Fluid Simulation with FLIP Method

Weigeng Peng[*]

University of Toronto

Abstract

This paper talks about the implementation of FLIP method for fluid simulation using a hybrid Lagrangian/Eulerian method.

Keywords: Fluid, FLIP, Animation

1 Introduction

Fluid is a part of our daily life. The running water from a faucet, the steam released during cooking are all part of the fluid which we're interested in simulating. In this paper we take a generic approach starting with a block of fluid dropping down in midair. The fluid is constrained in a 2D plane. We assume that the only external force is the constant gravity.

2 Related Work and Background

Many work has been done for fluid simulation. [Stam 2001] provides us with a method for unconditionally stable fluid mainly for smoke. [Foster and Metaxas 1996] uses particle in cell method to simulate liquid fluid. Then [Zhu and Bridson 2005] uses interpolation of PIC and FLIP for fluid simulation. We mainly follow [Bridson 2008] for our implementation of fluid simulation.

3 Method

3.1 Hybrid Lagrangian/Eulerian method

Lagrangian method is a particle method, that treats the fluid element as individual particles, in this view we have information about the particle position and velocity. Eulerian method divides the region into cells with fixed position that carries information about the velocity vector field and pressure in each cell. In Eulerian view we store the pressure at the center of the cell, and the velocity component at the edge of the cell. For our implementation, we will start in Lagrangian method to perform advection. Then we covert particles to Eulerian method to solve for pressure. We follow a slight variation as the procedure introduced in [Stam 2001]

$$u_0 \xrightarrow{\text{advect}} u_1 \xrightarrow{\text{force}} u_{2\,\text{particle}\to\text{grid}} \xrightarrow{\text{pressure projection}} u_{3\,\text{grid}\to\text{particle}} \quad (1)$$

3.2 Incompressible Navier-Stokes Equation

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u} - \nu\nabla^2\mathbf{u} = -\frac{1}{\rho}\nabla p + \mathbf{g} \quad (2)$$

Where $\mathbf{u}$ is the velocity vector of the fluid, ν is the kinematic viscosity, p the pressure, ρ the density of the fluid, and g gravity. For the purpose of this paper, we assume the fluid is inviscid. We can therefore drop the third term from Eq. 2. From the incompressible condition, we also have a divergence-free constraint

$$\nabla\cdot u = 0 \quad (3)$$

3.3 Advection

We start our algorithm with advection by considering

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u} = 0 \quad (4)$$

[*]e-mail:weigeng.peng@mail.utoronto.ca

In the Lagrangian view, advection is simply

$$x^{t+1} = x^t + \Delta t v^t \quad (5)$$

while keeping the velocity of the particle unchanged.

3.4 External Force

Then we add the gravity components to update the velocity by considering

$$\frac{\partial \mathbf{u}}{\partial t} = \mathbf{g} \quad (6)$$

3.5 Pressure

Finally we convert from Lagrangian to Eulerian view to solve for pressure

$$\frac{\partial \mathbf{u}}{\partial t} = -\frac{1}{\rho}\nabla p \quad (7)$$

Imposing condition from Eq. 3

$$u^{t+1} = u^t - \frac{\Delta t}{\rho}\nabla p \quad (8)$$

$$0 = \nabla\cdot u^{t+1} = \nabla\cdot u^t - \frac{\Delta t}{\rho}\nabla^2 p \quad (9)$$

$$\nabla\cdot u^t = \frac{\Delta t}{\rho}\nabla^2 p \quad (10)$$

This is the a Poisson equation which we can solve using Conjugate Gradient method. We can find pressure that will give us a divergence-free velocity field by plugging the pressure into Eq. 8.

3.6 PIC and FLIP

The PIC method then takes this updated velocity from Eq. 8 to update the particles position. FLIP method updates the particle velocity by taking the difference of u^{t+1} and u^t. This requires us to store the velocity field after the conversion from Lagrangian to Eulerian. It's also common to use a linear combination of these two methods to update the velocity.

$$u^{t+1} = (1-\alpha)u^{t+1}_{\text{PIC}} + \alpha u^{t+1}_{\text{FLIP}} \quad (11)$$

3.7 Conversion Between Lagrangian and Eulerian Representation

To convert particles to a 2D plane, we use bilinear interpolation. For every particle in a cell, we translate and re-scale the cell to vertices $(0,0),(1,0),(0,1),(1,1)$ including the particle and with the respective weight $(1-x)(1-y), x(1-y), y(1-x), xy$ for each corner. This gives us a component velocity vector field, which we can use to construct a system of equation to solve for Eq. 10. To convert from Eulerian back to Lagrangian representation, we use bilinear interpolation again to find the velocity of the particles after pressure projection Eq. 8.

3.8 Boundary Conditions

For this implementation, we use Dirichlet boundary condition. The velocity at the boundaries are set to zero. We need to make sure that after advection every particles are pushed away from the wall.

Another boundary condition to consider for liquid fluid simulation is the free surface boundary condition. That is the pressure of the surface of the fluid is zero. To achieve this we have to first construct a level set

$$\phi(\mathbf{x}) = \begin{cases} > 0, & \mathbf{x} \text{ outside fluid} \\ = 0, & \mathbf{x} \text{ on the surface} \\ < 0, & \mathbf{x} \text{ inside fluid} \end{cases} \tag{12}$$

We do this defining a kernel

$$k(s) = \begin{cases} (1-s^2)^3 & s < 1 \\ 0 & s \geq 1 \end{cases} \tag{13}$$

We can construct the level set by

$$\phi = \left\| \mathbf{x} - \frac{\sum_i k(\frac{\|\mathbf{x}-\mathbf{x_i}\|}{h})\mathbf{x_i}}{\sum_i k(\frac{\|\mathbf{x}-\mathbf{x_i}\|}{h})} \right\| - \frac{\sum_i k(\frac{\|\mathbf{x}-\mathbf{x_i}\|}{h})\mathbf{r_i}}{\sum_i k(\frac{\|\mathbf{x}-\mathbf{x_i}\|}{h})} \tag{14}$$

With this level set, we can determine the type of each cell, as air or fluid. If an air cell is next to a fluid cell. We can use linear interpolation to determine the value of pressure in the fluid cell for which the pressure acting the surface is zero, while ignoring the air cells.

3.9 CFL condition

At the start of every iteration, we impose CFL condition so that the particles never moves more than one grid cell distance.

$$\Delta t = \frac{dx}{u_{max}} \tag{15}$$

3.10 Initial Setup and Configuration

In this implementation, I have a 6 by 6 grid cells, with 100 particles. The initial velocity vector field is zero. The boundary is a square with side length 2. Hence each grid cell's size is $\Delta x = \frac{2}{6}$. The radius from Eq. 14 is a constant $r = \frac{\Delta x}{2}$, and $h = 3r$.

4 Result and Conclusion

This implementation is not complete, it still has a few issues that remains to be solved. It's been observed from [Jiang et al. 2015] that PIC method is stable but dissipative, meaning it looses some information. FLIP method is not stable, but non-dissipative. However, in my implementation, both methods behaves about the same. In my implementation, once the fluid drops down to the boundary, it doesn't spread away. From Sec. 3.2 it is assumed that the liquid is inviscid, thus this is not expected. I believe that the issue may be from the free surface pressure solver.

References

BRIDSON, R. 2008. *Fluid Simulation.* A. K. Peters, Ltd., USA.

FOSTER, N., AND METAXAS, D. 1996. Realistic animation of liquids. *Graphical Models and Image Processing 58*, 5, 471–483.

JIANG, C., SCHROEDER, C., SELLE, A., TERAN, J., AND STOMAKHIN, A. 2015. The affine particle-in-cell method. *ACM Trans. Graph. 34*, 4 (jul).

STAM, J. 2001. Stable fluids. *ACM SIGGRAPH 99 1999* (11).

ZHU, Y., AND BRIDSON, R. 2005. Animating sand as a fluid. *ACM Trans. Graph. 24*, 3 (jul), 965–972.