Level of Detail Flow Simulation

Gergely Klár

Department of Control Engineering and Information Technology
Budapest University of Technology and Economics, Budapest, Hungary

Abstract

In this paper we present a framework to simulate visually plausible large scale flow of fluids or smoke. To maintain real-time speed, we define the simulation over a coarse grid which is refined with a more detailed grid at places where fine details may emerge, like around moving obstacles. The detailed grids also act as fixed frames of reference to the surrounded obstacles to prevent the need for working with moving boundaries in the flow.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Fluid simulation

1. Introduction

Modelling the flow of gases, fluids, and smoke has been a field of interest for engineering sciences for a long time. Computational fluid dynamics is a well developed field, but application of its findings for computer graphics started only in the last decade.

Flow simulation can be used to visualise a diverse range of natural phenomena including rivers, smoke or wind.

There are two main approaches to re-create fluids or smoke for computer graphics. The particle based approach animates a large number of elements based on their interactions with each other and the environment. On the other hand, the grid based approach simulates the dynamics of the fluid or smoke in a predefined region.

While these particle based methods are well suited for simulations where are many interactions with the environment and the flow path is complex, grid based methods are particularly important for smoke simulation. The particle based simulation of smoke would require too many particles to produce pleasing effects.

The main driving dynamics of a flow are captured by the Navier-Stokes equations, which describe the motion of incompressible flows. These equations can be decomposed to four principal terms, including *advection*, *diffusion*, *pressure*, and *external forces*.

To be used in computational fluid dynamics, the original partial differential equation is redefined in a finite difference form over grids of scalar and vector quantities. The resolution of the grid defines the amount of details that can appear in the flow.

2. Previous Works

The main challenge in flow simulation for computer graphics is presenting realistic results while keeping the grid resolution as low as possible thus reducing the required computational time. Fedkiw et al. [FSJ01] presented a method called vorticity confinement to reinsert fine details lost due numerical dissipation. This way the grid's resolution can be decreased without loosing important visual features.

Increase in accuracy can be achived by using a staggered grid in which vector quantities are represented at cell boundaries and not in cell centres [GDN98].

Due to the complex nature of the computations involved, running simulations fast enough to allow the user to interact with the flow in real-time presents further challenges. The evolution of modern graphics hardware gave birth to methods and principles enabling partial differential equation (PDE) solving on the GPU at much higher rates. As Harris presented in his work [Har05], the driving dynamics of a flow defined by the Navier-Stokes equations can be efficiently and easily computed in GPU. The highly parallel nature of the computations involved makes the GPU a really suitable and fast platform. The use of texture images as a discretized approximations of the vector and scalar fields is a key concept for PDE solving on GPU, thus for the principle of fluid dynamics simulation on the GPU as well.

Harris' work utilizes an implicit method based on Stam's

© The Eurographics Association 2008.

delivered by EUROGRAPHICS DIGITAL LIBRARY www.eg.org diglib.eg.org

Stable Fluids [Sta99], which is required because of the nature of GPGPU computations and which also provides a stable simulation for arbitrary large time-steps and velocities.

3. Hierarchical Grids

To enable larger scales of simulated flow whilst keeping computational costs low, we define a level of detail hierarchy over the scalar and vector fields used in the equations.

Let's denote the coarse grid covering the whole field as the main grid, and denote the smaller, but refined grid as the sub-grid. The sub-grid overlays a portion of the main grid, but in such manner that several cells of the main grid are covered by the refined grid as shown in Figure 1. If the proportion of the grids were such that the sub-grid covers only a few of the main grid's cells, it would lead to sharp changes at the edges of the sub-grid.

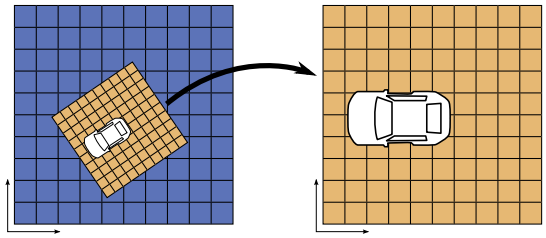

Figure 1: *Relation of the main grid and the sub-grid.*

The sub-grid is defined to surround the obstacles that might cause fine detailed changes in the flow. The obstacle should be fully contained in the sub-grid and additional space is required for the details to emerge between the obstacle boundaries and the sub-grid's edges.

The sub-grid is fixed to the object it contains, and its frame of reference follows the objects, should it be moving. This will result in a difference between the grid's and the sub-grid's frame of reference, that should be addressed during computations. The benefit of fixing the frame to the object is that this way we can tackle the need to work with moving obstacles and the boundary conditions have to be calculated only once.

The result of the prior constraints is that only one object or objects moving together should be contained by a single grid. Allowing multiple, independently moving obstacles in the same grid would be contradicting to our aim to avoid the need of working with moving obstacles.

Admittedly, multiple obstacles can be enclosed by separate refined grids, but these grids should never overlap during the simulations.

The flow in the sub-grid emerges from the difference between the frames of references. Zero velocities in respect to the main grid's frame are non-zero velocities in a moving

```
for each advectant of the main grid:
  advect(advectant);
advect(velocity);
project(velocity);

for each advectant of the sub-grid:
  advectOnSubGrid(advectant);
advectVelocityOnSubGrid(subGridVelocity);
correctVelocities();
project(subGridVelocity);

updateTime();
updatePosition();

feedback();
```

Listing 1: *Detailed simulation step*

sub-grid's frame. We address this difference by adding complementary velocities to the sub-grid. Our aim is to modify the sub-grid's velocities to counter the motion and to keep zero velocity parts of the grids still and non-zero velocity parts moving the proper direction. We have to deal with linear and circular motion of an obstacle separately, because circular motion could be approximated with linear motion only if the motion's centre is far off from the sub-grid.

In case of linear motion the complementary velocity is uniform over the sub-grid, and equals the inverse of the obstacle velocity, rotated and scaled to the sub-grid. For circular motion the complementary velocity is derived from the angular velocity for each cell of the sub-grid.

4. Implementation

Each simulation step consists of the following operations: simulation step on the main grid, then simulation step on the sub-grid, repositioning of the sub-grid and finally sub-grid to main grid feedback. Listing 1 details these operations.

Each step of the main grid's simulation is computed as in Harris' implementation [Har05]. Projection computations are the same for both grids. Repositioning of the sub-grid is done by moving it according to the motion of the contained obstacle. Other operations on the sub-grid are detailed in the following sections.

4.1. Advections

The advection algorithms in the sub-grid are different if the advectant is the velocity field and if it is some other field. In either case, values from the main grid are taken into account, but during the velocity field's advection values read from the main grid are converted to the sub-grid's frame of reference.

In general the implicit method for advection of a field $\mathbf{u}$ by the velocity field $\mathbf{v}$ at point p with time-step Δt is

© The Eurographics Association 2008.

$$\mathbf{u}_{new}(p) = \mathbf{u}(p - \Delta t \mathbf{v}(p))$$

To advect fields of the sub-grid other than the velocity field the following equation is used:

$$p' = p - \Delta t \mathbf{v}(p) \tag{1}$$

$$\mathbf{u}_{new}(p) = \begin{cases} \mathbf{u}(p') & \text{if } p' \text{ is inside the sub-grid,} \\ \mathbf{U}(\Phi(p')) & \text{otherwise} \end{cases} \tag{2}$$

where $\mathbf{U}$ is the corresponding field of the main grid and Φ is the sub-grid to main grid coordinate transformation.

The sub-grid's velocity advection includes an additional transformation Θ to convert velocities from the main grid's frame of reference to the sub grid's, and a vector $\mathbf{w}$, the complementary velocity. The complete equation for velocity advection in the sub-grid is the following:

$$\mathbf{v}_{new}(p) = \begin{cases} \mathbf{v}(p') & \text{if } p' \text{ is inside,} \\ \Theta(\mathbf{V}(\Phi(p'))) + w & \text{otherwise} \end{cases} \tag{3}$$

where p' is as in equation (1) and $\mathbf{V}$ is the main grid's velocity field.

For each point in sub-grid's velocity field $\mathbf{v}$ we can decompose the velocity to a regular and a complementary part.

$$\mathbf{v}(p) = \mathbf{v}_r(p) + \mathbf{v}_c(p)$$

In equation (3) vector w equals to this $\mathbf{v}_c(p)$, and its value is defined by the motion of the sub-grid.

If the sub-grid is still, then $\mathbf{v}_c(p) = 0$ for all p. If it is making linear motion, then $\mathbf{v}_c(p) = \Theta(-v_o)$ for all p, where v_o is the obstacle's velocity and Θ is as in equation (3). If the sub-grid is making circular motion, then $\mathbf{v}_c(p)$ is different for each p.

In case of circular motion, velocity $\mathbf{v}_c(p)$ is derived from the angular velocity. Because the implicit method used, $\mathbf{v}_c(p)$ has to be defined for each p, so $p - \Delta t \mathbf{v}_c(p)$ and p lie on the same circular path.

Using the tangential velocities would result in an inward spiral instead of a perfect circle, since quantities from circular paths having larger radii would carried to paths of smaller radii. Therefore we define $\mathbf{v}_c(p)$ as the chord between p and the point time-step times the angular velocity degrees back in the circular path. The following equation defines this chord.

Let c_r denote the centre of rotation, ω the angular velocity, Δt the time-step of the simulation, and let $r = p - c_r$.

$$r' = \text{rotate}(r, -\omega \Delta t)$$

$$\mathbf{v}_c(p) = (r - r') \frac{1}{\Delta t}$$

If $\mathbf{v}(p) = \mathbf{v}_c(p)$ for each p with the $\mathbf{v}_c(p)$ defined above, the advection defined in equation (2) will cause the advectant to be carried along perfectly in circles around the centre of rotation.

During the advection of the velocity field the velocities have to be corrected after each advection. The magnitudes of the velocities before and after are the same, but their directions are different. To redeem this, the velocities of the sub-grid have to be rotated toward the centre of the circular motion as if it were centripetal acceleration.

This can be viewed as rotating the velocity field's vectors to follow the rotation of the grid.

4.2. Feedback

Because every level of the hierarchy on the whole depend on an other one, efficient feedback of the simulated quantities is crucial. The sub-grid's dependence on the main grid is handled during advection. Feedback addresses the converse.

During feedback, for each cell of the main grid the corresponding cells, if any, of the sub-grid should be determined, their values filtered, and should be registered back to the main grid. Fortunately this task can be committed to the GPU by rendering a full-screen quad for the main grid and then rendering a scaled and properly aligned quad for the sub-grid. The rendering is done using a simple pass-through fragment program for the scalar fields such as inks or smoke densities. The resulting frame is the result of the current simulation step and should be used in the subsequent step for the main grid.

However, if the advectant is the velocity field itself, and additional step is required to transform the sub-grid's velocities to the main grid's frame of reference. For this step we render the sub-grid's velocity field to a temporary texture. The fragment program used for rendering first subtracts the complementary velocity, then converts the resulting velocity to the main grid's frame of reference for each fragment.

4.3. Diffusion

The discrete nature of the methods results in diffusion of the advected quantities, particularly of velocities. This diffusion causes distortion during circular motion, and makes the advected quantities to spiral inward.

To resolve this problem, the advected velocities should be boosted by a small amount, using a dissipation term in advection. The exact boosting rate heavily depends on the other parameters of the simulations, but can be found easily by experimenting.

© The Eurographics Association 2008.

5. Results

The main achievement of our work is that large scale flows with relatively small moving elements can be simulated without a demanding simulation of large scalar and vector fields while preserving fine details.

To evaluate the performance of the hierarchical grids method we made several experiments. First, to determine a reference value for the frame rate, we run the simulation on a single, 512×512 grid that run at 7 frames per second. Than we run the experiment using hierarchic grids of several size ratios and resolutions. The size ratio defined the resolution of the sub-grid, so the sub-grid could be considered as a portion of the grid of the reference experiment.

The experiments consisted of simulating a scenario with a single fine grid, then making the same simulation, but this time using hierarchical grids. We tested the hierarchical grids method with different size ratios, while examining the frame rates of the simulation.

Ratio	Sub-grid	Main grid	FPS
1/2	256×256	256×256	19
1/2	256×256	128×128	28
1/4	128×128	256×256	32
1/4	128×128	128×128	60

Table 1: *Performance results.*

The observed results are shown in Table 1. The columns of the table represent, from left to right, (i) the size ratio of the main grid and the sub-grid, (ii) the resolution of the sub-grid, which directly arises from the size ratio, (iii) the resolution of the main grid, and (iv) the observed frame rate.

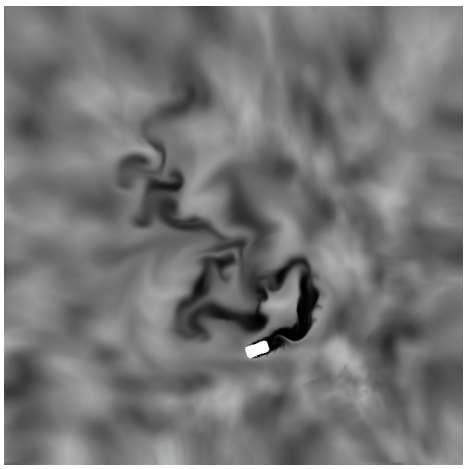

Figure 2: *Test simulation using* 256×256 *resolution for both grids and size ratio of* $1 : 2$.

The position and orientation of the sub-grid during simulation is demonstrated in Figure 2. The area of the image marked by the white square is simulated using the sub-grid.

6. Future work

The algorithms presented in this paper are defined for two dimensional flows but their main ideas are independent from this constraint.

Simulation of three dimensional flows is of much interest nowadays and due to its high computational cost simplification methods are worth being researched.

Utilizing this technique for three dimensional flows would provide significantly more performance gain than for two dimensional flows. Providing means to reduce the required grid resolution in two dimensions would reduce the solution time and space requirements by the second power, but doing likewise in three dimensions would reduce by the third power.

For today's hardware this would not only make a difference on the highest achievable speed and resolution, but it would determine the usability of three dimensional flow in applications.

7. Acknowledgements

This work has been supported by the National Office for Research and Technology (Hungary), by OTKA-71992, and by the Croatian-Hungarian Action Fund.

References

[FSJ01] FEDKIW R., STAM J., JENSEN H. W.: Visual simulation of smoke. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), ACM Press, pp. 15–22.

[GDN98] GRIEBEL M., DORNSEIFER T., NEUNHOEFFER T.: *Numerical simulation in fluid dynamics: a practical introduction.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1998.

[Har05] HARRIS M.: Fast fluid dynamics simulation on the gpu. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Courses* (New York, NY, USA, 2005), ACM Press, p. 220.

[Sta99] STAM J.: Stable fluids. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1999), ACM Press/Addison-Wesley Publishing Co., pp. 121–128.

© The Eurographics Association 2008.