Basic Fluid Simulation

Christian Oeien

August 22, 2016

Abstract

Implementation of a 2D fluid simulator is presented. A fine-resolution picture flows on the fluid and the viscosity and density of the fluid at a position is mapped from the color at the moving picture.

1 Introduction

Computer simulation of continuous phenomena in general has two main problems. Firstly a mathematical model must be provided that predicts the phenomenon. This may be a subset of the physical laws involved. Secondly an implementation must evaluate a finite set of discrete calculations that renders a simulation. Care must be taken to prevent that numeric rounding nullifies an expected result, or that instabilities arise from the periodicity introduced by taking finite steps.

When moving the picture along the fluid, I use randomized sub-pixel position offset, a technique generally referred to as the Monte-Carlo method. The mentioned instability may be reduced by using the Leap Frog technique.

The Navier-Stokes equations describes the mechanics of a fluid in general. My simulation considers a Newtonian fluid. Principles modeled by my simulator are the pressure-velocity relation, advection, venturi-effect and rotation. Calculations are done in synchronous steps on a grid of cells.

2 Material and Methods

By using generic programming, as the C++ template system provides, I factored out the cell-system. This was specialized in an implementation of a wave-simulation as well as the fluid simulator as reported here.

The generic framework takes a cell-type implementation as parameter. The cell-type contains the physical variables and interactions its with four neighboring cells.

3 Results

The implementation is found at `HTTP://github.com/biotty/rmg/graphics/flow/fluid.cpp` and the generic header-files in the same directory.

Each cell in the flow-grid is responsible for mathematically modelling the physics of the flow, in this case the formulae and state for the physics of a fluid at a certain viscosity and density. The grid itself is part of the generic program and is agnostic to any physics.

I found that the principle of advection is critical for modeling a fluid; without advection the movements in the flow are not fluid-like. The venturi-effect is then needed to make any turbulence happen as naturally in a fluid. The correctness of the movement, especially when turbulent, is further improved by modelling rotation.

The rendering of a picture on the fluid is done at a different resolution than the fluids cell-grid and the movement is interpolated with a sub-cell random offset. This prevents some pixels to be smeared out due to pixel-rounding.

The fluid-parameter feedback is based on color in a cell, and in this case we need to take the most common color in the cell. The picture is color-mapped with 256 color entries. I initially quantize the input-picture using an external quantizer tool.

Input forces at given locations are used to accelerate the fluid in order to create movement. Viscosity and density for certain colors are also provided as input, as well as default values.

4 Discussion

It is not difficult to trigger instability in this simulator, by providing big values for forces. But if velocities are given instead of forces, instability happens too easily to be practical. Care must be taken with any fluid model, but the 'leap frog' technique would reduce likelihood of this problem occurring. I do not implement that method, and instead take synchronous steps of the physical parameters in the cells. In this way, the physics being modeled by the cells are clear to see in the implementation, and I believe it would be obscured by the more stabilizing numerical technique. It is not difficult to provide inputs that results in a stable simulation.

The fluid model and the generic cell-framework are both fixed at 2D. The physics are the same and would be trivial to translate to 3D, but the implementation would contain more math and structures compared to the physics formulae. Therefore 2D is more effective to demonstrate the physical correctness. Also, the visual outputs are even easier to study and appreciate in 2D. The computation-time would also increase significantly if done for 3D.

Non-Newtonian fluids change parameter with the amount of stress or tension in the fluid. My model is Newtonian, but note that the fluid-parameters are parametric per location, and are subject to change while the fluid moves.