Project 2: Fluid System

The Animation of Natural Phenomena

Adrien Treuille - Carnegie Mellon University

Due 10/27 23:59:59

Overview:

In this project you will implement a fluid system with the Stable Fluids algorithm plus vorticity confinement. You must also record a video artifact of your system in action. The class will vote on the best artifacts, and the top three winners will receive extra credit. Additionally, you may implement some of the listed extensions (or invent your own!) for extra credit.

Reading:

- STAM, J . 1999. Stable Fluids. In *Computer Graphics* (SIGGRAPH 1999), ACM, 121–128.
- FEDKIW, R., STAM, J., AND JENSEN, H. 2001. Visual Simulation of Smoke. In *Computer Graphics* (SIGGRAPH 2001), ACM, 15–22.

Skeleton Code:

You have been provided with some skeleton code which you may use to jump-start your project. The code has density fields, velocity fields, and implements basic mouse interaction.

Required Features:

- **Implement Density Advection.** If you're using the skeleton code, then uncomment `Advection` from within `ScalarField::TimeStep`, and implement this function. Use the Semi-Lagrangian advection algorithm described in the Stable Fluids paper.
- **Implement Velocity Advection.** In the skeleton code, the function is `Advection` in `VectorField::TimeStep`, as described in Stable Fluids.
- **Implement Pressure.** The function is `Projection` in `VectorField::TimeStep`. The implementation is described in Stable Fluids.
- **Implement Vorticity Confinement.** The function is `VorticityConfinement` in `VectorField::TimeStep`. The implementation is described in Visual Simulation of Smoke.
- **Bounaries.** The edge of your grid should form a rigid boundary. This implies path clipping at the boundary during the advection, and correctly defining the pressure solver at the boundary. Also, be sure not to perform linear interpolation across the boundary!

Optional Features:

You must explicitly indicated which of these you have implemented. You demo must be able to turn each of these features on and off individually so they can be verified.

- ★ **Multiple densities.** Two different kinds of “smoke” (density) with two different colors.
- ★ **Temperature.** Causes hot air to rise. As described in this paper.
- ★ **Monotonic Cubic Interpolation.** Allows for subvoxel accuracy, as described in this paper.
- ★ **Fixed Objects.** Any edges between voxels could be marked as boundary edges, allowing for arbitrary boundaries.
- ★ ★ **Triangle mesh.** Instead of regular grid, simulate on a triangle mesh. (This allows cool boundary surfaces.)
- ★ ★ ★ **3D.** Implement the algorithm in 3D. Density rendering should work properly from any angle.
- ★ ★ ★ **A high resolution hyperbolic solver.** Higher order accuracy. Described in the appendix to this paper.
- ★ ★ ★ **Particles in Fluids.** Put your particle system in your fluid system, all the fluid field to exert forces on the particles.
- ★ ★ ★ ★ **Moving Objects.** Allows the user to drag solid objects within the flow. The objects come to rest at exact grid offsets. The movement of the object must exert a force on the fluid.
- ★ ★ ★ ★ **Tetrahedral mesh.** The 3D version of a triangle mesh for simulation. Described here.
- ★ ★ ★ ★ **Water.** Implement a water solver with zero air pressure. Must use a signed distance function for level set, but no need to use the full particle level set method.
- ★ ★ ★ ★ ★ **Two-way Coupling.** An extension to moving objects. Not only does the object affect the flow, but the flow exerts forces on the object.
- ★ ★ ★ ★ ★ ★ **Vortex Particles.** Increase turbulence with vortex particles.
- ★ ★ ★ ★ ★ ★ **Cloth.** Put a full cloth simulation on the object, must compute forces not only at particle locations but along the entire cloth surface.
- ★ ★ ★ ★ ★ ★ ★ **Particle Level Set Method.** For higher accuracy, simulate water with the full particle level set method.