User Guide

GPGPU Fluid Dynamics

DEVELOPMENT

Introduction

What Is This?

This code sample demonstrates fast, realistic fluid simulation on the GPU. The simulator solves the Navier-Stokes equations for incompressible fluid flow, as described in Chapter 38 of *GPU Gems*, "Fast Fluid Dynamics Simulation on the GPU," by Mark J. Harris. The article is included with this code sample.

Please note that portions of the algorithms described in this document and the accompanying source code may be subject to patent protection. For more information, see US Patent #6,266,071 B1.

This fluid simulation is an example of a GPGPU (General-Purpose computation on GPUs) application. Fluid simulation involves solving a set of time-dependent partial differential equations (PDEs). To solve them, the quantities governed by the equations (in this case velocity and pressure) are represented on a discrete rectangular grid. At each time step, the equations are solved using an iterative technique based on the *Stable Fluids* technique originally presented by Jos Stam [Stam, 1999]. Due to the parallel nature of GPUs, they are extremely well suited to this type of computation. This sample achieves over an order of magnitude speedup over an equivalent simulation running on a fast CPU.

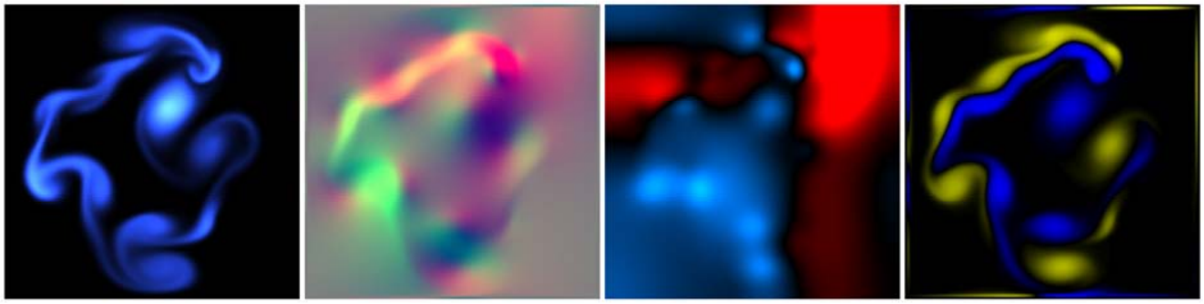

Figure 1: A GPU fluid simulation. These images show the state fields of the simulation. From left to right they are "ink", velocity, pressure, and vorticity.

Using this Sample

To use the fluid simulator, just use the mouse buttons to "paint" into the fluid. When you press the left mouse button and move the mouse you will squirt "ink" into the fluid. You can also press 'j' to inject fluid from four jets on the walls (You can turn the jets on and off individually with number keys 1-4.)

If you open the console, ('~' key), you can adjust sliders for various parameters of the simulation, including viscosity (how goopy the fluid is), vorticity confinement (how turbulent the fluid is), ink color, time step and grid scale.

The most expensive step in the simulation is the solution of the Poisson-pressure equation. (See page 649 of the accompanying *GPU Gems* article.) The reason is that this step requires many iterations of a Jacobi linear equation solver. The default setting is 50 iterations, but you can use more or fewer. Also, you can disable clearing the pressure field ('C' key) to enable the solution to converge faster. The problem here is that this allows errors to propagate between time steps, resulting in "wavier" behavior.

The simulator allows you to draw obstacles in the flow and flow fluid around them. To enable obstacles, press 'o'. Then hold down the control key while you draw with the left mouse button to draw obstacles (hold the alt key to erase them). When you have drawn your obstacles, release control and then inject some fluid. The fluid flows realistically around the obstacles. Try drawing a wall and then erasing a hole through it. When you inject fluid towards the hole, you will see a "smoke ring" come out the other side.

To simulate viscous fluid, increase viscosity on the slider pane. Now your fluid will behave more like "goop". Try pushing the goop through holes in obstacles.

The simulator also provides some vector field visualization modes. These overlay particle or vector arrays over the fluid display to show a vector-field representation of the selected field. There are three modes: Quiver Plot, which draws small oriented and colored vectors at regular spacing over the grid; Points, which draws small colored and / or scaled points at regular spacing over the grid; and Spring-Points, which draws points as above, but displaced from their position according to the intensity and direction (in the case of velocity) of the vector field at that point. See the Table 2 for settings.

The included GPU Gems article discusses further extensions to the simulator that you can try on your own.

Table 1. Mouse Controls

Button	Description

Button	Description
Left Button	Inject ink and velocity into fluid. (“Paint”)
Right Button	Show Menu.
CTRL + Left Button	When enabled, paint obstacles into flow field.
ALT + Left Button	When enabled, erase obstacles.

Table 2. Keyboard Commands

Key	Description
j	Fire ink jets.
1,2,3,4	Toggle ink jets 1-4
o	Toggle arbitrary interior obstacles (hold CTRL to paint them).
v	Toggle vorticity confinement.
p	Pause, resume simulation.
.	Take a single simulation time step (when paused).
r	Reset fluid simulation state.
R	Clear obstacles and reset fluid simulation state.
`,~	Toggle HUD (parameter sliders).
t	Cycle display mode: ink, pressure, vorticity, velocity.
!	Display FPS.
?	Print help text to console.
b	Toggle bilinear interpolation of float textures for display.
m	Toggle display 8-bit texture (fastest) / float texture (high quality)
W	Toggle display on white / black background.
Esc, q	Exit
Keys for Vector Field Visualization	

Key	Description
f	Cycle through vector field modes: [off (default), quiver, color points, scaled points, spring points, scaled spring points]
F	Cycle through vector field grid spacings (increases number of points / vectors until the point resolution is the same as the simulation resolution, and then cycles back to the default of one point per 4 simulation grid cells.)
G	Cycle through vector field visualization input fields [velocity (default), pressure, vorticity, "ink"]
0 (zero)	Disable display of simulation textures (use this after enabling vector field modes with 'f' to see only the vector field visualization.)
Keys to Support Performance Measurement	
C	Toggle pressure field reset each frame[1]
D	Toggle fluid display updates (disable when measuring simulation performance).
L	Run 100 time steps of the simulation and print average step time.
U	Toggle texture updates (glCopyTexSubImage2D).

Known Bugs

There are no known bugs. Please report bugs to Mark Harris (mharris@nvidia.com).

References

Stam, J. 1999. "Stable Fluids." In *Proceedings of SIGGRAPH 1999*.

Harris, M.J. 2004. "Fast Fluid Dynamics Simulation on the GPU". In *GPU Gems: Programming Tips, Tricks, and Techniques for Real-Time Graphics*. Randy Fernando, ed. Addison Wesley.

[1] This allows the number of Jacobi solver iterations to be reduced, and thus enables much higher simulation speed. However, it also allows errors to propagate across time steps. The net affect is that the fluid has pressure waves that make it act more "gelatinous".

Notice

Trademarks

Copyright

NVIDIA Corporation
2701 San Tomas Expressway
Santa Clara, CA 95050
www.nvidia.com