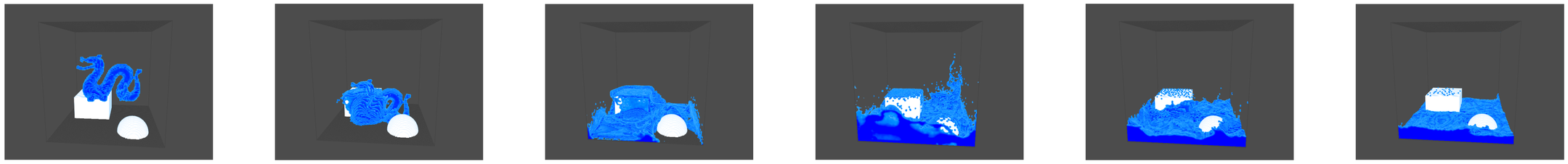
Chocolate Milk: A Fluid Simulation Framework

Dan Knowlton, Advisors: Dr. Norman Badler, Aline Normoyle

Penn Engineering



Abstract

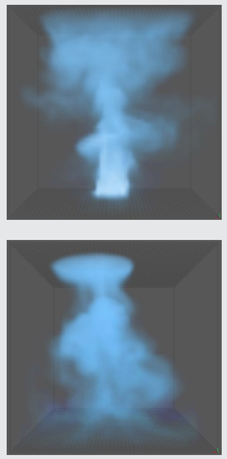
Chocolate Milk is a physically based fluid simulation framework built with C++. The goal of the framework is to break down each stage of fluid simulation into isolated modules that can be easily be connected, exchanged, and replaced. The framework includes multiple simulation and rendering methods.

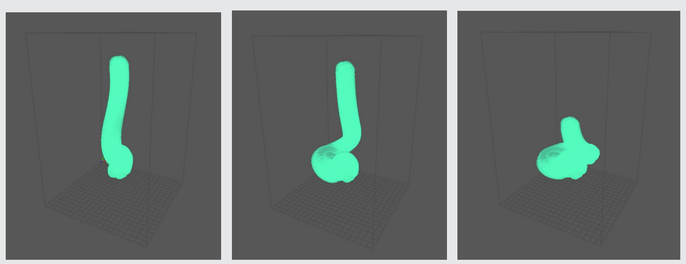
Features

At its core, Chocolate Milk is a Marker-and-Cell based fluid simulator which supports many different types of simulation methods and features:

- Simulation Techniques:
  - Particle-In-Cell
  - Particle Levelset
  - Fluid Implicit Particle Method
  - Smoke (density/temperature)
- Preconditioned Conjugate Gradient Matrix Solver
- Runge Kutta 2nd Order Advection
- Viscosity
- Complex OBJ boundaries and pressure projection solve
- Signed Distance Field Renderer

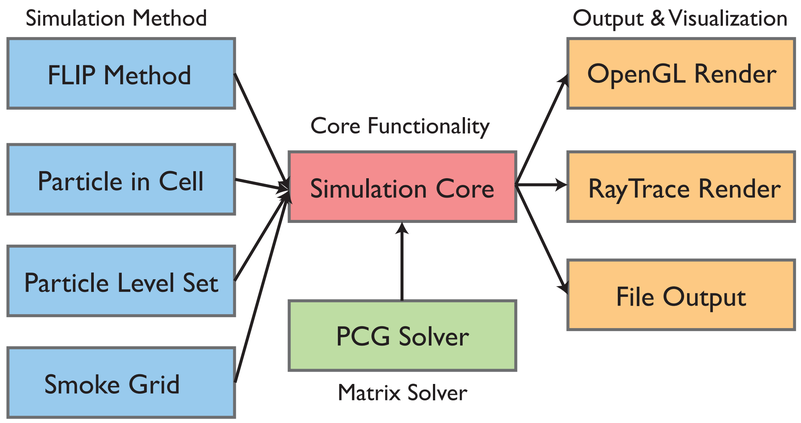
Smoke Render





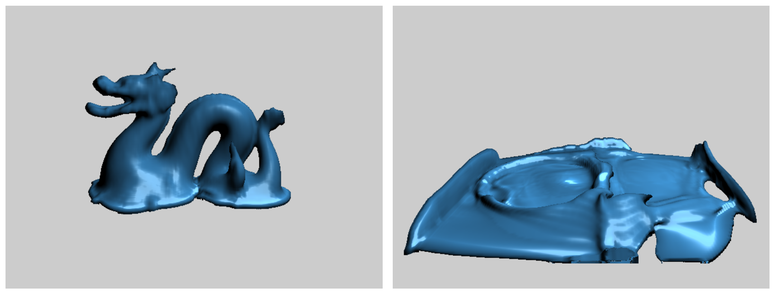
Viscosity Implementation

Framework Design



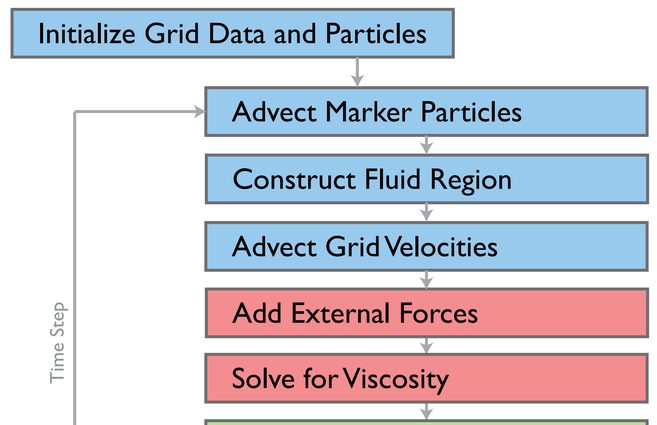
The main design of the framework focuses on creating a series of independent modules that can be interchanged. The simulation core maintains all of the simulation data structures and shared algorithms. A variety of simulation methods can be connected to the simulation core, and the final output of the simulation can be handled in a variety of rendered and data outputs.

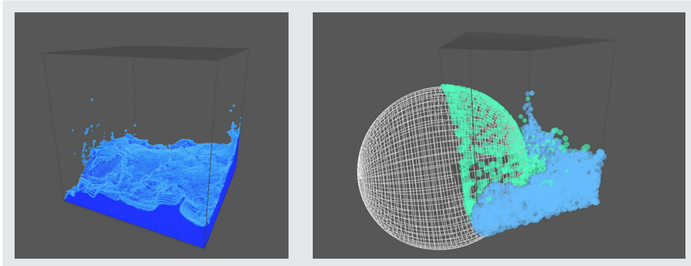
Rendering



Rendering of the fluid simulation is handled by raytracing the fluid grid directly using a raymarching technique. On the grid, a signed distance field is maintained with negative values representing regions inside of the fluid. Once a ray has intersected the fluid region, the gradient of the distance field can be computed to calculate a surface normal for lighting calculations.

Sample Simulation Loop





OpenGL Levelset Visualizer

OpenGL Particle Visualizer

Future Work

Plans for the next stages of the framework's development include: full integration with a pathtracing renderer, a possible GPU implementation for some of the modules, more sophisticated handling of "splashes" and escaped particles, and modeling more physical phenomena such as surface tension.