Galilean Invariance for Fluid Simulation

Maurya Shah[1] Frédéric Pighin[1] Jonathan Cohen[2] Penne Lee[1]
[1]University of Southern California, Institute for Creative Technologies
[2]Rhythm and Hues

Introduction

In computer graphics fluids are often simulated by solving the Navier-Stokes equations. In our implementation we use the following form of the Navier Stoke equations for incompressible fluids:

$$\begin{aligned}\frac{\partial u}{\partial t}+(u\cdot\nabla)u+\nabla p &= \nu\nabla^2 u-(T-T_0)z+gz+\varepsilon h(N\times\omega)\\ \nabla\cdot u &= 0\\ \frac{\partial T}{\partial t}+(u\cdot\nabla)T &= k\nabla^2 T\end{aligned}$$

These partial differential equations are solved on a grid of voxels. This grid is a discrete representation of three-space. Unfortunately fluid simulations are notoriously difficult to predict. In particular slight variations in the initial conditions can have dramatic impact in turbulent flows. This makes the problem of deciding a grid size for the simulation extremely difficult. With open boundaries, the fluid quickly dissipates when it hits the boundaries of the simulation domain (as in figure 1 (a)). To prevent this, large simulation domains have to be used to keep the fluid away from the boundaries of the simulation. In this work, we describe a novel technique for implementing an adaptive grid for fluid simulation. The technique is largely based on the principle of Galilean Invariance.

Galilean Invariance

We propose to use the principle of Galilean Invariance to speed up fluid simulations. Galilean Invariance states that the laws of physics remain the same if the coordinate system is moving with a uniform speed. We take advantage of this property to adapt the simulation grid to the motion of the fluid. Our simulator is an implementation of Josh Stam's semi-Lagrangian technique [Stam 1999] with open boundaries. Our adaptive scheme both translates and reshapes the simulation domain according to the evolution of the fluid.

Translating the grid

At each time step we predict the velocity of the geometric center of the fluid. The simulation grid is then re-centered around the location of this point. This technique allows the simulation of buoyant flows with a small grid of voxels. As the fluid moves in space so does the simulation domain. This way the fluid never hits the boundaries. To display the simulation results we translate the computed values according to the trajectory of the geometric center. This transformation corresponds to the change of coordinate system $x(t)\rightarrow x(t)+\int_0^t v(\tau)d\tau$, where $v(\tau)$ is the velocity of the moving domain. Figure 1 (b) shows the result of moving the domain along with the fluid flow. It's easy to see that a much bigger simulation grid would have been required to achieve the same simulation with a static grid.

Reshaping the grid

A more efficient approach is to reshape the voxel grid according to the state of the simulated fluid. We use the boundary conditions as a criterion to determine the shape of the grids. Voxel areas which meet the boundary conditions are discarded from the simulation. The voxel grid is expanded to give room to the fluid expansion. Figures 2 shows the result of the fluid simulation with a translating and a reshaped grid at different stages of the simulation. In this figure the green areas correspond to active voxels which participated in the simulation.

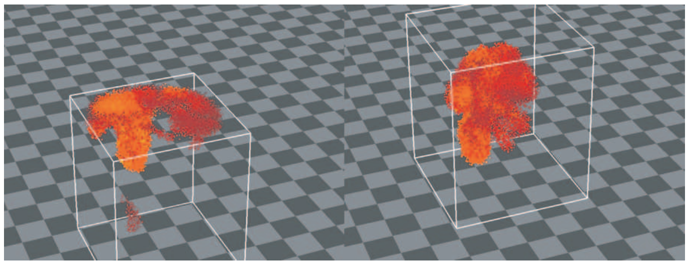

Figure 1: Domain translation: (a) fixed grid, (b) translated grid.

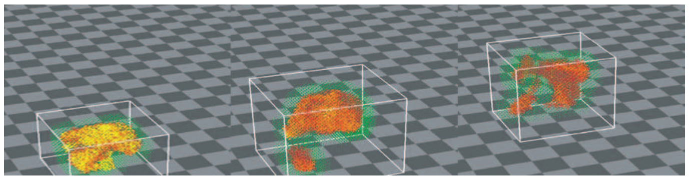

Figure 2: Domain reshaping.

We perform the translation and reshaping of the simulation domain without copying simulation results in memory. Only the specification of the simulation boundaries is changed. This is a much more efficient implementation. The transformations we perform on the simulation domains are not without effects on the simulation results. Some amount of dissipation occur at the boundaries. It affects the simulation in the long run. However this dissipation is small because we try to maintain the boundaries away from the fluid. As a result the visual quality of the flow remains the same.

Conclusion

The use of these two techniques in fluid simulation yields significant savings in memory and computation over traditional approaches while generating flows with similar visual quality. To our knowledge, it is the first time that the principle of Galilean Invariance is used to speed up fluid simulation.

References

STAM, J. 1999. Stable fluids. In *Proceedings of SIGGRAPH 99*, 181–188.