

Dye Advection in a Three-Dimensional Fluid

Ethan Li (ethanli@stanford.edu)

Aug. 11, 2016

Hey, everyone! I'm Ethan, and my project applies fluid dynamics simulation to mimic the phenomenon of dye being carried in milk with soap droplets.



Simulating Milk with Dye and Soap

Michael Zoidis & Jodie Southgate. "Ólafur Arnalds & Nils Frahm - a2 (Official Video)". Erased Tapes Records. Published on YouTube, Nov 2012.

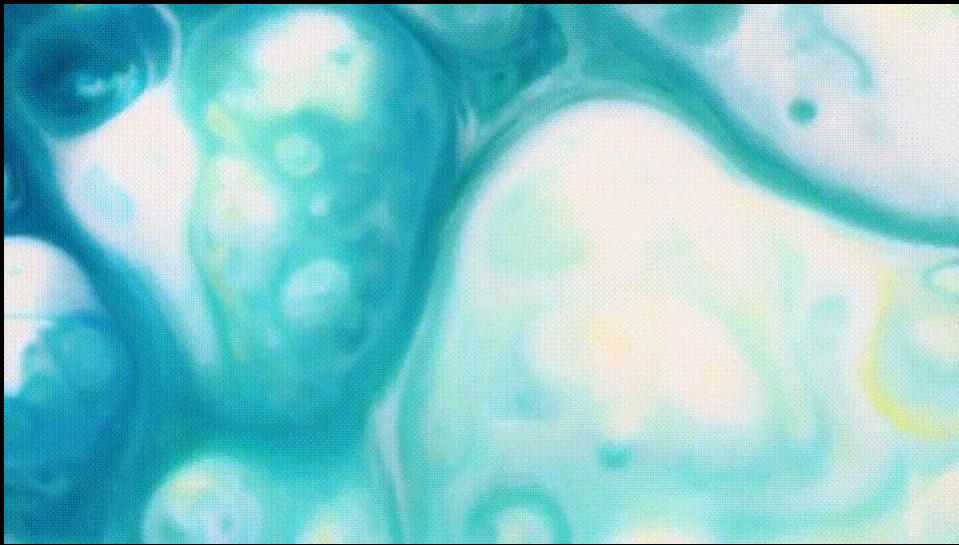
I first saw this in a video by Michael Zoidis and Jodie Southgate, and I was struck by how beautiful it was.



Simulating Milk with Dye and Soap

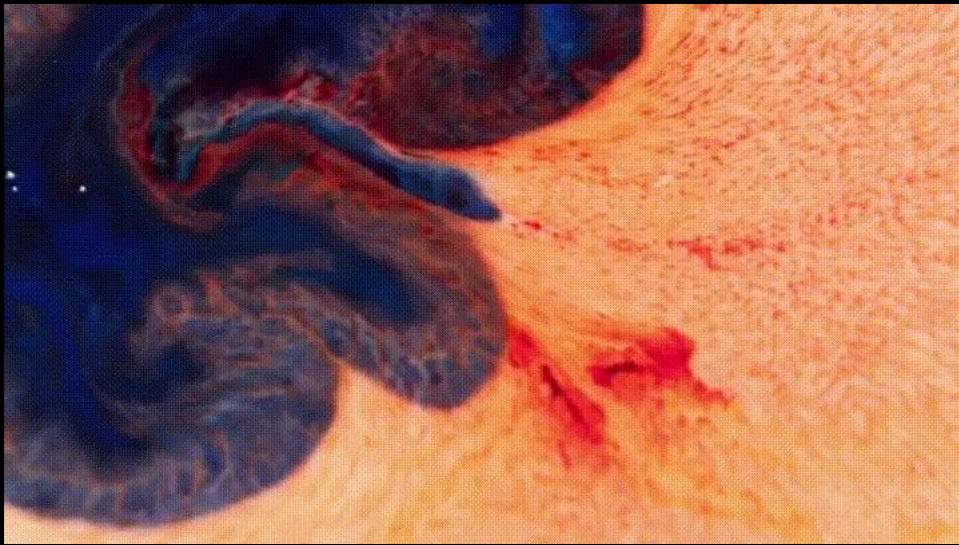
Michael Zoidis & Jodie Southgate. "Ólafur Arnalds & Nils Frahm - a2 (Official Video)".
Erased Tapes Records. Published on YouTube, Nov 2012.

So I wanted to be able to simulate this on a computer, though to a much lower level of detail for reasonable simulation times.



Michael Zoidis & Jodie Southgate. "Ólafur Arnalds & Nils Frahm - a2 (Official Video)".
Erased Tapes Records. Published on YouTube, Nov 2012.

I wanted to simulate fluid coming up from underneath the surface and displacing fluid on the surface - in other words, convection.



Michael Zoidis & Jodie Southgate. "Ólafur Arnalds & Nils Frahm - a2 (Official Video)".
Erased Tapes Records. Published on YouTube, Nov 2012.

And I wanted to see independent flows near the surface - like how the orange layer is sliding past the blue layer here.

Reimplementing Stam's *Stable Fluids*

Jos Stam, "Stable Fluids". In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '99*, pp. 121-128.

Jos Stam. "Real-Time Fluid Dynamics for Games". *Proceedings of the Game Developer Conference*. March 2003.

Mark J. Harris, "Fast Fluid Dynamics Simulation on the GPU". *GPU Gems*, volume 1. Addison-Wesley, 2004.

Dan Morris, "Notes on Stable Fluids (Jos Stam, SIGGRAPH 1999)".

To start out, I did a modern reimplementation of Jos Stam's landmark 1999 method for real-time fluid dynamics simulation.

$$\nabla \cdot \vec{u} = 0$$

$$\frac{\partial \vec{u}}{\partial t} = -(\vec{u} \cdot \nabla) \vec{u} + \nu \nabla^2 \vec{u} + \vec{f}$$

$$\frac{\partial \rho}{\partial t} = -(\vec{u} \cdot \nabla) \rho + \kappa \nabla^2 \rho + s$$

Jos Stam, "Stable Fluids". In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '99*, pp. 121-128.

The goal of Stam's algorithm is to simulate this form of the Navier-Stokes Equations. The first two equations govern fluid velocity, and the third equation governs dye concentration.

Grid Simulation

Update Velocity, Update Dye, Render

Jos Stam, "Stable Fluids". In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pp. 121-128.

Stam's method is a grid-based simulation, as David summarized in lecture 13. My reimplementation was done in C++ using data structures from the Eigen library.

Velocity Update

(Diffuse), Advect, Project

Jos Stam, "Stable Fluids". In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pp. 121-128.

For the velocity update, we add velocities set by the user, advect the field, and project out the divergence.

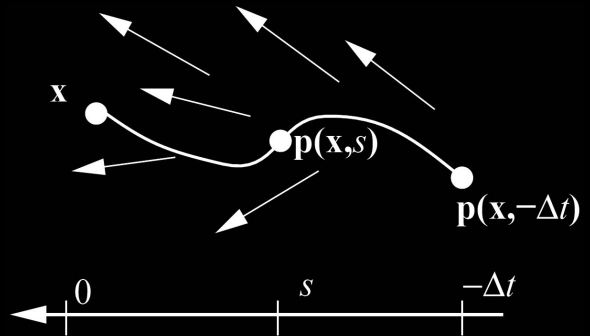
Dye Update

(Diffuse), Advect

Jos Stam, "Stable Fluids". In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pp. 121-128.

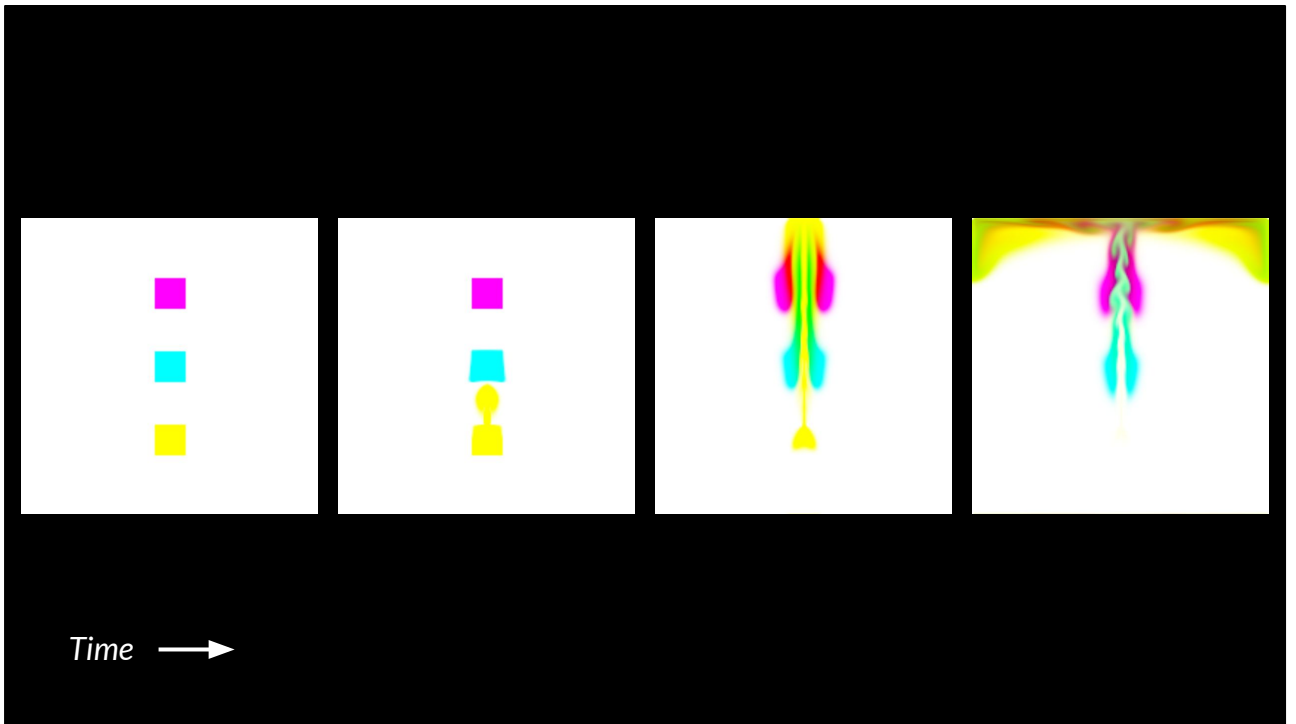
The dye update looks just like the velocity update.

Advection Semi-Lagrangian Backtrace



Jos Stam, "Stable Fluids". In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '99*, pp. 121-128.

The advection step is the backtrace David introduced in Lecture 13.



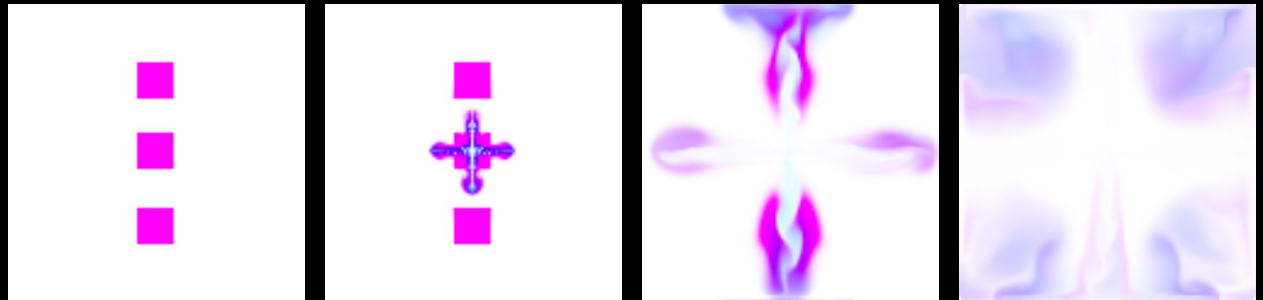
Here are some results I got after implementing Stam's algorithm. I included four frames of the simulation over time, from left to right.

Extending Stam's *Stable Fluids*

For visual improvements, I extended Stam's algorithm.

Three-Dimensional Grid

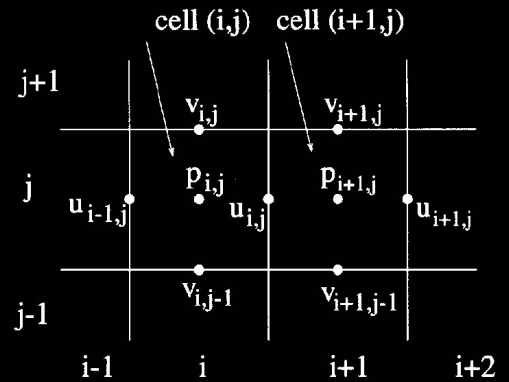
First, to be able to simulate convection, I made my fluid system three-dimensional. This was conceptually straightforward, but implementation took a lot of time to troubleshoot.



Time →

Here, I render the top layer of a shallow 3-d grid. In these initial conditions, I had 3 magenta squares in the top layer and a long cyan rectangle in a middle layer of the system, which isn't rendered. You can see the cyan dye being pulled up into the top layer, demonstrating that convection is happening.

Staggered Grid



Michael Griebel, Thomas Dornseifer, and Tilman Neunhoeffler, "Numerical Simulation in Fluid Dynamics", Society for Industrial and Applied Mathematics, pp. 26-28, 1998.
Colin Braley and Adrian Sandu, "Fluid Simulation for Computer Graphics". 2010.

I then changed the algorithm to use a staggered grid for better accuracy.

RK2 Backtrace

$$x_{mid} = x_G - \frac{1}{2}\Delta t u(x_G)$$

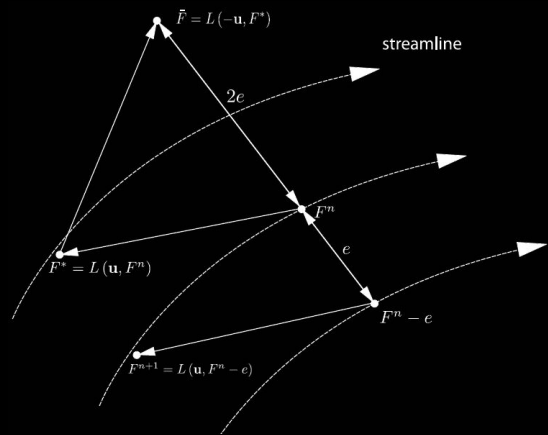
$$x_p = x_G - \Delta t u(x_{mid})$$

Kristofer Schlachter, "Introduction to Fluid Simulation".

Next, I upgraded the backtrace to second-order accuracy by also using a half-way backtrace.

BFCEC

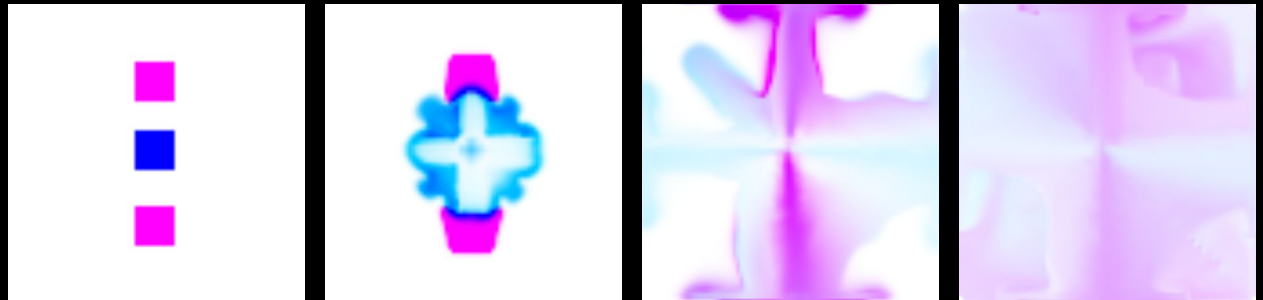
3 Backtraces per Advection



ByungMoon Kim, et al., "Advections with Significantly Reduced Dissipation and Diffusion". *IEEE Trans. Vis. Comput. Graphics*, 13(1):135-144, January 2007.

Santiago D. Costarelli, et al. "GPGPU Implementation of the BFCEC Algorithm for Pure Advection Equations". *Cluster Computing*, 17:243-254, 2014.

Finally, I implemented Kim et al.'s back and forth error compensation and correction to reduce numerical dissipation from discretization error. The idea is that we do 3 different backtraces and combine them in a way that cancels out a lot of error.



Time →

Compared to the previous simulation, we get better preservation of the flows of different dyes and better preservation of finer shape features.

OpenMP Parallelization

Joel Yliluoma, "Guide into OpenMP". 2016.

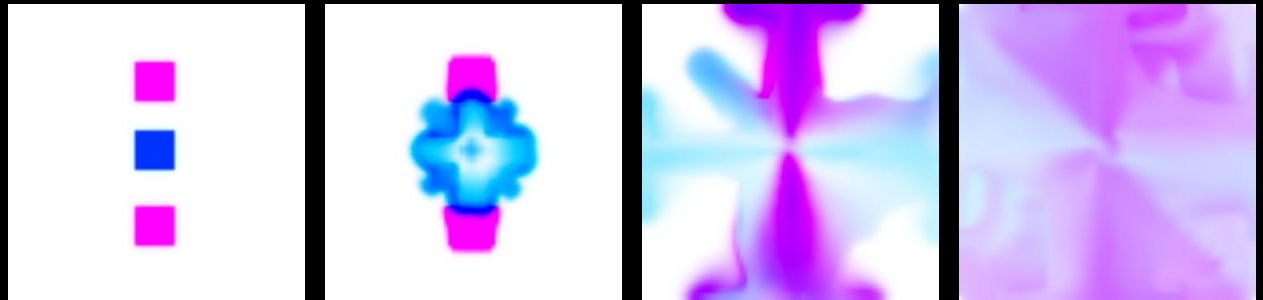
It turns out that, with BFECC, the backtrace is where the program spends most of its time, so I parallelized this step using OpenMP.

Subsurface Scattering

Depth-Dependent Blur & Blend

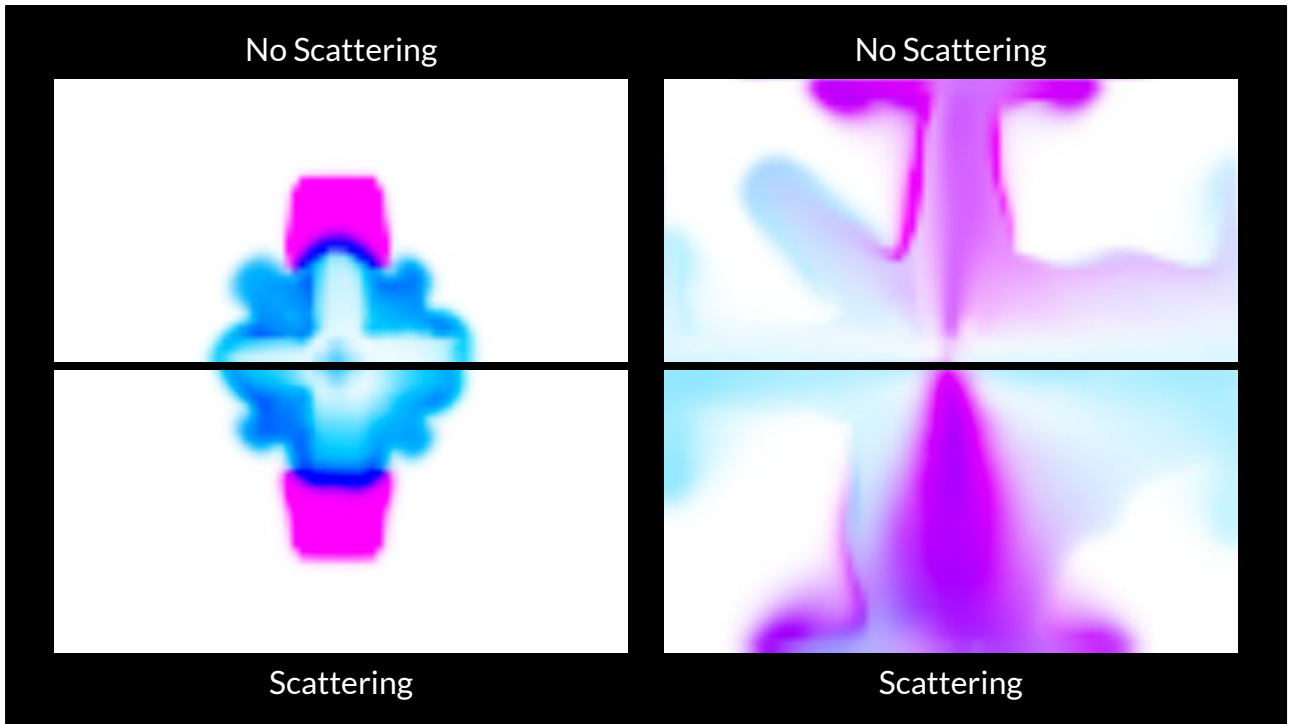
Simon Green, "Real-Time Approximations to Subsurface Scattering".
GPU Gems, volume 1. Addison-Wesley, 2004.

To approximate the effect of subsurface scattering in a flat pan of milk, I sent the top several layers of the grid to the fragment shader as a 3D texture. I performed depth-dependent box blurring and subsequent blending of these layers to shade my rectangle. My inspiration for this scattering approximation came from the GPU Gems chapter on subsurface scattering in 3-d objects



Time →

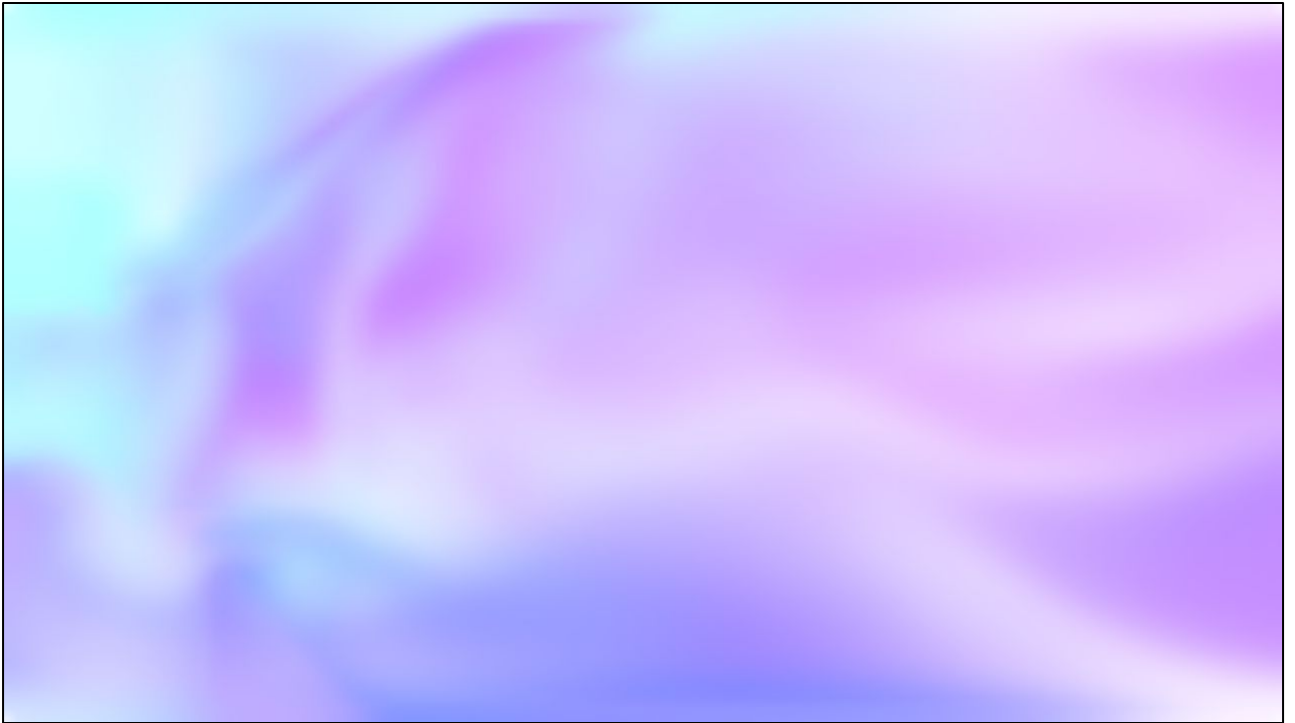
My new shader on the exact same simulation makes the layers look smoother and more realistic, and it's possible to see flows on different layers slide past each other.



Here's a split-image comparison of the two simulations. We can see more flows beneath the surface. The blending also reduces aliasing from the large grid cell sizes of the simulation.

Results

Next, I did some big semi-interactive renders with my software, which supports adding dye and soap during the simulation. I built on the framework from the LearnOpenGL's game tutorial for my user interaction.



First, here's a screenshot from one of my simulations. I achieved this with a 128-by-72-by-8 grid.



For comparison, here's a screenshot from the video I mentioned at the start of my presentation. As you can see, my software gets up to this level of detail with the structure of the flows, which I consider a success for my project given the expense of simulating large grids.



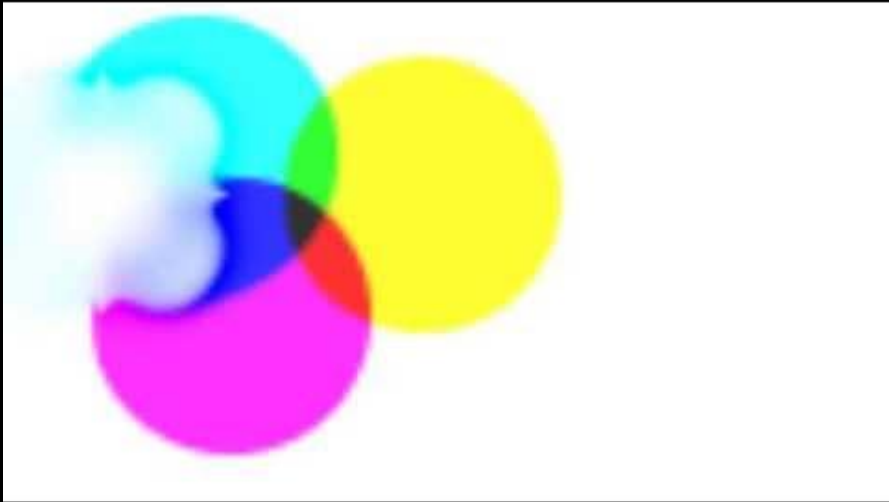
<https://youtu.be/oABozp6HhHs>

Here's a simulation I did on an 80-by-80-by-6 grid. Interactive rendering took about 100 ms/frame on my 2-year-old dual-core chromebook.



<https://youtu.be/AWqSpmojNfo>

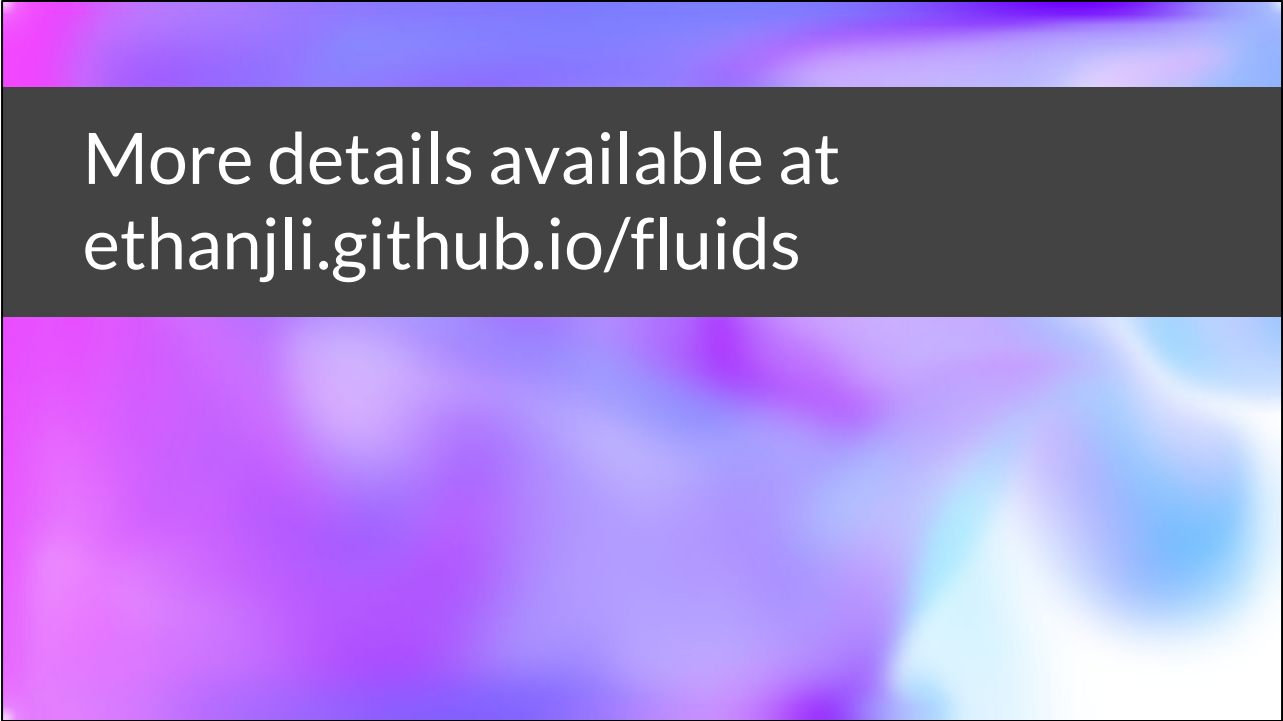
Here's another simulation I did, with the same grid size as the first screenshot. On my 4-year-old quad-core laptop, interactive rendering took about 200 ms/frame



https://youtu.be/PK_edspBgWY



<https://youtu.be/hIC-CZc4cEg>



More details available at
ethanjli.github.io/fluids