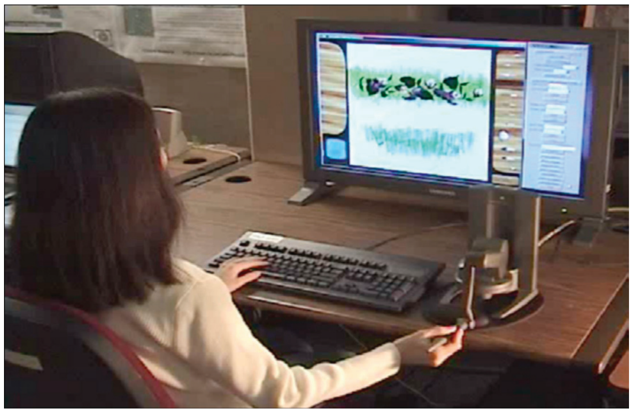
Simulation and Rendering of Viscous Fluid Paint on GPUs

Department of Computer Science University of North Carolina at Chapel Hill November 2004

The Challenge

We have created a digital painting medium, IMPaSTo, which leverages the GPU both to simulate and to render a virtual material similar to oil or acrylic paint. Using our paint model, users can interactively create works in a realistic, thick *impasto* style on the computer.



We model the paint using a simplification of fluid dynamics equations including a conservative, volume-preserving advection scheme, and render paint interactively using a high-accuracy 8-wavelength implementation of the Kubelka-Munk diffuse reflectance model. Using the GPU we are able to perform all of these calculations in real-time as the user manipulates a virtual 3D brush.

Highlights

- **Real-time, viscous fluid simulation based on the Stokes equations with conservative advection**
- **Kubelka-Munk pigment layer compositing and blending**
- **Eight component color space**

Paint Dynamics

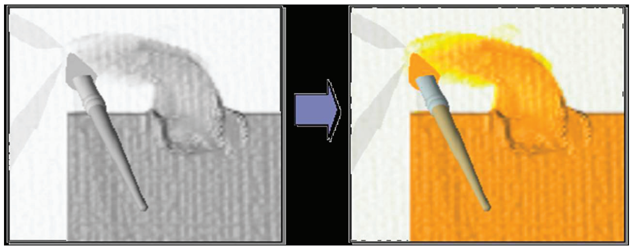
Given the computational resources necessary, the motion of paint can be modeled convincingly using the Navier-Stokes equations which determine a velocity field for the fluid, **v**:

$$\frac{\partial \mathbf{v}}{\partial t} = -(\mathbf{v} \bullet \nabla)\mathbf{v} + \nu \nabla^2 \mathbf{v} - \nabla p + \mathbf{F}, \quad \nabla \bullet \mathbf{v} = 0, \quad (1)$$

subject to appropriate boundary conditions. We use a simplified model for the fluid dynamics, which captures only the dominant terms that contribute to **v**, and use carefully chosen heuristics to capture the rest of the desired behavior. Additionally, for performance we treat a painting as a stack of height fields rather than a general 3D volumetric fluid. We identify two dominant components of **v**. The first is the boundary velocities, which come from the motion of the 3D brush at the brush-paint interface. The second comes from pressure and incompressibility. We simulate paint on the GPU using a 5-stage simulation pipeline.

The first stage in the paint pipeline computes a penetration texture containing the exact region of contact between the 3D brush and the canvas's displaced heightfield surface. In Stage 2 we use a vertex program to compute per-vertex brush velocities by finite differencing, and then a fragment program turns these into a texture containing the 3D brush's velocity at every canvas texel. This stage also computes an approximate pressure contribution to the velocity from the gradient of the penetration depth texture from Stage 1. Next, in Stage 3, a fragment program advects paint volume and paint color using a conservative advection scheme with the velocity texture from Stage 2. Finally, fragment programs in Stage 4 and 5 compute the amount of paint per-texel transferred back and forth between the brush and canvas.

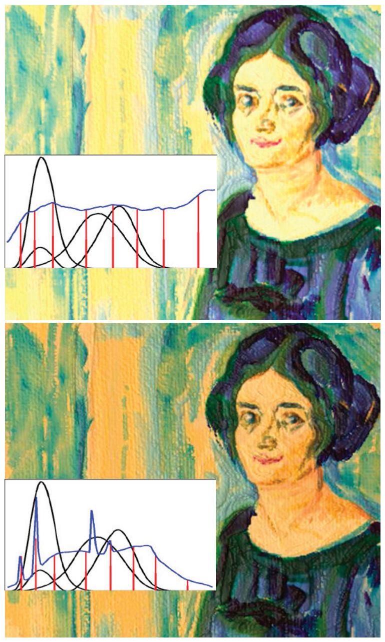
By using the GPU, we are able to compute many per-pixel collision queries, velocity calculations, advection operations and paint transfer operations in parallel every cycle taking advantage of the GPU's SIMD capabilities. Furthermore, many of the operations required in computing the paint transfer between the 3D brush and canvas are naturally implemented by rasterizing the 3D brush surface under a 2D orthographic projection, a job at which the GPU excels. Once the brush is rasterized on the GPU, implementing all the remaining computation also on the GPU allows us to avoid costly readback.



Paint Dynamics: Floating-point textures store the concentration and height field values manipulated by the user. The resulting textures are used as input to the rendering system.

Paint Rendering

For rendering, we use the Kubelka-Munk (K-M) diffuse reflectance model which accurately captures the color characteristics and non-linear blending of pigmented materials like paint. We have realized an implementation of these equations in a form suitable for efficient execution using fragment shaders.



Color Manipulation: We represent a painting's color by storing pigment concentrations at each cell, and we calculate the reflectance of multiple layers of paint using the Kubelka-Munk model. Using Gaussian quadrature, we are able to reduce full-spectrum data to eight sample wavelengths for use in real-time rendering calculations, giving us both high spectral accuracy and efficient utilization of the vector data types available to GPU fragment programs. This also allows us to relight a painting under any full-spectrum illuminant.

We allow the user to choose from a palette of up to eight paints, and implement a paint layer as either two 4-channel textures or one texture with 8 components packed as half-precision floats. We also evaluate the K-M equations using 8 sample wavelengths, rather than the 3 RGB samples some previous authors have used.

To properly evaluate the K-M equations, approximately 450 linear floating point operations (add, multiply) and 60 non-linear operations (division, trigonometric functions) must be performed per pixel. Every updated pixel must perform the same operations. By using GPU-based fragment programs, different pixels may be evaluated by every pipeline in parallel. Also, many of the linear operations can be resolved by simple dot product operations. The GPU-supported 4 channel floating point types, along with the single instruction dot products, create significant improvements when executing these steps. Finally we have found that acceptable results can be obtained with lower than 32 bit precision (the float data type). By using GPU supported 16 bit floating point types (the half data type), we are able to achieve a significant speed-up.

Our rendering pipeline consists of several fragment shaders which convert the initial pigment concentration and heightfield data at each canvas pixel into RGB for display. All of our textures are either half or full precision floating point, as determined necessary.

Results

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 run the simulation and rendering fragment programs as needed with the computational rendering window clipped to the vicinity of the brush. On an engineering sample GeForce FX 6800 board we were able to push over 1.5M texels per second through our entire 5 stage simulation pipeline, and maintain complete interactivity under normal usage.

Project Leader

Ming C. Lin, professor

Graduate Research Assistants

William Baxter, Jeremy Wendt

Research Sponsors

Intel Corporation
National Science Foundation
NVIDIA Fellowship Program
Link Fellowship Foundation
Office of Naval Research
U.S. Army Research Office

Selected Publications

W. Baxter, J. Wendt, and M. Lin, IMPaSTo: A Realistic, Interactive Model for Paint, *Proc. of ACM Non-Photorealistic Animation and Rendering,* 2004, 45-56.

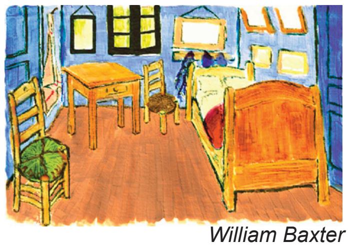
W. Baxter, J. Wendt, and M. Lin, Simulation and Rendering of Viscous Fluid Paint on GPUs, Proceedings of the 2004 ACM Workshop on General Purpose Computing on Graphics Processors, August 7-8, 2004. pp. C-25.

Key Words

Fluid simulation, General purpose computation on GPUs, Non-photorealistic rendering, Painting systems, Simulation of traditional graphical styles.

For More Information

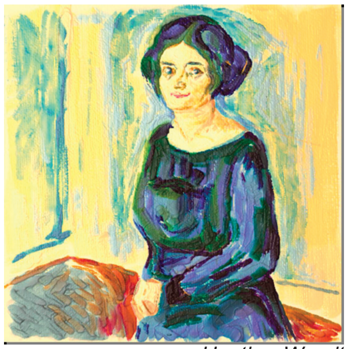
http://gamma.cs.unc.edu/impasto



William Baxter

Eriko Baxter

John Holloway



Heather Wendt

John Holloway