Grid Based Fluid Simulation

Keren He *
McGill University

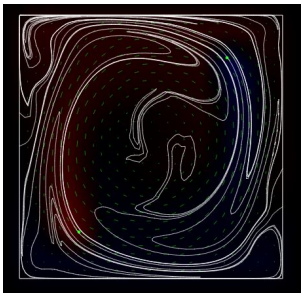

Figure 1: *2 sources implement of my simulator*

Abstract

In this report, I will present my implementation of simulation of viscid fluid using staggered grid combine with semi-Lagrange method. My implementation is mainly based on siggraph 2007 course notes written by Robert Bridson and Real-Time Fluid Dynamics for Games written by Joe Stam et al.

Keywords: Fluids, Physically Based Animation, Grid base

1 Introduction

Fluid like water or gas etc are important phenomenon in our daily lives. Physically-based fluid simulation is beginning to make an impact in real-time games. Incompressible Navier-Stokes equations are used by animators to implement fluid flows such as water or gas. There are 2 equations

$$\frac{\partial \overrightarrow{u}}{\partial t} + \overrightarrow{u} + \frac{1}{\rho \nabla} = \overrightarrow{u} + \nu \nabla \cdot \nabla \overrightarrow{u} (1)$$

$$\nabla \overrightarrow{u} = 0 (2)$$

The first one is the momentum equation which tells us how the fluid acts with various forces acting on it. The second equation is the incompressible equation where we would assume the fluid to maintain a constant volume during the simulation. However, in real world, the fluid does change its volume but these changes are very minimal. For simplicity, we can just assume our fluid to be incompressible. As presented in stams fluid paper and Robert Bridsons course notes, I can split Nvier-Stokes equation into three steps. The steps are advection, body force, and pressure solver combined with incompressibility. In Roberts course note, he used inviscid fluid where he neglected the viscosity of the fluid. In stams paper, he included an extra step, diffusion, to account for the viscosity of the fluid.

*e-mail:ke.he@mail.mcgill.ca

2 Related Work

There are two major approaches on fluid simulation. One is Lagrange approach, also known as partial based simulation. The other is Eulerian approach, also known as grid based simulation. Comparing to particle based approach, grid based simulation is better at tracking smooth surface but is also relatively slower and suffers from mass loss. Particle based simulation treats fluid as small particles. This makes it easier to account for the conservation of mass. However, it will not look as smooth as grid based approach due to the issues with smooth surface. Robert Bridson released the book Fluid Simulation for Computer Graphics in 2009 which presented a complete grid based method to simulate fluid such as water and gas. In 1999, Joe Stam published Stable fluids and introduced the idea of semi-Lagrange where we can advect the velocity for each grid using particles. Later in 2003, he published Real-Time Fluid Dynamics for Games, a simpler version of his paper in 1999, where he presented a simple and rapid simulation of 2D fluid for game engine.

3 Technical Details

3.1 Overview

Here I will present the skeleton of the algorithm I used for grid based fluid simulation.
While simulating
Calculate timestep
Get source from user interface
Add internal force
Advect velocity
Pressure projection
Advect temperature

3.2 Choosing a Timestep

The CFL condition can help dealing with the complex situation which I want to simulate fluid as fast as possible while not destabilizing the simulation. According to CFL, I need to choose a timestep such that any particle inside the velocity field will only

be able to move within one grid length distance. The equation is

$$\triangle t = k_{CFL}\frac{\triangle x}{\overrightarrow{u}_{max}}$$

k_{CFL} is a factor of peoples choice to bust up the simulation time, in foster and fedkiws paper kcfl factor can be up to 5. I choose to use max value equation presented by Bridsons notes where he considered that the body force will affect the simulations time step and in my simulation, the initial velocity is zero for the source, directly dividing the max velocity will cause a computation error. The equation I used to calculate the max velocity is the following

$$\overrightarrow{u}_{max} = max(|\overrightarrow{u}|) + \sqrt{\triangle x|\overrightarrow{F}|}$$

3.3 Model the fluid as staggered grid

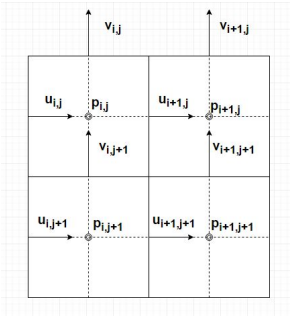

Figure 2: *2D Mac grid cell*

Staggered grid fluid modeling was first introduced by Harlow and Welch in 1965. They used maker particles to simulate fluid with a free surface. The maker particles method is no longer popular to track the surface between air and fluid, yet staggered grid structure is still very useful in fluid simulation, as using this method can resolve the problem of velocity and pressure decoupling. In the project, I used staggered grid where my simulation field is partitioned into (N+2) *(N+2) staggered grids, N is the number of staggered grid in a row or column. As we did in assignment 4, we stored all information in the center of the grid, staggered grid stored information such as pressure and temperature in the center. The main differences about staggered grid is that it stores horizontal velocity at the left and right edge of each grid and vertical velocity at top and bottom of each grid (See figure1). It is therefore redundant to store velocity at both sides of the grid. In my implementation, I stored vertical velocity at the center of the top edge of each grid and horizontal velocity at the center of the left side of each grid.

3.4 Advection of velocity

I used semi-Lagrange method introduced by Stam. The third order of Ruttan Kutta method (RK3) was used to trace back the particles. As we did in assignment 4, we used forward Euler method to trace back particles. However, forward Euler is extremely unstable due to the fact that the eigenvalues of the Jacobian generated by the central differences are purely imaginary, thus always outside the region of stability. As discussed in many papers, a minimum of RK2 level of stability is required to simulate a stable fluid. Therefore, I decided to use the RK3 method in my project.

$$k_1 = f(y^n)$$

$$k_2 = f(y^n + \frac{1}{2}\triangle tk_1)$$

$$k_3 = f(y^n - \triangle tk_1 + 2\triangle tk_2)$$

$$y^{n+1} = y^n + \frac{\triangle tk_1}{6} + \frac{4\triangle tk_2}{6} + \frac{\triangle tk_3}{6}$$

If we treat the center of the grid as a particle, the pressure and temperature information at the time step n+1 of the center of the grid is determined by such information possessed by such particle in its previous time step, n. The same method of Velocity update as in assignment 4 was used, but as different particles locations at time step n+1 were used, so I simply used the location where the value is stored.

3.5 Interpolation of scaler field

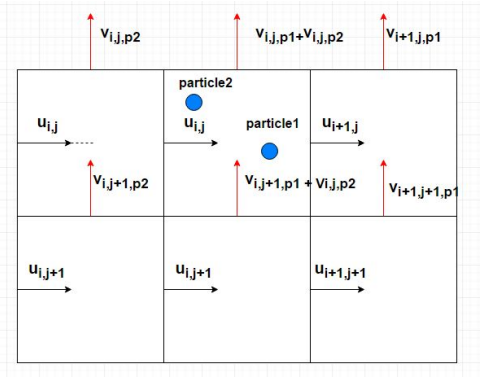

Figure 3: *interpolation of verticle velocity*

Comparing to interpolation of scalar field in assignment 4, it is more complex to interpolate the velocity field for staggered grid. It is not the bilinear interpolation that makes this step complex, but rather the index of the array and where to allocate the scalar field based on the particles locations. Based on the location of the vertical velocity at the top center of the grid, a particle position at (x0, y0), and a grid size of dx * dx, the index of the first cell (the top left cell) used should be i = floor(x0 / dx - 0.5) and j = floor(y0 / dx) because the index of the first grid would only be changed if a particle is moved from the left side to the right side of the cell. In terms of horizontal velocity, the index of the first cell that should be interpolated has a value of i = floor(x.x / dx) and j = floor(x.y / dx - 0.5) This is similar to the vertical case where the values of i and j are only affected if the particle is positioned at the top half and the bottom half of the grid.

3.6 Pressure solve with Incompressibility

Velocity needs to be updated according to the pressure at each time step. The equations are

$$\overrightarrow{u}_{i,j}^{n+1} = \overrightarrow{u}_{i,j}^{n} - \triangle t\frac{1}{\rho}\frac{p_{i,j} - p_{i-1,j}}{\triangle x}$$

$$\overrightarrow{v}_{i,j}^{n+1} = \overrightarrow{v}_{i,j}^{n} - \triangle t\frac{1}{\rho}\frac{p_{i,j} - p_{i,j-1}}{\triangle x}$$

Thus, we can take the incompressible equation of Navier-Stokes equations where the divergence of velocity should be equal to zero and expanded to

$$\nabla \cdot \overrightarrow{u} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}$$

We can approximate the divergence using discrete finite central difference which is

$$(\nabla \cdot \overrightarrow{u}^{n+1})_{i,j} \approx \frac{\overrightarrow{u}_{i+1,j}^{n+1} - \overrightarrow{u}_{i,j}^{n+1}}{\triangle x} + \frac{\overrightarrow{v}_{i,j+1}^{n+1} - \overrightarrow{v}_{i,j}^{n+1}}{\triangle x}$$

. By combining the two information above, I will be able to solve the pressure for each grid. The combined equation for updating pressure is

$$(4p_{i,j} - p_{i+1,j} - p_{i,j+1} - p_{i-1,j} - p_{i,j-1})$$
$$= -(\triangle x((u_{i,j} - u_{i-1,j}) + (v_{i,j} - u_{i,j-1})))$$

Here I will neglect timestep and rho since I am already doing the simulation in the iterations of time, and I am not simulating more than one type of fluid so I can simply assume the density of the fluid ito be 1. I am solving the pressures for all the grids in my simulation field so that I can put the whole system into $A\overrightarrow{x} = \overrightarrow{b}$ form. In such form, each row of A is corresponding to the coefficients of the left side of the combined equation and it is obvious that matrix A is symmetric. Vector $\overrightarrow{x}$ is the pressure for all the grids. Vector $\overrightarrow{b}$ is the left side of the combined equation which I can get the values from the velocity field. The only question left is to find out how to solve for $\overrightarrow{x}$. There are many existing methods for solving $\overrightarrow{x}$. Among these methods, I used the Modified Incomplete Cholesky Conjugate Gradient Level 0 which is recommended by Bridon.

3.7 Boundary conditions

I used the simplest boundary condition described in Stams Real-Time Fluid Dynamics for Games paper. Since fluid is simulated in an empty box and in staggered grids, it is simple to set the velocity at the boundary to be zero. For other quantities such as pressure and temperature information, are stored at the center of the grid. Therefore, the solid grids at the boundary have the same pressure, temperature and density values of the fluid cell they are next to.

4 Results

Figure.1 shows an example of my fluid animation with two sources which RK3 was used to trace back the particle quantities in the advection step, and the Modified Incomplete Cholesky Conjugate Gradient level zero (mcg) was used to solve the pressure. More sample runs are provided in the demo video. In theory, using Gauss-Seidel will be less accurate than using mcg for pressure solving. Indeed, when I run my simulation in forward euler particle trace method for both Gauss-Seidel and mcg, the residual of mcg converges to be smaller than 1.2*10-3 in roughly 30 iterations, while the residual for runnig Gauss-Seidel 30 times is roughly 1.7*10-3. Therefore, mcg is slightly more accurate and faster in terms of iteration. For both methods, the residual is the infinity norm of d-Ax. However, if we look at the overall time of simulation, mcg is slower than Gauss-Seidel because mcg is doing much more calculations than Gauss-Seidel during the implementation for each iteration. There are computational errors for all the calculations, and I believe that is why mcg and Gauss-Seidel accuracy are not differed by much. In theory, comparing to RK3, using forward euler in particle tracing is unstable and inaccurate especially when time step is very big. Since CFL condition involves particle tracing method, in this comparison I decided to hard code my time step size to be 0.5 and the pressure solver to be mcg for more accuracy. I let both methods run for more than 400 total time steps, and I observed that RK3 is much more stable than forward euler. After running 400 total time steps, forward euler completely lost control and rapidly swang left and right from source in a huge angle, on the other hand RK3 swing left and right from source in a much smaller angle. Initially I was thinking of implementing the smooth free surface of fluid but it is very difficult for me to understand the implementation of implicit surface function and implementation of extrapolation of velocity into air and solid wall so I decided to leave it as future work.

5 Conclusions

In my project, I have implemented a simple 2d grid based fluid simulator in a box which is an extension of Assignment 4. For the future, I am looking forward to implementing a smooth free surface for water simulation while using a higher rendering technic at the same time to simulate real time fluid.

References

BRALEY, C., TECH, V., AND SANDU, A. 2009. Fluid simulation for computer graphics: A tutorial in grid based and particle based methods.

BRIDSON, AND MLLER-FISCHER, 2006. Fluid simulation for computer animation. ACM SIGGRAPH 2006 Course.

ENRIGHT, D., LOSASSO, F., AND FEDKIW, R. 2005. A fast and accurate semi-lagrangian particle level set method. *Computers and Structures 84*, 6 (Feb./), 479–490.

FOSTER, N., AND FEDKIW, R. 2001. Practical animation of liquids. In *Proceedings of SIGGRAPH 2001*, ACM Press / ACM SIGGRAPH, E. Fiume, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM, 23–30.

STAM, J. 1999. Stable fluids. In *Proceedings of SIGGRAPH 1999*, ACM Press / ACM SIGGRAPH, W. Longman, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM, 121–128.

STAM, J., 2003. Real-time fluid dynamics for games.