SIGRAD (2015)
L. Kjelldahl and C. Peters (Editors)

Real-Time Fluids – Optimizing Grid-Based Methods

Jaime Alvarez Losada[†1], Eike Falk Anderson[1] and Oleg Fryazinov[1]

[1]The National Centre for Computer Animation, Bournemouth University, United Kingdom

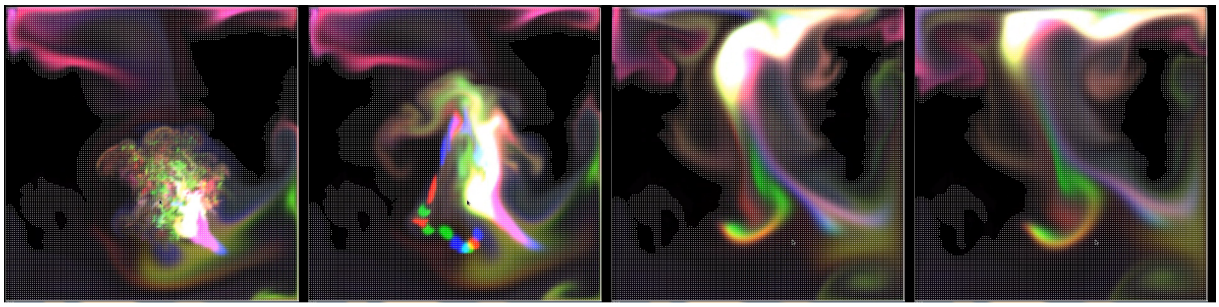

Figure 1: *Several steps of a simulation, showing insignificant cells (black areas) being culled from the simulation.*

Abstract

A fluid simulation suitable for use in real-time virtual environments running at interactive frame rates has the potential to greatly improve the quality of the virtual environments it is used in. To this end we present a method suitable for use in real-time grid-based fluid simulation that considerably reduces the amount of data being processed at each simulation step by removing unused cells from the simulation grid.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation; I.6.8 [Simulation and Modeling]: Types of Simulation—Visual

1. Introduction

Fluid simulation is a major research area in modern computer graphics. Most existing methods focus on offline simulations, where the quality is more important than the speed of the simulation. However, growing demand from the video games industry and interactive media has shifted attention to real-time fluid simulation methods.

Traditionally the methods to simulate fluids for real-time applications can be distinguished into two main categories: Eulerian methods, which are based on the calculation over a discretised simulation space and Lagrangian methods, which are particle-based.

In this work we are focusing on Eulerian-based methods, which are more traditionally researched and which simplify the obtaining of effects such as compressibility of the fluids and diffusion. At the core of Eulerian methods is a fixed space subdivided by a regular grid where the parameters of the fluid (density, velocity, etc.) are calculated per cell, where one needs to find a balance between the quality of the simulation and the resolution of the grid. To increase the performance of fluid simulations based on a regular grid, we are proposing to isolate the cells that contain dynamically changing information and perform the simulation only on these cells.

The main contributions of this work are:

1. A description of the steps to reduce the number of active cells in the simulation that results in more efficient calculations;
2. An explanation of simple queries preceding the simu-

† first-author of this submission is a student

lation step that update the additional cells' information throughout the grid as the simulation progresses (Figure 1).

2. Background

There exist a number of physically-based approaches for fluid simulation. Compared to the popular Langrangian approaches to fluid simulation, an example of which is the recent discussion of Smoothed Particle Hydrodynamics (SPH) by Ihmsen et al. [IOS*14], the body of work related to grid-based approaches is relatively small, possibly because the overall quality of results achieved using particle-based approaches is higher. The grid-based Eulerian approach as presented by Stam [Sta03], however, lends itself particularly well to real-time simulation, as the fixed size of grid cells greatly simplifies the simulation step. This also simplifies the implementation of such simulations as GPU shaders [CLT07].

In grid-based fluid simulations a regular grid of a fixed size enclosing the simulation area is used. In this grid each cell is used to store the fluid parameters (such as velocity, density and temperature) for that particular point in space. At setup of the simulation, the grid cells are initialized to hold the initial values for the fluid parameters, and at each simulation step, changes to the cells will be calculated using the Navier-Stokes equations [Sta03].

For incompressible fluids, adaptive structures are used. Thus, Irving et al. [IGLF06] combine the cells of the simulation into tall slabs to increase efficiency and decrease memory consumption. A similar idea was used by Chentanez et al. [CM11] for water simulation on a large scale. However, the type of adaptive grid structure employed by these approaches may not be ideal for use in real-time fluid simulation, as Kallin [Kal09] suggests that a dynamically changing adaptive grid might be too computationally expensive.

For compressible fluids and smoke simulation, however, these methods are not truly suitable because of the density parameters that must be considered.

3. Method Description

Our method is based on the real-time grid-based approach presented by Stam [Sta03]. For simplicity, we present this in 2D rather than 3D, but the method itself can easily be extended to 3D.

A fluid is modelled as a velocity vector field and density scalar field which are described by Navier-Stokes equations:

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \times \nabla)\mathbf{u} + \nu \nabla^2 \mathbf{u} + \mathbf{f}$$

$$\frac{\partial p}{\partial t} = -(\mathbf{u} \times \nabla)p + k \nabla^2 p + S$$

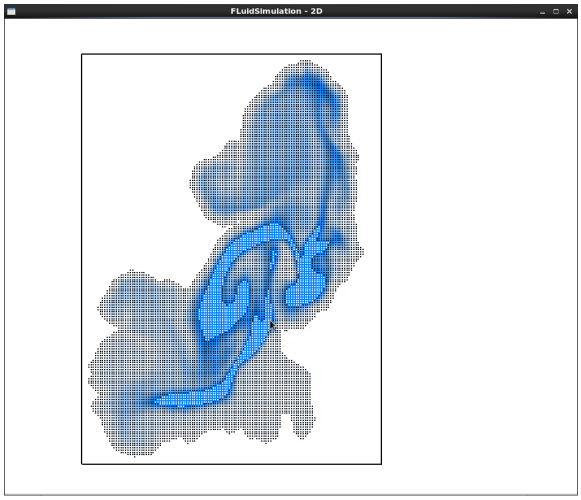

Figure 2: *Cells with active surface/participation flag (black dots) denoting the simulation area, and boundary cells (dark blue outlines).*

Within the data structure, grid cells contain the parameters of the fluid, such as density, pressure, and so on. We extend the data structure for grid cells by a single flag that states whether the given cell should be simulated at the given moment or whether the parameters have to be propagated from neighbouring cells. This is used as an area marker that is set for meaningful cells, i.e. those cells of the grid that contribute to the current time step, which allows the cells that do not contribute at all to be removed from the simulation. The grid size itself is not affected by this and by essentially putting aside the non-required cells, allowing the currently unused grid cells to be returned to the simulation at a later time (Figure 1), we avoid the overhead that would be caused by dynamically growing or shrinking the grid. Through an inexpensive look-up this single flag allows us to quickly retrieve information regarding the areas where the simulation actually takes place on the current frame. This in turn allows, for example, an efficient calculation of the current bounding box for the simulation.

To improve the simulation's efficiency, we perform two additional queries before each simulation step.

- On the first step we find which cells from the given regular grid participate in the simulation, for example those that carry non-zero (within a given precision) density parameters of fluid. The boundary of the grid allows us to find the bounding volume of the simulation and to process only neighbouring cells of cells isolated on the boundary during the next step of the simulation (Figure 1).
- On the second step we find the cells inside the simulation area that have identical or similar (within the given precision) parameters. Formally for two cells with velocities $\mathbf{u}_1$ and $\mathbf{u}_2$ and densities p_1 and p_2 respectively this similarity

can be defined as:

$$|1 - \mathbf{u}_1 \cdot \mathbf{u}_2| < \varepsilon_u$$

$$|p_1 - p_2| < \varepsilon_p$$

Here by ε_u and ε_p we denote tolerance parameters which define similarity. Also we assume that velocity vectors are stored as normalised.
This essentially allows us to restrict the areas for which the full simulation is required only to the boundary cells and to use the values from the boundary cells for the interior cells (Figure 2).

Finally we calculate the simulation step and update the surface flag in cells that become relevant (gain fluid) or lose relevance (no longer contain fluid) to the simulation. The very same step allows us to identify the areas with similar (within the given precision) parameters to set the participation flag for the next simulation step. Note that by varying precision we can obtain larger areas with similar parameters and therefore increase the efficiency of the simulation.

Initial results for two simulations using a 400x400 grid and running on a 64bit Linux machine with 8GB Ram and an Intel Xeon E5-1650 CPU – the first simulation implementing a conventional grid-based fluid simulation, the second adding our improvements to this simulation – showed that in the worst case (all grid cells contributing to the simulation) the performance of our method was no worse than the conventional simulation, fluctuating between 5 and 7 frames per second. In the best case (low velocity fluid, meaning that many cells could be culled) our method allowed the simulation to run about 60 times faster (445 to 440 frames per second), while on average running around 9 times faster than the conventional case (fluctuating between 45 and 60 frames per second). The same set of simulations running on a 64bit Windows machine with 4GB RAM and an Intel Core i5-2450M CPU achieved a similar performance, with the conventional case as well as our method's worst case achieving on average 4.5 frames per second. In the best case our method achieved frame rates fluctuating between 248 and 288 frames per second, while on average framerates fluctuated between 24 and 57 frames per second.

4. Discussion and Future Work

As the evaluation of the surface/participation flag for each cell effectively culls cells that do not contribute to the simulation, in simulations where the fluid does not fill all of the simulation space there is a noticeable improvement that our method achieves in terms of processing time, when compared to existing methods. The worst case, i.e. when the fluid extends to the complete simulation area and all grid cells contribute to the simulation, meaning that the additional queries would have no effect and no longer need to be performed, our method achieves a similar performance to existing methods.

There are further improvements that we have not yet implemented. These are mainly related to which grid cells can be removed from the simulation while maintaining overall fidelity of the simulation. One has to find the best balance between speed and precision to determine setting of our simulation flag. Finding the proper balance for different cases is an area for further research.

References

[CLT07] CRANE K., LLAMAS I., TARIQ S.: Real-Time Simulation and Rendering of 3D Fluids. In *GPU Gems 3*, Nguyen H., (Ed.). Addison-Wesley Professional, 2007, ch. 30. 2

[CM11] CHENTANEZ N., MÜLLER M.: Real-time eulerian water simulation using a restricted tall cell grid. In *ACM SIGGRAPH 2011 Papers* (2011), SIGGRAPH '11, pp. 82:1–82:10. 2

[IGLF06] IRVING G., GUENDELMAN E., LOSASSO F., FEDKIW R.: Efficient simulation of large bodies of water by coupling two and three dimensional techniques. In *ACM SIGGRAPH 2006 Papers* (2006), SIGGRAPH '06, pp. 805–811. 2

[IOS*14] IHMSEN M., ORTHMANN J., SOLENTHALER B., KOLB A., TESCHNER M.: SPH Fluids in Computer Graphics. In *Eurographics 2014 - State of the Art Reports (STARs)* (2014), pp. 21–42. 2

[Kal09] KALLIN D.: *Real-Time Large Scale Fluids for Games*. Master's thesis, KTH, Stockholm, 2009. 2

[Sta03] STAM J.: Real-time fluid dynamics for games. In *Game Developer Conference 2003* (2003). 2