Fluid Simulation

Aviral Gupta (170101014), Devansh Gupta (170101022)

We implemented 2D fluid simulation as described in Jos Stam's papers "Stable Fluids" and "Real-Time Fluid Dynamics for Games." Our program allows real-time interaction and multiple visualization options.

Implementation

Our fluid simulation follows the structure described in Jos Stam's papers, which models fluid flow with the Navier-Stokes Equations, but focuses on maintaining stability and visual believability rather than absolute accuracy.

Velocity Solver

The velocity field solver is broken into four steps: 1) add forces, 2) diffuse, 3) advection, 4) project. We implement the field as two grids--one for the current field and one for the next step (these are swapped after each sub-step).

Add Forces

This step simply updates the velocity grid using a force grid. It follows a very simple equation: $\overline{V}' = \overline{V} + \overline{F} \bullet \Delta T$. The mass of the grid cell is implicitly assumed to be one (so force is equal to acceleration). The force grid is zeroed out after adding if the user does not continue to apply force (i.e. releases the mouse button).

Diffuse

In the diffusion step, we compute the diffusion “in reverse” by solving what velocities at the new point in time would diffuse backwards to the previous velocities. It is stable because for any time step, the new velocity must be less than or equal to some velocity in a previous time step. In order to do this, we must solve a sparse linear system. We use a Gauss-Seidel relaxation algorithm as recommended in Jos Stam’s fluid dynamics paper. A simpler computation of the exchange of particles at each grid cells with its neighboring grid cells is unstable in practice (presumably because it is less accurate than solving a system of equations).

Advection

This step models the self-advection of the velocity field. This simply means that the velocity field “is moved along itself.” As explained in Jos Stam’s paper, we consider each cell center as a particle and back trace the particle from the next state grid to the current state grid. We then bilinearly interpolate this the particle’s position in the current state grid with the four nearest grid cell centers to get the velocity of that particle and thus the new velocity in the next state grid cell. The back-tracing step is done using a fourth order Runge Kutta method, as explained in Kenneth I. Joy’s “Numerical Methods for Particle Tracing in Vector Fields.”

Project

The add forces, diffuse, and advection steps tend to produce non-conservative vector fields. The project step remedies this by using the Hodge decomposition. Any vector field can be expressed as the sum of a divergence-free vector field and the gradient of a scalar. We can then take the Hodge decomposition equation to get an equation that only involves our original vector field and the scalar field:

$$\overline{V} = \overline{U} + \Delta q \quad \Delta \bullet \overline{U} = 0 \qquad \Delta \bullet \overline{V} = \Delta^2 q$$

This final equation is a is a Poisson equation, it can be solved by converting the Laplacian operator Δ^2 into its discrete matrix form and treating the whole equation as a sparse linear system of the form $A\overline{x} = \overline{b}$ where $A$is the matrix form of the Laplacian, $\overline{x}$ is q(the scalar field we need to compute to make our velocity field divergence-free), and $\overline{b}$ is $\Delta \bullet \overline{V}$ (which we can compute from the current velocity field). Thus, project computes the divergence of the velocity field, uses Gauss-Seidel relaxation to compute q and then subtracts Δq from the velocity field to get a new, divergence-free velocity field.

Density Solver

Similar to our velocity field solver, we implement the density field with two grids representing the current and next time step. The density step is comprised of three subsets: 1) add density 2) diffuse 3) advection.

Add Density

This step is very simple; it takes the density grid and sums it together with a density source grid. The density source grid is zeroed out after adding if the user does not continuously supply more density via the UI.

Diffuse

This diffusion step is identical to the velocity diffusion step except it is for a three-dimensional color vector instead of a two-dimensional velocity vector. The exact same principles and math apply.

Advection

This advection step is identical to the velocity advection step except it is for a three-dimensional color vector instead of a two dimensional velocity vector. The exact same principles and math apply.

Boundary Conditions

We assume that the fluid is contained in a box with solid walls: No flow should exit the walls. This simply means that the horizontal component of the velocity should be zero on the vertical walls, while the vertical component of the velocity should be zero on the horizontal walls. For the density fields we simply assume the continuity.

Features and Interaction

The program starts out as a blank black screen. The user can add fluid by right-clicking with the mouse and add forces by left-clicking. The full list of features and visualization modes is shown below:

Adding density (fluid) - right-button mouse click (and drag)
Adding force - left-button mouse click and drag in desired direction of force. The force affects the initially clicked cell, and the strength of the force is proportional to the length of the drag.
Reset - "q" key to restart the simulation.
Brush size - "[" and "]" keys to decrease/increase brush size.
Changing fluid color - "w" for white (default), "r" for red, "g" for green, "b" for blue, "y" for yellow, "m" for magenta, "c" for cyan.
Diffusion rate - "-" and "+" to decrease/increase rate of diffusion.
Visualization modes - "1" key to display fluid density only; "2" key to display velocity field only; "3" key to display both fluid density and velocity field.
Show forces - "f" to toggle visualization of adding forces.

References

Stam, J. (1999). Stable Fluids. *SIGGRAPH 99 Conference Proceedings, Annual Conference Series*, 121-128.

Stam, J. (2003). Real-Time Fluid Dynamics for Games. *Proceedings of the Game Developer Conference*.