Fluid Simulation Via Disjoint Translating Grids

Sanjit Patel* — USC-ICT
Anson Chu — USC-ICT
Jonathan Cohen — Rhythm and Hues Studios
Frederic Pighin — USC-ICT

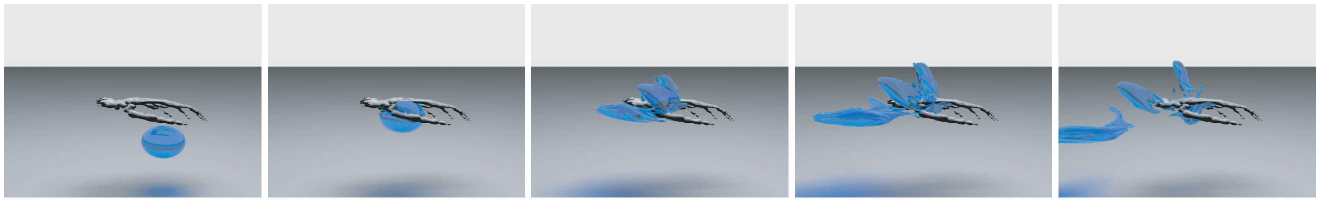

Figure 1: A ball of water collides with a skeleton hand. The frames in this animation were efficiently computed using our technique: the ball and the splashes belonged to domains that were merged and split to minimize computation cells.

We present an adaptive fluid simulation technique that splits the computation domain in multiple moving grids. Using this technique, we are able to simulate fluids over large spatial domains with reasonable computation times.

Motivation

Fluid dynamics describes the behavior of smoke, steam, water pouring into a container, ocean waves, fire, clouds, explosions, and many other phenomena that appear frequently in movies, video games, and virtual reality simulations. From a computer graphics perspective, computational fluid dynamics (CFD) techniques can be categorized as either Lagrangian or Eulerian.

Lagrangian techniques translate the Navier-Stokes equations into the material coordinate frame and treat the fluid flow as a particle system, where pressure is an inter-particle force that seeks to preserve uniform particle density. While these techniques excel at representing splashy and particulate fluid flows, they require excessive numbers of particles to simulate smooth water surfaces.

Most computer graphics applications have leaned towards the Eulerian approach, where the Navier-Stokes equations are expressed in a fixed global coordinate frame. Finite-difference methods can be made unconditionally stable with semi-Lagrangian methods as suggested by [Stam 1999], and have been used to simulate gases and liquids. Eulerian methods excel at representing interfaces via Levelset methods, and incompressibility can be enforced efficiently using the projection method. While Eulerian methods can be unconditionally stable, excessively large time steps introduce severe numerical viscosity, and hence ruin physical and visual accuracy. This is why [Enright et al. 2002] recommends limiting the time step to no more than 5 times the Courant-Friedrich-Levy (CFL) condition. For splashy simulations, this can lead to excessively small time steps.

In general the Lagrangian representation is superior for modeling free-falling water in terms of speed, accuracy, and simplicity. The goal of the present research is to exploit this observation to accelerate simulations of splashy fluids, by mixing a Lagrangian method for free-falling water with an Eulerian method for contained water. Our method splits a single simulation into multiple computational grids, where the motions of the computational grids are treated analytically whenever possible (e.g. when splashes are in free-fall). During the simulation, these grids are resized, split, or merged depending on the motion of the fluid to improve computational efficiency and accuracy.

*patels@ict.usc.edu

Our work is an extension of [Rasmussen et al. 2004] where we allow the grids to have arbitrary rectilinear motions and individual adaptive timesteps.

Simulation framework

The power of our approach is to concentrate computational cells and cycles where they are needed. First spacially, our algorithm does not require gridding space that does not contain fluids. Second, since each disjoint grid is simulated independently, each can use a different time step. As a result, a fast moving splash might require small time steps while in the same simulation a slow moving body of water can afford much larger timesteps.

Our algorithm is designed to manage a collection of grids. The grids are used to simulate unconnected volumes of fluid. As the fluid is set in motion, grids might merge (when two volumes of fluid collide) or split (when a volume of fluid breaks into multiple unconnected components). The complexity of our framework lies in managing multiple grids in a way that is physically valid and efficient. Our algorithm revolves around a scheduling queue that contains an entry for each grid. The grids are ordered by increasing timestep, so that the top of the queue points to the grid that has the smallest timestep (as prescribed by the CFL condition). At each step, multiple events can occur.

- Merge. Two grids collide and are merged in a single grid.
- Split. A grid splits into multiple separate grids.
- CFL step. In the absence of merging or splitting, the grid that has the smallest time-step is simulated.

At each iteration, the algorithm checks for potential events and executes them in sequential order. A global clock is maintained and is updated as events are processed. Each grid hosts a local clock that reflects events local to the grid. We use an octree-based approach to decide when to split and merge grids. These data structures allow checking for these conditons at a fine resolution.

References

ENRIGHT, D. P., MARSCHNER, S. R., AND FEDKIW, R. P. 2002. Animation and rendering of complex water surfaces. *ACM Transactions on Graphics 21*, 3 (July), 736–744.

RASMUSSEN, N., NGUYEN, D. Q., GEIGER, W., AND FEDKIW, R. P. 2004. Directable photorealistic fluids. *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (July), 193–202.

STAM, J. 1999. Stable fluids. In *Proceedings of SIGGRAPH 99*, Computer Graphics Proceedings, Annual Conference Series, 121–128.