A Viscous Paint Model for Interactive Applications

William Baxter, Yuanxin Liu and Ming C. Lin
Department of Computer Science
University of North Carolina at Chapel Hill
{baxter,liuy,lin}@cs.unc.edu

Abstract: *We present a viscous paint model for use in an interactive painting system based on the well-known Stokes' equations for viscous flow. Our method is, to our knowledge, the first unconditionally stable numerical method that treats viscous fluid with a free surface boundary. We have also developed a real-time implementation of the Kubelka-Munk reflectance model for pigment mixing, compositing and rendering entirely on graphics hardware, using programmable fragment shading capabilities. We have integrated our paint model with a prototype painting system, which demonstrates the model's effectiveness in rendering viscous paint and capturing a thick,* impasto*-like style of painting. Several users have tested our prototype system and were able to start creating original art work in an intuitive manner not possible with the existing techniques in commercial systems.*

1 Introduction

For centuries, artists have used traditional media and tools to express their thoughts and feelings creatively. Recently there has been a growing interest in non-photorealistic rendering and in simulating artists' traditional media and tools.

In painting, each paint medium has its own characteristics. Viscous paint media, such as oils and acrylics, are popular among artists for their versatility and ability to capture a wide range of expressive styles. However, it is a challenge to design an interactive model that correctly captures the physical behavior of viscous paint, because of the complex underlying set of partial differential equations that govern that motion.

With the increasing trend to use simulation techniques to automatically generate physically-based, realistic special effects, the modeling of fluid-like behavior has recently received much attention. Most of this attention, however, has been focused on the animation of very low-viscosity fluids such as water or air. But many fluids that we encounter on a daily basis are of a more viscous nature. Familiar examples include honey, glue, mud, ketchup, and thick paints. A method for simulating such media interactively must be capable of treating both the high viscosity and the complex free-surface boundary conditions with unconditional stability.

In addition to focusing on the plausible physical behavior of viscous paint, we are also aiming at providing an expressive vehicle for the users to *interactively* create original works using computer systems. This set of dual goals introduce strict constraints and new challenges on the design and implementation of a computational model for a viscous paint medium.

Main Contributions: In this paper, we present an interactive method for modeling very viscous paint media based on a novel stable solver for the viscous Stokes equations, particularly tailored for use in painting simulation. The solver can compute 3D flow throughout the fluid medium and allows realistic mixing of material properties (e.g. pigmentation) internally. It uses a Volume-of-Fluid (VOF) / level set technique to track the free surface and is completely stable based on our observation and user's experiences. It further supports an intuitive, physically-based interaction paradigm for emulating traditional painting settings [Baxter et al. 2001].

Our main goal is to create a paint model that simulates and renders viscous paint, such as oil or acrylic paint, for interactive applications. To more accurately recreate the non-linear chromatic behavior of real paint blending, we have also implemented color mixing and compositing based on the Kubelka-Munk (K-M) model. In order to achieve real-time performance, this is implemented in graphics hardware using programmable fragment shading capabilities. This approach allows for both real-time calculation of the K-M reflectances and dynamic lighting of the paint surface which would otherwise be difficult to attain on a desktop PC.

We have augmented a prototype painting system, which demonstrates the capabilities of our viscous paint medium with real-time Kubelka-Munk mixing and compositing, and several users have created paintings using the system. By manipulating the virtual brush naturally the user can build up layers of paint and paintings in a thick, impasto style. The multiple layers are then rendered using K-M optical composition on graphics hardware. Together with the physically-based modeling of the 3D deformable brushes, our viscous paint model allows the artists to express their creativity freely and paint naturally through a more familiar 3D painting interface than the typical 2D mouse and widgets.

Organization: The rest of the paper is organized as follows. We provide a brief survey of related work in Sec. 2. We present an overview of the interactive painting system and the user interface in Sec. 3. We describe our method for modeling viscous fluid in Sec. 4. Section 5 presents our real-time implementation of Kubelka-Munk model using graphics hardware. We discuss the implementation issues and demonstrate the performance of our model in Sec. 6.

2 Previous Work

A number of researchers have investigated paint and fluid simulation, and also paint rendering. We present a brief summary of related work below.

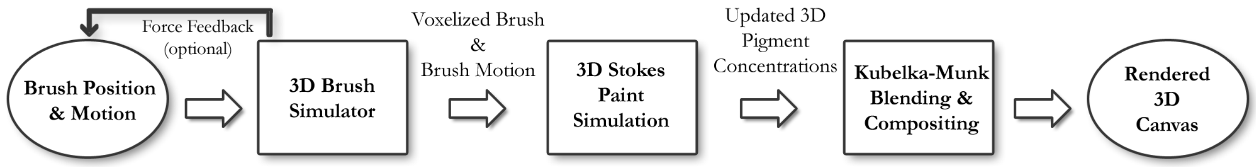
2.1 Fluid and Paint Simulation

[Kass and Miller 1990] used the linearized shallow-water equations to simulate surface waves. The method is fast, stable and interactive, but cannot handle viscous flow, and only simulates the surface height. Internal flow is not computed. [O'Brien and Hodgins 1995] combined a particle system with shallow-water equations to simulate splashing of low viscosity fluid.

[Chen and Lobo 1995] used 2D Navier-Stokes equations, taking the pressure to be proportional to height to get the third dimension. The method is interactive though the physical justification for interpreting pressure as height is questionable. Also, since the method is fundamentally 2D, the internal flow and mixing are unknown.

[Foster and Metaxas 1996] used an explicit marker-and-cell (MAC) method based on [Harlow and Welch 1965] to simulate low viscosity free-surface liquid. Being an explicit method, it was subject to the so-called CFL and viscosity timestep restrictions ($\Delta t < O(\Delta x)$, and $\Delta t < O(\Delta x^2)$, respectively), making it unsuitable for use in interactive applications.

[Stam 1999] introduced the first unconditionally stable solver for the Navier-Stokes equations to the graphics community. The



Force Feedback (optional)
Brush Position & Motion
3D Brush Simulator
Voxelized Brush & Brush Motion
3D Stokes Paint Simulation
Updated 3D Pigment Concentrations
Kubelka-Munk Blending & Compositing
Rendered 3D Canvas

Figure 1: *System Architecture*

solver's use of implicit backwards-Euler integration for viscosity allows for high viscosity fluids, but the method does not address the complications or stability issues introduced by the presence of a free surface boundary condition.

Recently, [Foster and Fedkiw 2001; Enright et al. 2002] presented convincingly accurate particle level set methods for low-viscosity free surface flow, but these methods are quite computationally intensive, requiring minutes per frame for simulation.

[Carlson et al. 2002] presented simulations of melting and flowing of high-viscosity fluids based on the MAC method. While their method treats viscosity implicitly, advection is still performed explicitly, making it subject to the CFL timestep restriction. Also their method for handling the free surface boundary conditions is not clear and likely subject to a timestep restriction. [X. Wei and Kaufman 2003] and [Cockshott et al. 1992] present cellular-automata models for viscous fluid and paint, respectively, which are interactive, but neither is based on the actual physical equations that describe viscous fluid.

[Curtis et al. 1997] used a form of the shallow water equations in their watercolor simulation. Their explicit formulation is subject to timestep restrictions and is inappropriate for very viscous or very thick layers of fluid.

2.2 Paint Rendering

Alvy Ray Smith's original "Paint" program [Smith 1978] perhaps offered one of the first 2D methods for simulating the look of painting. A paint rendering model that offers the look of thick, viscous paint with bump-mapping can be found in [Cockshott et al. 1992].

[Kubelka and Munk 1931; Kubelka 1948; Kubelka 1954] presented the Kubelka-Munk (K-M) equations to accurately approximate the diffuse reflectance of pigmented materials like paint given descriptions of their constituent pigments and their concentrations.

In computer graphics, [Hasse and Meyer 1992] demonstrated the utility of the K-M equations for rendering and color mixing in both interactive and offline applications, including a simple "airbrush" painting tool. [Dorsey and Hanrahan 1996] used K-M layer compositing to accurately model the appearance metallic patinas. [Curtis et al. 1997] also used the K-M equations for optically compositing thin glazes of paint in their watercolor simulation. None of these implementations offers the real-time rendering desired for interactive applications.

3 Overview

In this section we give a brief overview of our painting system and its user interface design.

3.1 System Architecture

In order to test the effectiveness of our viscous paint model, we have created an enhanced interactive painting system based on our previous prototype called dAb [Baxter et al. 2001]. Unlike the earlier prototype system, the new dAb system allows the user to choose between a haptic stylus or a tablet interface, either of which serves as a physical metaphor for the virtual brush. The brush head is modeled with a spring-mass particle system skeleton and a subdivision surface. It deforms as expected upon contact with the virtual canvas. A wide selection of common brush types is made available to the artist.

Our novel viscous paint medium supports important new features, such as impasto-like strokes and the ability to build up many layers of paint, both wet and dry, gouging effects from brush marks, per-pixel lighting effects, rendering of paint bumps, etc. The surfaces of the brush, palette, and canvas are coated with paint using this model. A schematic diagram illustrating how various system components are integrated is shown in Fig. 1.

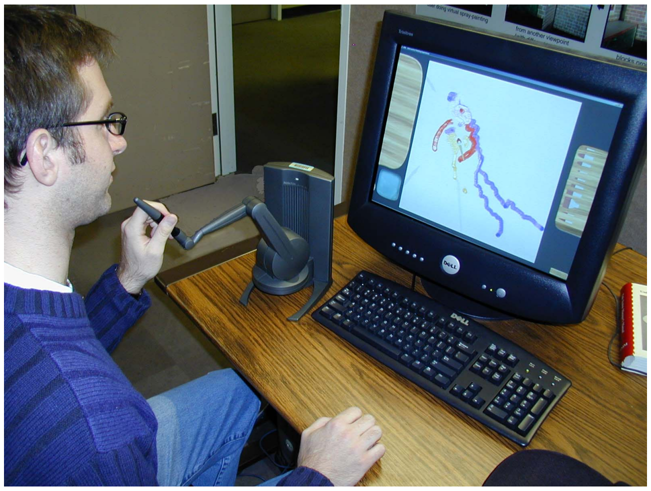


Figure 2: *A Prototype Painting System Using $PHaNTOM^{TM}$ with the Graphical User Interface, the Virtual Canvas and the Brush Rack.*

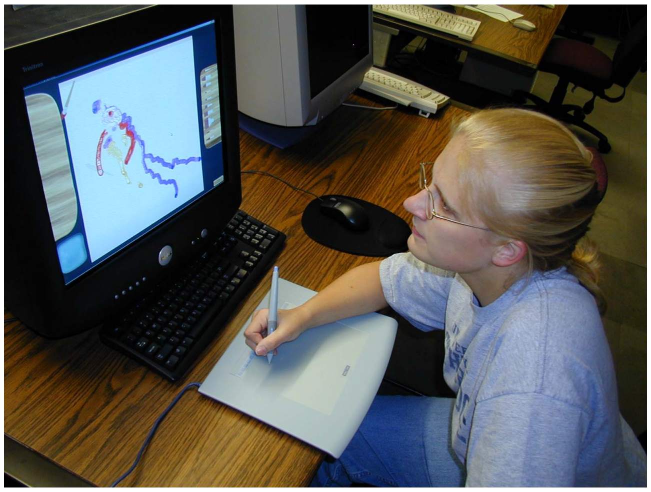


Figure 3: *The Physical System Setup with a Tablet Interface.*

3.2 User Interface

Our prototype paint system runs on a dual-processor PC equipped with NVIDIA's GeForceFX graphics card. One processor is used for optional force display, while the other is dedicated to compute the brush dynamics and paint transfer. The color pigment mixing, composition and rendering takes place on the graphics hardware.

Fig. 2 and Fig. 3 show the physical setup of our system with both the haptic and tablet interfaces.

Unlike the previous dAb system, our new prototype system can infer the user's intention from the current user's hand position. For example, as the user moves the brush handle over the brush rack and hovers over a particular brush type, a new brush is automatically selected without the user explicitly pressing on the stylus button. As the brush moves over to the virtual palette, the palette is automatical brought to the center of the screen to allow color picking and paint mixing. The user no longer needs to explicitly toggle the space bar to bring it up.

We also add the ability for having unlimited numbers of undo's and redo's, in addition to varying degree of quick drying, saving and loading a clean or previously painted canvas. The implementation of undos differs from our previous dAb system, since the current paint model allows multiple wet layers, instead of two layers, of paint. This introduces much richer and more complex optical effects not available with the existing paint systems.

4 Interactive Paint Simulation

In this section, we describe a physically-based paint model based on interactive, stable viscous fluid simulation. Our numerical solver for fluid simulation offers the following characteristics:

- Stable implicit viscosity solver;
- Hybrid linear system solver combines incomplete Cholesky preconditioned conjugate gradient (PCG) with successive over-relaxation (SOR).
- Stable, semi-Lagrangian update of surface and color;
- Stable treatment of the free-stress surface boundary conditions;

We simulate the viscous fluid behavior using the 3D incompressible Stokes equations:

$$\frac{\partial \mathbf{u}}{\partial t} = \nu \nabla^2 \mathbf{u} - \nabla p + \mathbf{F} \tag{1}$$

where u is the velocity of the fluid, ν the kinematic viscosity, and p is the pressure. F represents externally applied forces. We assume constant density, since most familiar viscous materials are homogeneous. Equation 1 is coupled with the equation of continuity,

$$\nabla \cdot \mathbf{u} = 0, \tag{2}$$

which enforces incompressibility and the conservation of mass. The Stokes equation is a simplification of Navier-Stokes applicable for highly viscous flows. The simplification arises from the observation that the contribution of the advection term which appears in Navier-Stokes, $(\mathbf{u} \cdot \nabla)\mathbf{u}$, is negligible for viscous fluids with low Reynolds number flows. This can be understood as the velocity field diffusing so rapidly throughout the fluid that the fluid's inertia does not have time to exert influence on the flow.

4.1 Numerical Method

We use a standard staggered 3D grid as in [Harlow and Welch 1965; Foster and Metaxas 1996; Griebel et al. 1990] and others, with the vector components such as velocity stored on cell edges and scalar quantities (including color channels) stored at cell centers.

The numerical method used to solve the fluid flow equations is an operator splitting method like many previous, e.g. [Foster and Metaxas 1996; Stam 1999; Carlson et al. 2002]. We first compute a provisional velocity field, $\mathbf{u}^*$, that captures the effect of the viscous term, $\nu\nabla^2\mathbf{u}$, and any externally applied body forces, $\mathbf{F}$. This step uses a stable backwards-Euler integration step. We then solve a Poisson problem to find a pressure field, p, that will make $\mathbf{u}^*$ discretely satisfy the compressibility constraint, Eq. 2. Once obtained, the new pressure, p, is used to compute the final divergence-free velocity field, $\mathbf{u}$.

The above three-step temporal discretization scheme can be written succinctly as follows:

$$\begin{aligned} \mathbf{u}^* &= \mathbf{u}^n + \Delta t[\nu\nabla^2\mathbf{u}^* + \mathbf{F}] & (3)\\ \nabla^2 p &= \nabla \cdot \mathbf{u}^*/\Delta t & (4)\\ \mathbf{u}^{n+1} &:= \mathbf{u}^* - \Delta t \nabla p & (5) \end{aligned}$$

where n refers to the time step at which the variables are to be evaluated.

We model forces applied by the user using boundary conditions rather than the forcing term, $\mathbf{F}$, and choose to model a fluid viscous enough that gravity is not a significant influence. Thus, we typically set $\mathbf{F}$ to zero. For less viscous fluid where advection is important, the first step (Eq. 3 can be preceded by a velocity self-advection step as in [Stam 1999].

To model and track the evolution of the free surface – the interface between the fluid and air – we use a Volume-of-Fluid (VOF) method [Hirt and Nichols 1981] in which every cell in the computational domain is assigned a scalar value between 0 and 1 denoting the fraction of the cell which is fluid. For the purpose of placing boundary conditions on the simulation, a cell is treated as fluid if its VOF value is greater than one half. The precise location of the surface is taken to be the $vof = 0.5$ isosurface, though this is used only for rendering. The method for extracting the isosurface is discussed in Sec. 4.6. Unlike previous free surface methods, each step of our numerical method is stable, allowing us to take large time steps and maintain interactivity.

4.2 Viscosity

As can be seen from Eq.3, we solve for the effect of viscosity using an implicit Euler update, which is unconditionally stable [Stam 1999; Carlson et al. 2002]. The spatial discretization of Eq. 3 leads to a system of equations, $K\mathbf{u}^* = \mathbf{u}^n$,where $K = \mathbf{I} - \nu\Delta t\nabla^2_{\mathrm{D}}$ and ∇^2_{D} is the standard 7-point Laplacian stencil in matrix form. The system is actually three independent systems of equations, one for each velocity component, u^*, v^*, and w^*. Expanding the compact matrix notation above out into its constituent linear equations, the system of equations for the u^* component is:

$$\begin{aligned} K_c u^*_{i,j,k} + K_x(u^*_{i-1,j,k} + u^*_{i+1,j,k}) & \\ +K_y(u^*_{i,j-1,k} + u^*_{i,j+1,k}) & \\ +K_z(u^*_{i,j,k-1} + u^*_{i,j,k+1}) &= u^n_{i,j,k} \end{aligned} \tag{6}$$

where

$$\begin{aligned} K_c &= 1 + 2\nu\Delta t(1/\Delta y^2 + 1/\Delta x^2 + 1/\Delta z^2)\\ K_x &= -\nu\Delta t/\Delta x^2\\ K_y &= -\nu\Delta t/\Delta y^2\\ K_z &= -\nu\Delta t/\Delta z^2 \end{aligned}$$

Written as a matrix, K is a $D^3 \times D^3$ matrix, where D is the number of samples on each dimension of the 3D grid, but the matrix is very sparse, containing only $O(D^3)$ non-zero entries, making it amenable to solution with the conjugate gradient method. We use the conjugate gradient method with an incomplete Cholesky preconditioner. Pseudo-code algorithms for the conjugate gradient method as well as the preconditioner can be found in [Golub and Van Loan 1983].

The sparse matrix multiplies required by the conjugate gradient solver can be implemented simply by applying the matrix stencil to the grid, i.e. evaluating the right hand side of (6), at each (i, j, k) on the domain. For example:

```
for k = 0 to DEPTH
  for j = 0 to HEIGHT
     for i = 0 to WIDTH
       if (i,j,k)  in  domain
       then uout(i,j,k) := K_c * uin(i,j,k)
          + K_x * ( uin(i-1,j,k)+uin(i+1,j,k) )
          + K_y * ( uin(i,j-1,k)+uin(i,j+1,k) )
          + K_z * ( uin(i,j,k-1)+uin(i,j,k+1) );
```

would compute the product $K * u_{in}$. The check in line 4 to see if the grid cell is in the domain is used when handling domains with irregular geometry. For the implicit viscosity solver, this check is true if the cell in question is a fluid cell.

4.3 Pressure Solver

Given the tentative velocity field, $\mathbf{u}^*$, we must find a pressure field such that the divergence of $\mathbf{u}^* - \Delta t \nabla p$ is near zero by solving the Poisson problem (Eq. 4). For low viscosity flows, inertial forces dominate (i.e., advection) so there is much temporal coherence in the velocity field. Consequently, a small number of iterations of successive overrelaxation (SOR) per timestep is sufficient to yield realistic-looking results [Foster and Metaxas 1996]. However, in very viscous flow, momentum spreads out quickly, creating large accelerations and low temporal coherence. Thus it is necessary to use more solver iterations to enforce incompressibility as viscosity increases. After experimenting with several different schemes, we have found a particularly effective approach to be a combination of both conjugate gradient (CG) and SOR. Our SOR solver steps are identical to those in [Foster and Metaxas 1996; Griebel et al. 1990]. We use between 10-15 iterations of CG with an incomplete Cholesky preconditioner, followed by 3 or 4 iterations of SOR. The residual after applying CG tends to have a fair amount of high frequency content since CG is a "rougher" [Shewchuk 1994]. A few iterations of SOR applied after CG is particularly effective since SOR acts as a "smoother". In our tests, the CG/SOR combination was quantitatively more effective per CPU second than either technique alone. Comparisons were made by calculating convergence ratios for each technique given the same initial conditions and dividing the result by the computational time required.

4.4 Boundary Conditions

Each stage of the numerical method must be coupled with appropriate boundary conditions. For the diffusion step we use the no-slip Dirichlet velocity boundary condition, $\mathbf{u} = 0$, at wall boundaries, and set the free velocities on the fluid-air interface to discretely satisfy the continuity equation (2). We enforce the boundary conditions by setting the value of "ghost cells", which lie just outside the domain. For Dirichlet boundary conditions the ghost values on an edge along the interface are simply set to zero. Values just off the interface are set so that $(u_{ghost} + u_{neighbor})/2 = 0$, as shown in Fig. 4. For details on implementing the boundary conditions we highly recommend [Griebel et al. 1990].

For the pressure Poisson equation, Neumann boundary conditions are required, $\partial p/\partial n = 0$, where n is the boundary normal. These are implemented by copying the pressure value just inside the domain to the ghost cell just outside, before every CG or SOR iteration. Thus on the face of a boundary cell in the positive x direction we have, for example, $(p_{\mathrm{i}_{nside}} - p\mathrm{g}_{host})/\Delta x = 0$, which is the finite difference approximation to the above boundary condition.

4.5 Interaction

Rather than adding forcing terms to the Stokes equations to implement interaction with the fluid, we can achieve greater control of the fluid by setting Dirichlet velocity boundary conditions at the fluid surface. The velocities of surface cells adjacent to the brush are simply set to the brush's velocity. This is similar to the approach used for interaction with smoke in [Fedkiw et al. 2001].

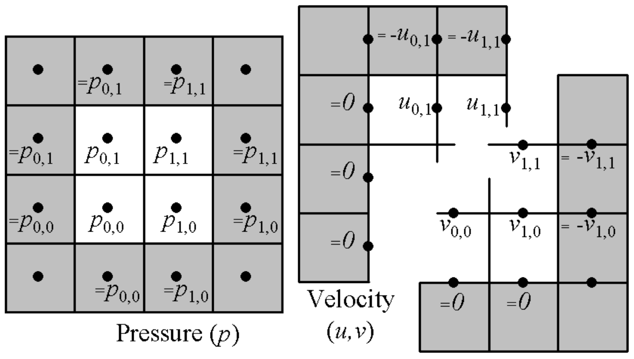


Figure 4: *Setting Pressure And Velocity Boundary Conditions.*

4.6 Free Surface

Unlike previous approaches, our method for handling the free surface of the fluid is stable even at high viscosity. As noted, we represent the surface implicitly as the level set of a fluid fraction function, $f(i, j, k)$, with the interface defined to lie on the $f = 0.5$ isosurface. Insofar as we define the surface using the level set of an implicit function, this approach is similar to that of [Foster and Fedkiw 2001; Enright et al. 2002], but the specific implicit functions used are different.

For rendering, we need to compute an approximation of the isosurface and its normals. The VOF technique represents the true 3D structure of the fluid; however, for use as a paint model a height field representation is acceptable, and it is much less costly to extract. We obtain a height field from the VOF values in a straightforward manner by computing one height value for each column of cells in the grid. We use a simple linear search to find the uppermost fluid cell then interpolate to estimate the isosurface location to sub-cell accuracy. Searches on successive columns can be greatly accelerated by starting the search at the height computed for the previous column.

The surface normals can be computed from the extracted height field surface, but more accurate normals can be obtained by directly computing the gradient of the VOF field. The normal is simply $\mathbf{n} = \nabla f / |\nabla f|$, which can be computed with second order central differences. The two normals computed at the cell centers closest to the fluid surface are interpolated.

To update the surface location, we advect the VOF values using the velocity $\mathbf{u}$ computed from Eqns. 3,4, and 5. The advection is described in Sec. 4.7.

In order to maintain a well defined and continuous surface, it is desirable to perform some additional filtering on the VOF values. There are two competing reasons to filter: first, excessive smearing of the interface introduced by advection leads to an ill-defined surface; and, second, sharp discontinuities in the VOF values lead to inaccurate normals. Essentially we desire the VOF field to always approximate a smoothed step function. To achieve this, we perform curvature-driven smoothing to reduce sharp features, and gradient-driven steepening to force flat regions towards either 0 or 1.

Mean curvature can be computed directly from the VOF values as the divergence of the normals, $\kappa = \nabla \cdot \mathbf{n}$ [Osher and Fedkiw 2002], which can be written:

$$\begin{aligned} \kappa \quad = \quad & (f_x^2 f_{yy} - 2 f_x f_y f_{xy} + f_y^2 f_{xx} \\ & + f_x^2 f_{zz} - 2 f_x f_z f_{xz} + f_z^2 f_{xx} \\ & + f_y^2 f_{zz} - 2 f_y f_z f_{yz} + f_z^2 f_{yy}) / |\nabla f|^3, \end{aligned}$$

where subscripts denote partial derivatives. The standard discretization of this equation using central differencing is second-order accurate.

For smoothing, we use a seven-point blurring kernel which updates each VOF value as a weighted convex combination of itself and its six neighbors:

$$f'_{(i,j,k)} = \frac{f_{(i,j,k)} + c_s \sum_{(l,m,n) \in neighbors} f_{(l,m,n)}}{1 + 6c_s},$$

where c_s is the smoothing amount. We have found a good choice to be $c_s = \text{clamp}((|\kappa| - 80)/100, 0, 1/6)$.

To repair smearing artifacts, we push VOF values toward the extremes of 0 and 1 using a function of the form: $f' := f + (f - 0.5) * c_p$. We have found a good choice for the push factor in conjunction with the above smoothing function to be $c_p = \max((200 - |\nabla f|^2)/2000, 0)$. Note that because of the choice of thresholds, this steepening operation will only operate on the smooth regions of the field, while the smoothing operation above only operates on very steep regions, so that neither undoes the work of the other.

The filtering reduces visual artifacts and serves to recreate some of the effects of surface tension, which is not included in our formal numerical model.

The free surface should also obey the no-stress conditions, which state that no momentum can be transferred across the interface [Hirt and Shannon 1968; Nichols and Hirt 1971; Griebel et al. 1990]:

$$\begin{aligned} p - 2\nu(\partial \mathbf{u}_n/\partial n) &= 0 \qquad (7)\\ \nu(\partial \mathbf{u}_n/\partial m + \partial \mathbf{u}_m/\partial n) &= 0 \qquad (8)\\ \nu(\partial \mathbf{u}_n/\partial b + \partial \mathbf{u}_b/\partial n) &= 0 \qquad (9) \end{aligned}$$

where n,m, and b are the surface normal, tangent and binormal, and $\mathbf{u}_q$ is the directional derivative of $\mathbf{u}$ in the q direction, $\nabla \mathbf{u} \cdot q$. These terms have the effect of slowing down surface waves [Hirt and Shannon 1968; Nichols and Hirt 1971]. This retardation of propagation speed increases with increasing viscosity. At very high viscosity, the free stress forces essentially damp out surface waves instantly. If the free stress conditions are ignored, as in [Foster and Metaxas 1996], fluids of high viscosity will move unrealistically because the surface cells will tend to retain too much momentum. [Hirt and Shannon 1968; Nichols and Hirt 1971] and others incorporated the above free stress terms by solving them for pressure and enforcing that value as a boundary condition on the free surface. However, this approach is unstable for high viscosities.

The source of the instability can be seen by writing out the finite difference approximations for the equation above on surface cell edges. For example, the typical discretization for a surface cell with only one empty cell in the positive x direction is $p_{i,j} = \nu(u_{i,j} - u_{i-1,j})/\Delta x$ [Griebel et al. 1990], where the p value is located at the center of the cell and the u values are on the right and left edges. the As viscosity, ν, becomes large, it is clear that any small fluctuation in velocity values will be magnified into a large positive or negative pressure boundary value. The large pressure in turn leads to a large velocity adjustment in the next time step, resulting in an unstable feedback loop.

Fortunately we have come upon a simple solution. Instead of explicitly incorporating the above free-stress equations, or omitting them entirely, we approximate their effect for very viscous fluid by simply zeroing out the surface velocities at the end of every time step. This is a reasonable approximation for the type of fluid we are interested in, and it does not suffer from instability. With the exception of this important modification, our handling of the free surface boundary conditions is just as in [Foster and Metaxas 1996; Griebel et al. 1990].

4.7 Scalar Advection

After solving for the velocity field $\mathbf{u} = (u, v, w)$, we advance both the VOF values and the color values on the 3D grid using the advection equation for a scalar, s:

$$\frac{\partial s}{\partial t} = -(\mathbf{u} \cdot \nabla)s.$$

We advect using the stable semi-Lagrangian method presented in [Stam 1999]. Specifically, we update the 3D scalar fields by tracing characteristics with an Euler integration step backwards in time:

$$\begin{aligned} \mathbf{x}^* &\equiv \mathbf{x} - \Delta t[\mathbf{u}(\mathbf{x})]\\ f^{n+1}(\mathbf{x}) &= f^n(\mathbf{x}^*). \end{aligned}$$

In general, the source location, $\mathbf{x}^* = (x^*, y^*, z^*)$, will not lie at the center of a cell, so the result is computed using trilinear interpolation of the eight nearest cells. If in backtracking we cross a boundary, the value of the scalar at the boundary is used.

4.8 Summary of Method

Here we present a compact summary of all the steps from beginning to end of one time step.

1. Set boundary velocities to zero (viscous stress approximation)
2. Compute $\mathbf{u}^*$ from the implicit diffusion equation
3. Set pressure boundary values according to Neumann boundary condition.
4. Solve pressure Poisson equation
5. Set surface boundary velocities using continuity equation
6. Advect the VOF values and color/pigments.
7. Extract surface mesh from VOF, and compute surface normals.

5 Paint Rendering

In this section, we describe our realization of the Kubelka-Munk model for paint rendering, which uses modern graphics hardware programmability.

5.1 Kubelka Munk Model

The Kubelka-Munk (K-M) model was developed around 1950 as a simple way to model and predict the diffuse reflectance of pigmented materials, such as paint, based on the constituent pigments and their concentrations in a neutral medium such as oil [Kubelka 1954]. The model computes reflectance, R, and transmittance, T, through a layer of material as a function of pigment concentrations, c, and each pigment's per-wavelength absorption and scattering coefficients, K and S.

The use of K-M equations for computer graphics has been explored by several researchers [Hasse and Meyer 1992; Dorsey and Hanrahan 1996; Curtis et al. 1997]. These references give ample background on the subject, so we will only cover the equations very briefly. Our chief addition to the discussion is a realization of these equations suitable for efficient implementation in hardware shading languages.

Given the K and S values for a pigment at a particular wavelength, the diffuse reflectance at that wavelength of a thick layer which completely hides the substrate is:

$$R_\infty = 1 + \frac{K}{S} - \sqrt{\frac{K}{S} + 2\frac{K}{S}} \qquad (10)$$

For a finite layer of optical thickness, d, the reflectance and transmittance are given by

$$R = \frac{1}{a + b \coth bSd} \tag{11}$$

$$T = \frac{b}{a \sinh bSd + b \cosh bSd}, \tag{12}$$

where $a = 1 + K/S$ and $b = \sqrt{a^2 - 1}$. In the limit as $d \to \infty$ Eqn. 11 reduces to Eqn. 10, and $T \to 0$.

If we have multiple pigments mixed together, each with their own K and S values and concentration by dry weight c_i, we can compute the K and S of the mixture as

$$K_{\text{mix}} = \sum_i c_i K_i$$

$$S_{\text{mix}} = \sum_i c_i S_i.$$

Finally, for two layers L_1 and L_2, with L_2 on top of L_1, we can compute the reflectance and transmittance of the optical composition of the layers given their individual R and T values:

$$R_{12} = R_2 + \frac{T_2^2 R_1}{1 - R_1 R_2} \tag{13}$$

$$T_{12} = \frac{T_1 T_2}{1 - R_{1_\text{b}} R_{2_\text{b}}} \tag{14}$$

Note that in general $R_{12} \neq R_{21}$, i.e. the reflectance of the top of a composite is different from the reflectance of the bottom. Note also the reflectances in the denominator for T_{12} are bottom reflectances. The implication is that when computing the composition of several layers, it is most efficient to compute the overall combination from the bottom up, starting with the canvas reflectance R_0. In this way, one avoids not only the cost of evaluating Eqn. 14 since $T_{01\ldots N}$ is always zero, but also avoids having to compute the reverse reflectances of composites that would be required by Eqn. 14.

5.2 Hardware Implementation

For efficient hardware implementation, first we have started by limiting the number of pigments to either four or eight in our prototype, so that the pigment concentrations can be stored in one or two standard four-component RGBA textures. From these four or eight pigments, any arbitrary mixture can be made. If the initial primary pigments are widely separated in the colorspace, a large gamut of colors can be generated. In fact, the number of pigments and pigment textures is not the computational bottleneck in the hardware shader, so it is quite possible to expand the number of primary pigments somewhat beyond eight, with negligible performance penalty. The bottleneck lies more in the software which must perform advection on every pigment channel.

We use K and S values computed at the RGB color wavelengths typically used in graphics, just as in [Curtis et al. 1997], and have used a similar method to the one presented there to derive appropriate values for K and S.

We have implemented two different approaches to shading with K-M on hardware, a "thicker" style and a "thinner" style. The former makes the assumption that the transmittance of the paint is very low and thus the pigments at the surface completely hide the underlayers, allowing us to use the R_∞ reflectance. For this we need to compute an interpolated value for pigment concentrations at the surface location. We do this on a column-by-column basis as we perform the isosurface extraction.

The second rendering method composites the color of each layer of cells in the voxel grid using Eqn. 13. This requires a separate pass of the shader for each layer. After the Nth pass, the frame buffer holds the composited reflectance of the bottom N layers, and this reflectance is copied to a texture to use as the input base reflectance for pass $N + 1$.

After computing the RGB diffuse reflectance of the pigment mixture at each texel with the K-M equations, we complete the shading with a standard ambient-diffuse-specular light model. We use the surface normals computed in software from the VOF values to render the surface with bump mapping. The canvas and paint reflectance texture is also texture-mapped onto a quad to fill the screen.

In our prototype we have used a voxel grid with 16 layers. To render the composite of all these layers is quite expensive, even with a hardware shading language. To improve the performance we typically window the computation to only perform the K-M shading pass on portions of the canvas which have been modified. Once computed, we can cache and reuse the total reflectance values for most of the canvas each frame.

5.3 Paint Transfer

For depositing paint on the canvas using the brush, we have implemented some simple heuristics similar to those used in [Baxter et al. 2001]. We use the same surface mesh-based brush model as well, but at the beginning of each time step the brush mesh is rasterized onto the voxel grid and brush voxels tagged as wall boundaries which have their velocities set to the brush velocity, as discussed in Sec. 3.4.

For simplicity, in the current system the brush is assumed to have only a single color, but that color is continually blended with the color of the paint it touches to allow the brush to "pick up" paint from the canvas. The amount of color change depends on the number and the color of voxels of paint in contact with the brush.

The brush color is used to update the color of any voxels it comes in contact with, and the VOF values of cells adjacent to the brush are pushed towards 1, so as to have the effect of depositing paint volume on the surface. Additionally, during advection of the VOF and color fields, a color being advected from a cell with near-zero VOF is taken to have the brush color.

6 Results

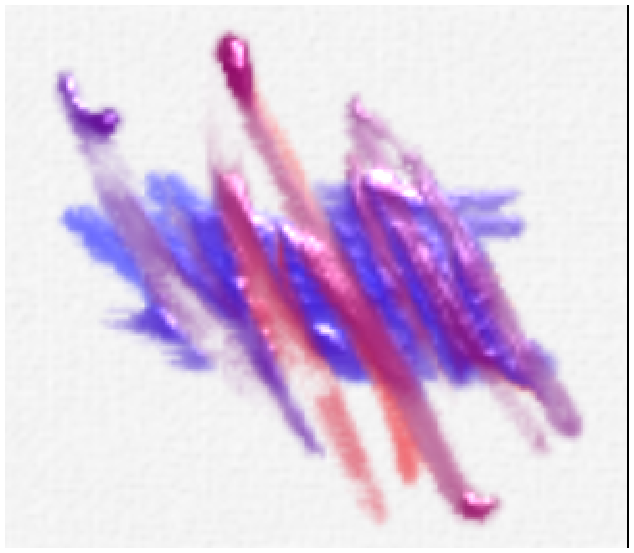


Figure 5: *Thick painting strokes created with our viscous paint model.*

We have implemented our viscous paint model on a 2.5GHz Pentium IV machine. Please see the accompanying video for demonstrations of interaction with the paint model. When used for two-dimensional flow, our viscous free surface simulation runs

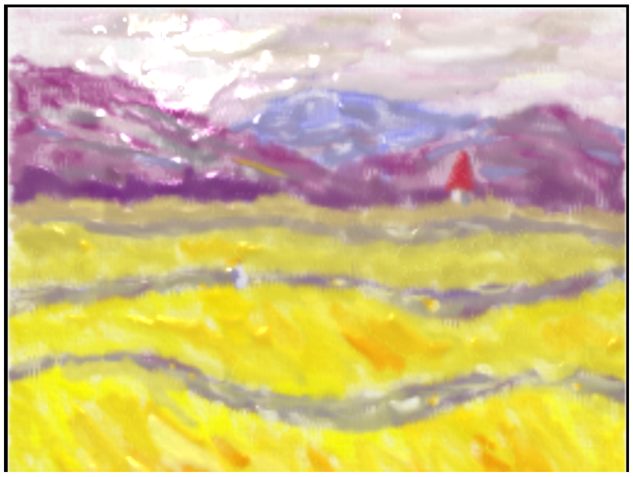


Figure 6: *An image hand-painted using our paint model.*

at 64×64 resolution at over 70 frames per second with rendering of tracer particles. In three dimensions we can compute the flow on a $32 \times 32 \times 16$ grid at 20 frames per second. Since the method is stable, the time step does not need to be reduced even when the fluid undergoes rapid motion. In contrast, a simulation restricted by the CFL or viscosity timestep conditions would not be able to keep the simulation synchronized with wall clock time, since it would have to take many smaller substeps when fluid velocity is large.

We have integrated our paint model with a prototype painting system to simulate an oil-paint-like medium. We provide the user with a large canvas, then window the fluid simulation to calculate flow only in the immediate vicinity of the brush. This optimization is reasonable since a very viscous paint medium essentially only moves in regions in which it is agitated. We render the results by extracting a height field and normal map from the paint fluid as described in Section 4.6. For the rendering we have implemented the Kubelka-Munk reflectance model [Kubelka 1954] using fragment programs on an NVIDIA GeForceFX graphics board. This gives the paint medium more realistic color mixing than is obtained from simple additive RGB blending [Hasse and Meyer 1992; Curtis et al. 1997].

Fig. 5 shows an example of the type of effect produced by our fluid model. Several images created by the users of our prototype painting system are shown in Figs. 6-9. The abstract expressionist painter in particular was fond of the paint model we have introduced. Notice the impasto style of the painting in Fig. 9. The effects were created directly by the painter without any special image processing. He also commented on the ease with which he was able obtain a sense of depth in the paint and how this differered from the commercial applications he has used. Most of the paintings were created by amateur artists within a couple of hours, without much training or elaborate instruction. The footage in the supplementary video demonstrates the interactive performance of our solver and the stable behavior of the viscous fluid generated by our paint model.

7 Summary and Conclusion

In this paper, we presented a novel viscous paint model for interactive applications. The main characteristics of our viscous paint model include:

- An interactive viscous fluid model that captures the dynamic behavior of impasto-like thick paint;
- Real-time color pigment mixing and compositing based on the diffuse reflectance model described by Kubelka and Munk;

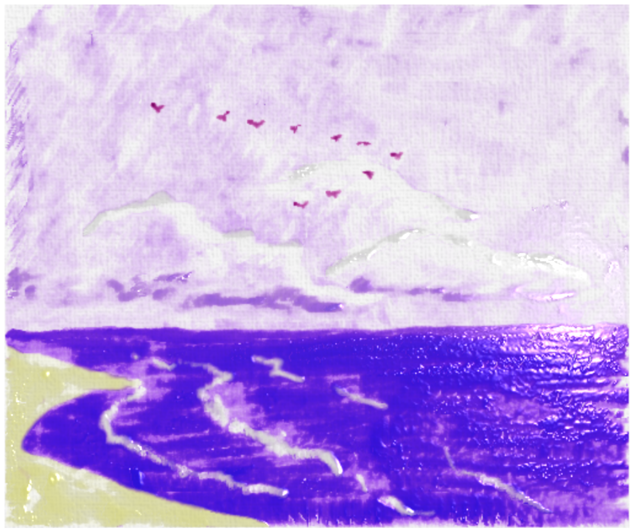


Figure 7: *An image hand-painted using our paint model.*

- Seamless integration with an interactive painting system using an improved user interface and new, volumetric paint representation.

In the future we are interested in investigating fast methods for accurately enforcing conservation of volume, which the current model does not do. Further work is also necessary to more accurately model the fluid-surface interface, especially in the case of a porous surface like canvas. Finally, the current model's relatively coarse resolution makes it unsuitable for very thin layers of material. We believe there is potential to combine our model with 2D methods to more accurately simulate of a wider variety of media.

References

BAXTER, W., SCHEIB, V., LIN, M., AND MANOCHA, D. 2001. Dab: Haptic painting with 3d virtual brushes. *Proc. of ACM SIGGRAPH*, 461–468.

CARLSON, M., MUCHA, P. J., R. BROOKS VAN HORN, I., AND TURK, G. 2002. Melting and flowing. In *Proceedings of the ACM SIGGRAPH symposium on Computer animation*, ACM Press, 167–174.

CHEN, J., AND LOBO, N. 1995. Toward interactive-rate simulation of fluids with moving obstacles using navier-stokes equations. *Graphical Models and Image Processing* (March), 107–116.

COCKSHOTT, T., PATTERSON, J., AND ENGLAND, D. 1992. Modelling the texture of paint. *Computer Graphics Forum (Eurographics'92 Proc.) 11*, 3, C217–C226.

CURTIS, C. J., ANDERSON, S. E., SEIMS, J. E., FLEISCHER, K. W., AND SALESIN, D. H. 1997. Computer-generated watercolor. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., 421–430.

DORSEY, J., AND HANRAHAN, P. 1996. Modeling and rendering of metallic patinas. In *SIGGRAPH 96 Conference Proceedings*, Addison Wesley, H. Rushmeier, Ed., Annual Conference Series, ACM SIGGRAPH, 387–396. held in New Orleans, Louisiana, 04-09 August 1996.

ENRIGHT, D., MARSCHNER, S., AND FEDKIW, R. 2002. Animation and rendering of complex water surfaces. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, ACM Press, 736–744.

FEDKIW, R., STAM, J., AND JENSEN, H. W. 2001. Visual simulation of smoke. In *SIGGRAPH 2001, Computer Graphics Proceedings*, ACM Press / ACM SIGGRAPH, E. Fiume, Ed., 15–22.

FOSTER, N., AND FEDKIW, R. 2001. Practical animations of liquids. In *SIGGRAPH 2001, Computer Graphics Proceedings*, ACM Press / ACM SIGGRAPH, E. Fiume, Ed., 23–30.

FOSTER, N., AND METAXAS, D. 1996. Realistic animation of liquids. *Graphical models and image processing: GMIP 58*, 5, 471–483.

GOLUB, G. H., AND VAN LOAN, C. F. 1983. *Matrix Computations*. The Johns Hopkins University Press.

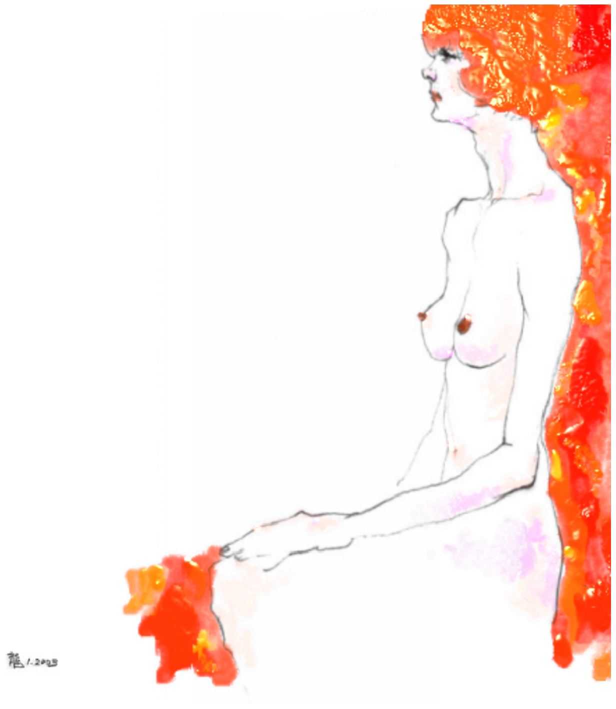


Figure 8: *An image hand-painted using our paint model.*

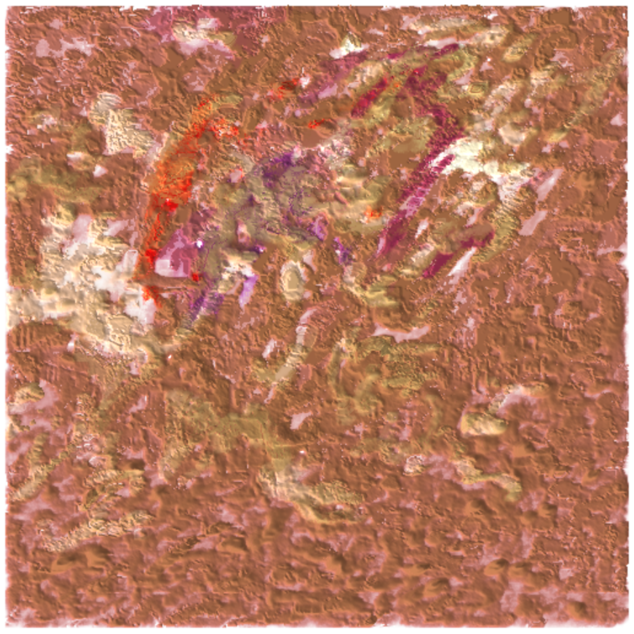


Figure 9: *A painting by abstract expressionist painter, John Holloway*

GRIEBEL, M., DORNSEIFER, T., AND NEUNHOEFFER, T. 1990. *Numerical Simulation in Fluid Dynamics: A Practical Introduction*. SIAM Monographcs on Mathematical Modeling and Computation. SIAM.

HARLOW, F. H., AND WELCH, J. E. 1965. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *The Physics of Fluids 8*, 12 (December), 2182–2189.

HASSE, C. S., AND MEYER, G. W. 1992. Modeling pigmented materials for realistic image synthesis. *ACM Trans. on Graphics 11*, 4, p.305.

HIRT, C. W., AND NICHOLS, B. D. 1981. Volume of fluid (vof) method for the dynamics of free boundaries. *Journal of Computational Physics 39*, 1, 201–225.

HIRT, C. W., AND SHANNON, J. P. 1968. Surface stress conditions for incompressible-flow calculations. *Journal of Computational Physics 2*, 403–411.

KASS, M., AND MILLER, G. 1990. Rapid, stable fluid dynamics for computer graphics. In *Proceedings of the ACM SIGGRAPH symposium on Computer animation*, ACM Press, 49–57.

KUBELKA, P., AND MUNK, F. 1931. Ein beitrag zur optik der farbanstriche. *Z. tech Physik 12*, 593.

KUBELKA, P. 1948. New contributions to the optics of intensely light-scattering material, part i. *J. Optical Society 38*, 448.

KUBELKA, P. 1954. New contributions to the optics of intensely light-scattering material, part ii: Non-homogenous layers. *J. Optical Society 44*, p.330.

NICHOLS, B. D., AND HIRT, C. W. 1971. Improved free surface boundary conditions for numerical incompressible-flow calculations. *Journal of Computational Physics 8*, 434–448.

O'BRIEN, J. F., AND HODGINS, J. K. 1995. Dynamic simulation of splashing fluids. In *Computer Animation '95*, 198–205.

OSHER, S., AND FEDKIW, R. 2002. *Level Set Methods and Dynamic Implicit Surfaces*. Applied Mathematical Sciences. Springer-Verlag.

SHEWCHUK, J. 1994. An introduction to the conjugate gradient method without the agonizing pain. Tech. Rep. CMUCS-TR-94-125, Carnegie Mellon University. (See also http://www.cs.cmu.edu/ quake-papers/painless-conjugate-gradient.ps.).

SMITH, A. R. 1978. Paint. TM 7, NYIT Computer Graphics Lab, July.

STAM, J. 1999. Stable fluids. In *Siggraph 1999, Computer Graphics Proceedings*, Addison Wesley Longman, Los Angeles, A. Rockwood, Ed., 121–128.

X. WEI, W. L., AND KAUFMAN, A. 2003. Interactive melting and flowing of viscous volumes. *Proc. of Computer Animation and Social Agents*.