows an example of a computer-generated brush strokes with dark edges.

Water Advection

For each cell, we measure the volume of water that is exchanged with all neighboring cells. As an example,

Copyright © 2005 John Wiley & Sons, Ltd.

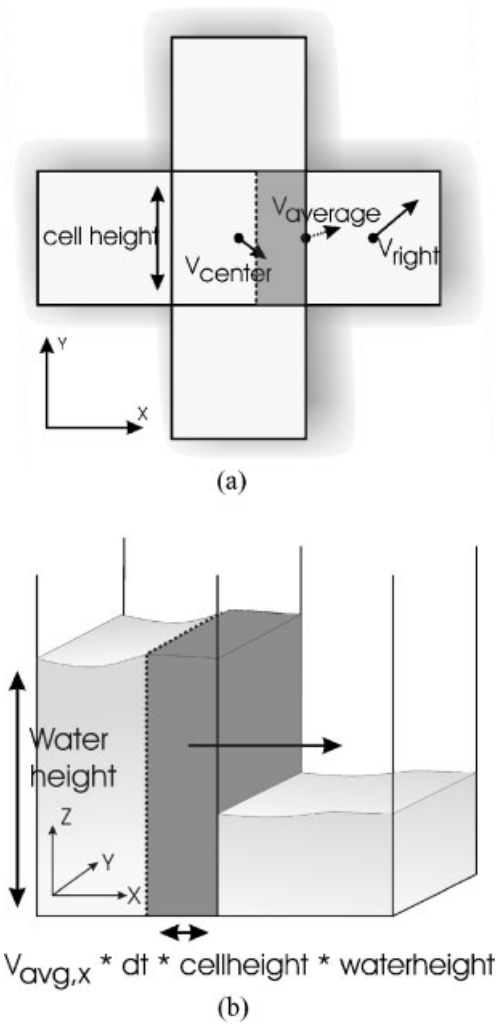


Figure 3. Moving water to a right neighboring cell. The dark areas represent the volume of displaced water.

we will calculate the amount of water that flows from the center cell to its right neighbor, with velocities $\vec{v}_{\text{center}}$ and $\vec{v}_{\text{right}}$ respectively, as shown in Figure 3(a):

$$\vec{v}_{\text{average}} = \frac{\vec{v}_{\text{center}} + \vec{v}_{\text{right}}}{2} \tag{4}$$

Any point along this border has velocity $\vec{v}_{\text{average}}$ and travels within a given time step Δt a distance $\Delta x = (\vec{v}_{\text{average}})_x \Delta t$ in horizontal direction. Therefore, the area covered by the volume of displaced water is given by the colored area in Figure 3(a), and equals Δx (cell height).

Finally, from the amount of water in the cell, the total volume of displaced water $\Delta V = \Delta x$ (cell height)$w_{\text{i,j}}$ can be determined. The change in water quantity is given by Equation (5), and is also shown in Figure 3(b)

$$\Delta w = \frac{\Delta V}{(\text{cell width} \times \text{cell height})} \tag{5}$$

The same procedure is followed to calculate the exchanged water quantities with the three remaining neighbors. We add up the results and divide it by four, because each neighbor contributes exactly one quarter to the total flux. Constraining the cell's upper and lower water quantities can be done by simply clamping all individual exchanged volumes of water. It also ensures conservation of mass.

Evaporation of Water in the Shallow Fluid Layer

This is modeled by removing at each time step a volume of water $\Delta V_{\text{top}} = \epsilon_{\text{shallow}} \Delta t$(cell width $\times$ cell height), according to the cell's water surface and the evaporation rate $\epsilon_{\text{shallow}}$.

Evaporation also occurs at the sides of cells that have neighbors without water. This way we incorporate the fact that water evaporates faster at the edges of a stroke.

Updating Pigment Concentrations

The velocity field not only causes movement of water but also governs the movement of pigment concentrations. In this last stage the changes in pigment concentrations will be calculated

Table 3 gives the three basic steps taken when moving pigment concentrations in the shallow fluid layer according to a given velocity field. It shows that this procedure is similar to moving water quantities, adding a scalar field of new pigment quantities $p_{\text{idx,source}}$ and solving the two terms in Equation (3):

- diffusion $\nu_p \nabla^2 w$
- advection $-(\vec{v} \cdot \nabla)p$

The pigment source term is comprised of amounts of pigment that are added by a brush.

Copyright © 2005 John Wiley & Sons, Ltd.

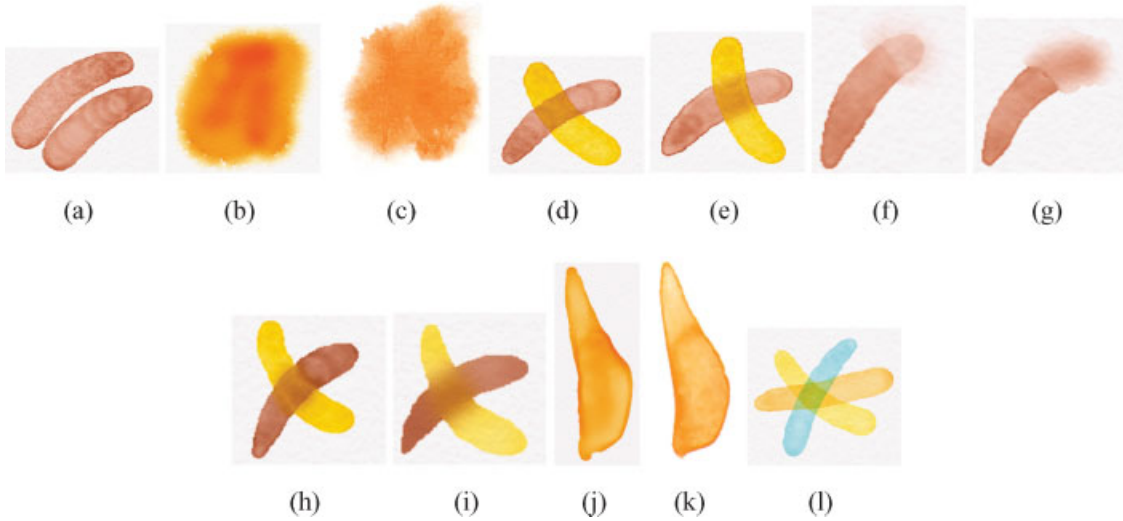


Figure 4. Strokes showing various watercolor effects: high and low pigment granulation causing a difference in accentuation of the canvas texture (a); wet on wet painting showing a feathery pattern in a computer-generated stroke (b) and a real stroke (c); the difference between pigment with high staining power, causing particles to remain stuck on the paper when painted over (d) and pigment with low staining power that is easily picked back up (e); strokes that are washed out with a wet brush, using pigment with high staining power (f) and low staining power (g); the difference between pigment with high density, which falls quickly to the surface (h), and low density, which stays longer in the shallow fluid layer (i); the 'dark-edge' effect in in a computer-generated stroke (j) and a real stroke (k); and 'glazing,' achieved by adding thin layers of watercolor one over another (l).

```
updatePigmentQuantities (p, source, dt)
addPigment (p, source)
diffusePigment (p, dt, diff_rate)
advectPigment (p, dt)
```

Table 3. Updating pigment quantities

The Diffusing Process of Pigment Concentrations

This again uses the Jacobi method for calculating the updated scalar field $p_{\text{idx,new}}$ of each pigment: $(\mathbf{I} - \nu_p \Delta t \nabla^2) p_{\text{idx,new}} = p_{\text{idx,old}}$, with ν_p indicating the pigment diffusion rate.

Advection of Pigment

Advection of pigment or the movement of pigment caused by the velocity vector field relies on the algorithm for moving water. The outgoing fraction of pigment for a similar situation to the one depicted in Figure 3(a) is:

$$\Delta p_{\text{idx}} = \frac{\Delta x(\text{cell height}) p_{i,j}}{(\text{cell width} \times \text{cell height})} \tag{6}$$

Boundary Conditions

One issue we did not consider so far is the fact that the movement of paint must respect the boundaries of a stroke. In our case, a boundary is defined as the interface between the paint and the atmosphere. The boundaries can move, however, as the stroke expands through capillary activity, as we will discuss later.

At any given time step, no pigment or water is allowed to travel across these boundaries. Fortunately, our water and pigment advection algorithms implicitly guarantee this condition. Both algorithms define at each cell the movement of substance to neighboring cells. If we know which cells belong to the stroke, we can simply check if a neighboring cell lies within the stroke and is allowed to receive material. Another consequence of a boundary is that it influences the velocity vector field, making fluid flow along it. This is done by setting the

Copyright © 2005 John Wiley & Sons, Ltd.

normal component of the velocity vector at boundary cells to zero.[7,20]

Surface Layer

Previous sections discussed the activity of water and pigment at the shallow layer. The pigment is initially dropped in the shallow fluid layer, but eventually ends up being deposited on the surface of the paper canvas. In the meantime, there is a continuous transfer of pigment between the shallow fluid layer and the surface layer (Figure 5).

The surface layer keeps track of the deposited amounts of pigment $(p_{\text{idx}})_{i,j}$ for each cell (i, j).

Pigment will be adsorbed and desorbed according to Equations (7) and (8) respectively:

$$\Delta p_{\text{ad}} = \Delta t(p_{\text{water}}(1.0 - w_{\text{frac}})(1.0 - h\gamma_{\text{idx}})\delta_{\text{idx}}) \quad (7)$$

$$\Delta p_{\text{de}} = \Delta t\left(p_{\text{dep}} w_{\text{frac}}(1.0 - (1.0 - h)\gamma_{\text{idx}})\frac{\delta_{\text{idx}}}{\omega_{\text{idx}}}\right) \quad (8)$$

Both equations depend on the amount of water w_{frac} in the shallow layer as a fraction of the maximum water allowed, the paper height fraction h at that cell, and several properties of pigment idx:

- Pigment granulation $0 \leq \gamma_{\text{idx}} \leq 1$.
- Pigment density $0 \leq \delta_{\text{idx}} \leq 1$.
- Pigment staining power $0 \leq \omega_{\text{idx}}$.

The granulation factor determines the influence of the paper height on the amount of pigment transfer. Pigment with high granulation will settle more easily in the cavities of the paper canvas. A high density factor results in pigment that is deposited more quickly. The staining power defines the pigment's resistance of being picked back up by the shallow water layer. Taking these observations into account, we end up with Equations (7) and (8). A similar transfer algorithm was used by Reference [6], but ignored the effect of water quantity.

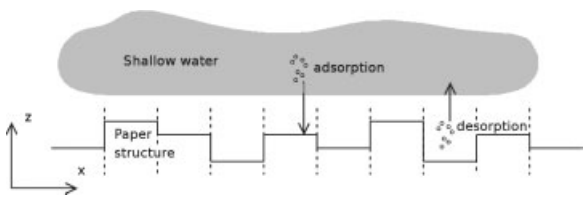


Figure 5. The transfer of pigment concentrations between the shallow fluid layer and the surface layer is influenced by pigment properties.

Capillary Layer

The capillary layer represents the inner paper structure. Within the simulation it is responsible for the movement of absorbed water, allowing a stroke to spread across its original boundaries.

At this point in the simulation, water movement is governed by microscopic capillary effects that can be described in terms of diffusion. The paper structure is represented as a two-dimensional grid of cells or 'tanks' that exchange amounts of water. If each cell has a different capacity, water will spread irregularly through the paper, which is the behavior we want to model. The first thing we have to do is to assign capacities to each cell by generating a paper-like texture.

Fiber Structure and Canvas Texture

The structure of the canvas affects the way fluid is absorbed and diffused in the capillary layer. The canvas texture should also influence the way pigment is transported and deposited. Canvas typically consists of an irregular adsorbent fiber mesh, with the spaces between fibers acting as capillary tubes to transport water. We use the algorithm described by Worley in Reference [22] to produce a textured surface. The same strategy was used by several authors.[6,23] It creates a procedural texture that can serve as a height field (Figure 5).

Capillary Absorption, Diffusion and Evaporation

The water of a brush stroke will gradually be absorbed by the paper canvas. At each time step an amount of water $\Delta w = \alpha \Delta t$ is transferred from the shallow fluid layer to the capillary layer, determined by the rate of absorption α. This amount is clamped according to the amount of water left in the shallow fluid layer, and the available capillary space.

Water in the capillary layer moves to neighboring cells through a diffusion process like we described in context of the shallow fluid layer, and disappears by evaporation at a rate $\epsilon_{\text{capillary}}$. The evaporation process removes each time step an amount of water

Copyright © 2005 John Wiley & Sons, Ltd.

$\Delta w = \epsilon_{\text{capillary}} \Delta t$ from every cell in the capillary layer that has no water left in the shallow fluid layer above.

Graphics Hardware Implementation

All algorithms we described so far were implemented on graphics hardware as fragment shaders using NVIDIA's high-level shading language Cg. The simulation loop relies on the framebuffer-object extension, allowing the results of a rendering pass to be stored in a target texture. Each operation in Tables 1, 2, and 3 is mapped to one or more fragment programs.

Texture objects with floating-point precision were created for each of the following data sets: 2D velocity vectors, water and pigment quantities from both the shallow fluid and the surface layer, water quantities in the capillary layer, and the canvas texture and its reflection coefficients. A total of four textures carry the pigment concentrations in both the shallow fluid layer and the surface layer, so the current implementation limits the number of pigments in each active cell to eight.

The above list excludes several textures that were used to store intermediate results. Several optimizations were made, including the reduction of texture look-ups by caching wet neighbors of a cell, and making sure only relevant parts of the canvas are updated by using an overlay grid that keeps track of areas that were touched by the brush. Just these modified sub-textures are processed in the next time step. Each tile is also annotated with a time-to-live value, based on a fair estimation of the paint's time to dry.

The brush is implemented as a fragment shader that writes pigment, water and velocity values in the appropriate textures according to user input.

Rendering the Canvas

The Kubelka–Munk (KM) diffuse reflectance model is also translated to a fragment program. It iteratively composites every glaze, including the canvas reflection coefficients, to produce the final image. The algorithm assumes that three parameters describing the low-level scattering properties of each layer are known: the layer's thickness d, and the attenuation and scattering coefficients σ_a and σ_s. From these attributes, the KM equations give us R, the fraction of light that is reflected and T, the fraction that is transmitted through the layer.

The input variable d is derived on a per-cell level by measuring the total fraction of pigment in both the shallow fluid layer and the surface layer. The attenuation and scattering coefficients are passed to the shader as uniform parameters. They are part of a user-defined palette, which contains coefficients for each RGB color channel. Finally, repeated application of the KM composition equation takes care of multiple, stacked layers.

Results

All results are created with our application on a Intel(R) Xeon(TM) 2.40 GHz system equipped with a NVIDIA GeForce FX 6800 graphics card. A Wacom tablet interface was used as a brush metaphor. In all cases the canvas measured 800×600 cells, with an overlay grid of 32×32 tiles. The frame rate of 20 frames/second is affected when a user covers a very large area within the same active layer and draws fast enough so that the drying process does not deactivates any tiles in the overlay grid. In this situation, user interaction is still possible at about half the normal frame rate. Users are provided with an intuitive interface, displaying the canvas and a default palette with 12 different pigment types. Basic operations include the possibility to save intermediate results, and canvas operations like drying, clearing, and starting a new layer.

Watercolor

The strokes in Figure 4 show examples of typical watercolor paint effects. Several images created with our system are depicted in Figure 6.

Oriental Black Ink

Although the brushes and techniques used in Oriental paintings are very different from those in Western painting, the mechanics of pigment and water are quite similar. The canvas is generally more textured and more absorbent, and the dense black carbon particles are smaller and able to diffuse into the paper. The former property is easily obtained in our simulation by generating a rougher canvas texture and using a higher absorption constant. Despite the fact that our canvas model does not simulate pigment particles inside the canvas structure, ink diffusion can still be handled by the top layer and produce the typical feathery pattern. The palette consists of very dark pigment with high

Copyright © 2005 John Wiley & Sons, Ltd.

(a)

(b)

Figure 6. Several computer-generated watercolor images.

(a)

(b)

Figure 7. An image in Oriental black ink created with the system (b), based on the orginal 'La Mort' by Marie-Ann Bonneterre (a).

density. Figure 7 depicts a computer-generated painting in black ink, compared with the original 'La Mort'.

Gouache

Gouache is watercolor to which an opaque white pigment has been added. This results in stronger colors than ordinary watercolor. A layer of paint covers all layers below, so paint is not applied in glazes. Also, gouache is not absorbed in the canvas but remains on the surface in a thick layer, creating flat color areas. These properties can be mapped to our model by replacing the KM optical model with a simpler algorithm that blends layers together based on local pigment

Copyright © 2005 John Wiley & Sons, Ltd.

Figure 8. An example of a computer-generated gouache image.

concentrations. Using a higher viscosity factor and loading more pigment in the brush results in thicker paint layers. Figure 8 shows an example of computer-generated gouache.

Conclusion and Future Work

In this paper we presented the results of our research on a physically-based system for creating images with watery paint. The goal was to design a system that runs in real time, and yet able to recreate the various effects specific to this medium. We implemented a prototype on graphics hardware using several fragment programs that operate on simulation data stored in texture objects. Results show that a wide range of effects can be achieved. The interactive process of creating them was positively evaluated by several users. Although most users did not indicate the currently used simple brush model as a major shortcoming, future work includes the design of a better brush model to produce more realistic stroke shapes. Current work also includes the creation of animated paintings as well as the design of several tools that are quite common in real painting but have no digital counterpart yet.

ACKNOWLEDGEMENTS

We gratefully express our gratitude to the European Fund for Regional Development (ERDF), the Flemish Government and the Flemish Interdisciplinary institute for Broadband Technology (IBBT), which are kindly funding part of the research reported in this paper. Part of the work is also funded by the European research project IST-2001-37116 'CUSTODIEV'. We thank Marie-Anne Bonneterre and José Xavier for providing us with a digital version of 'La Mort'. Our gratitude also goes to Koen Beets and Bjorn Geuns for giving helpful suggestions, valuable feedback, and for sharing their knowledge on the domain.

References

1. Van Laerhoven T, Liesenborgs J, Van Reeth F. Real-time watercolor painting on a distributed paper model. In *Proceedings of Computer Graphics Interna-tional 2004*, 2004; pp. 640–643.
2. Tunde Cockshott M. *Wet and Sticky: A novel model for computer-based paint-ing*. PhD thesis, Glasgow University, 1991.
3. Small D. Modeling watercolor by simulating diffusion, pigment, and paper fibers. In *Proceedings of SPIE*, **1460**, 1991.
4. Greene R. The drawing prism: a versatile graphic input device. In *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*. ACM Press: NY, USA, 1985; 103–110.
5. Strassmann S. Hairy brushes. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*. ACM Press: NY, USA, 1986; 225–232.
6. Curtis CJ, Anderson SE, Seims JE, Fleischer KW, Salesin DH. Computer-generated watercolor. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*. ACM Press/Addison-Wesley Publishing Co.: NY, USA, 1997; 421–430.
7. Foster N, Metaxas D. Realistic animation of liquids. In *Graphical Models and Image Processing*, 1996; pp. 471–483.
8. Kubelka P, Munk F. An article on optics of paint layers. In *Z. tech Physik*, Vol. 12, 1931; 593–601.
9. Wolfram S. *A New Kind of Scienc*e (1st edn). Wolfram Media, Inc.: Champaign, IL, 2002.
10. Zhang Q, Sato Y, Takahashi J-Y, Muraoka K, Chiba N. Simple cellular automaton-based simulation of ink behaviour and its application to suibokuga-like 3d rendering of trees. In *Journal of Visualization and Computer Animation* 1999; 27–37.
11. Yu YJ, Lee DH, Lee YB, Cho HG. Interactive rendering technique for realistic oriental painting. In *Journal of WSCG'2003* 2003; **11**: 538–545.
12. Baxter W, Scheib V, Lin MC, Manocha D. Dab: interactive haptic painting with 3d virtual brushes. In *SIGGRAPH 2001, Computer Graphics Proceedings*, Fiume E (ed.). ACM Press, 2001; 461–468.
13. Baxter W, Wendt J, Lin MC. Impasto: a realistic model for paint. In *Proceedings of the 3rd International Symposium on Non-Photorealistic Animation and Rendering*, 2004; pp. 45–46.

Copyright © 2005 John Wiley & Sons, Ltd.

14. Corel. *Corel Painter IX*. World Wide Web, http://www.corel. com/painterix, 2004.
15. 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*, Elsevier Science Inc., 1997; pp. 1571–1576.
16. Lee J. Simulating oriental black-ink painting, Vol 19. IEEE Computer Society Press, 1999; 74–81.
17. Guo Q, Kunii TL. ''nijimi'' rendering algorithm for creating quality black ink paintings. In *Proceeding of Computer Graphics International '03*, 2003; pp. 152–161.
18. Kundu PK, Cohen IM. *Fluid Mechanics* (2nd edn). Academic Press, 2002.
19. Stam J. Stable fluids. In *Siggraph 1999, Computer Graphics Proceedings*, Rockwood A (ed.). Addison Wesley Longman: Los Angeles, 1999; 121–128.
20. Stam J. Real-time fluid dynamics for games. In *Proceedings of the Game Developer*, Mar 2003.
21. Golub GH, Van Loan CF. *Matrix Computations* (2nd edn). Baltimore, MD, USA, 1989.
22. Worley S. A cellular texture basis function. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*. ACM Press: NY, USA, 1996; 291–294.
23. Sousa MC, Buchanan JW. Observational model of graphite pencil materials. *Computer Graphics Forum* 2000; **19**: 27–49.

Authors' biographies:

Tom Van Laerhoven is a research assistant in computer science at the University of Hasselt (UHasselt) in Diepenbeek, Belgium. He obtained a M.S. in computer science in 2000 at UHasselt (formerly LUC) and is currently working as a Ph.D. student at the Expertise centre for Digital Media (EDM), also at UHasselt. His research activities are concerned with computer animation, physically-based modeling and animation, non-photorealistic rendering, and parallel and distributed algorithms.

Frank Van Reeth is Professor of computer science at the University of Hasselt (UHasselt) in Diepenbeek, Belgium. He is deputy managing director of the Expertise centre for Digital Media (EDM) at UHasselt. He obtained a M.S. 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.

Copyright © 2005 John Wiley & Sons, Ltd.