Simulating clouds

Peter Asplund
Medieteknik, Linköping University

December 7, 2009

Abstract

In this report, I describe the methods I have used to add temperature information on top of Jos Stam's source code for real-time fluid dynamics. I also implemented a function called *vorticity confinement* which adds extra vorticity in places where numerical dissipation damps fine-scale motions. The extensions I have made on top of Stam's work is done by Harris[2] et al.

1 Introduction

When presented with this assignment, I had a hard time choosing what to do. Procedural clouds felt like an interesting area, creating something that looked like clouds with different mathematical methods. But when I read more about it, I delved even deeper into the cloud generation treasure, finding articles about cloud simulation in real time. This was an even more exciting area, with math that handled fluid and smoke simulations together with light calculations, all in real time. This was also my downfall, after a week of research and reading I realised I was in way over my head. The computational theory needed to turn complex intergrals and differential equations into C++-code was beyond what I could teach my self in a one week project. But finally I managed to find Jos Stam's paper on Real-time fluid dynamics in games [5] together with source code, something that could save part of my project. I have used all of his code, and tried to extend it into handling temperature, and also to add extra convection where it usually dissipates.

2 Previous work

It would be way to hard to describe Stam's work in this report, and I therefore refer to his paper instead of repeating the same information here. To try to sum it up in a paragraph: Stam presents a simple solution to the Navier-Stokes equations, one that - even though it does not have the strict physical accary needed in research of fluid flow - lets us quickly solve the equations of fluid flow with the visual quality needed for games. The equation that Stam solves are

$$\frac{\partial \boldsymbol{u}}{\partial t} = -(\boldsymbol{u} \cdot \nabla)\boldsymbol{u} + v\nabla^2\boldsymbol{u} + \boldsymbol{f} \tag{1}$$

$$\frac{\partial \rho}{\partial t} = -(\boldsymbol{u} \cdot \nabla)\rho + \kappa\nabla^2\rho + \mathcal{S} \tag{2}$$

Basically, a fluid is described by modeling a velocity field: a function that assigns a velocity vector to every point in space. This can then be affected by various forced, for instance heat rising, an object moving through it, etc. The Navier-Stoken equations are a precise description of this evolution of the velocity field over time. Given the state of the velocity and a current set of forces, the equations tell us exactly how the velocity will change over an infinitesimal time step.

A velocity field on it's own is not really visually interesting though. When we discuss clouds, the field will be visible by moving condensed water, but this could also be smoke or dust particles. But instead of modeling the speed of every water or smoke particle, we represent the particles as *densities*, allowing us to render it quickly in OpenGL. The rendering also automatically adds an interpolation, creating a visually softer result.

3 Adding functionality

Stam's paper and implementation handles smoke, which is first deployed in specific places, and then affected by a force created by the user. The user is then able to also add more smoke after the forces have been applied. Harris et al adds a lot of physics behind their calculation of cloud dynamics, and their behaviour. I am unable to add (or even understand) all of their calculations that will affect the behaviour of the particles (densities), but I have tried to narrow their paper down to two things that I think I am able to add to Stam's implementation. These are as mentioned earlier vorticity confinement and buoyancy.

Vorticity Confinement Vorticity confinement is a method of adding force where it is usually damped out because of numerical dissipation. Clouds, as well as smoke, contains rotational flows at a variety of scales, and vorticity confinement is used to add back the fine-scale motions that are dampened first. It works by first computing the vorticity $\omega = \nabla \times \boldsymbol{u}$, from which a normalized vector field

$$N = \frac{\eta}{|\eta|}, \quad \text{where} \quad \eta = \nabla|\omega| \tag{3}$$

is computed. The vector N point from areas of lower vorticity to areas of higher vorticity. From these vectors we compute the force that can be used to replace dissipated vorticity back in:

$$f_{vc} = \varepsilon h(N \times \omega). \tag{4}$$

Here ε is a user-controlled scale parameter and h is the grid scale.

Buoyancy Since the only source of temperature we have is the added water vapor, we use the density array to represent temperature as well. This temperature will be affected by the same convection and vorticity that the density. My hope was that this would show in the simulation, allowing newly added vapor to be hot and rush up towards the ceiling, while older vapor starts to spread out and slowly fall down.

The work that [2] has done on buoyancy involves many steps in computing the upwards lift that will act as a force upon the smoke/vapor, and I have not been able to fully understand all the steps in it. Therefore I chose to implement a simpler version which act as the smoke density technique originally presented by Fedkiw et al., [1]. In addition to calculating the velocity and pressure fields, a smoke simulation must maintain scalar fields for smoke density, d, and temperature T. The buoyant force is modified to account for the gravitational pull on light water vapor:

$$f_{buoy} = (-\kappa d + \sigma(T - T_0))\hat{k} \tag{5}$$

where κ is a constant mass scale factor and σ is a constant scale factor, T_0 is the ambient temperature and $\hat{k}$ represents the vertical direction. This allows for the smoke to rise as if it is warmer that its surrounding atmosphere.

4 Future work

There are several parts which I am not fully content with, in regards to what I was hoping to achieve when I started. I do understand that I can't expect

to implement Harris paper in a week's project, since it actually resulted in his dissertation paper [3], weighing in at 173 pages. The paper I started reading was a lot less extensive though, which gave me the impression that it could be done with some extra work.

I failed to implement the atmospheric parts of the paper (Rogers and Yau [4]), like pressure, potential temperature, mixing ratio (of hydrometeors) and thermodynamic equations. This is what actually makes the clouds behave like clouds, whereas they now more appear as light smoke.

References

[1] R. Fedkiw, J. Stam, and H. W. Jensen. Visual simulation of smoke. In *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 15–22, aug 2001.

[2] M. Harris, W. Baxter, T. Scheuermann, and A. Lastra. Simulation of cloud dynamics on graphics hardware. In *Proc. Graphics Hardware*, 2003.

[3] M. J. Harris. *Real-time cloud simulation and rendering*. PhD thesis, 2003. Director-Lastra, Anselmo.

[4] R. R. Rogers and M. K. Yau. *A short course in cloud physics / by R.R. Rogers and M.K. Yau*. Pergamon Press, Oxford ; New York :, 3rd ed. edition, 1989.

[5] J. Stam. Real-time fluid dynamics for games. In *Proceedings of the Game Developer Conference*, Toronto, Ontario, Canada, 2003.