LINKÖPINGS UNIVERSITET
Institutionen för Teknik och Naturvetenskap

TNM085 Modelleringsprojekt

Fluid Simulation

March 14, 2011

Robert Novo, robno767@student.liu.se
Martin Person, marpe357@student.liu.se
Mattias Persson, matpe621@student.liu.se
Johannes Ullström, johul223@student.liu.se

Examiner
Anna Lombardi

Abstract

The aim of this project was to create a real-time interactive fluid simulation. The fluid simulation utilizes SPH, which is a computional method used to simulate fluids. The simulation application was developed using the programming language C# and the environment Microsoft XNA for the visualization. Keywords: simulation, fluids, Navier-Stokes, SPH, marching cubes, C#, XNA.

Contents

List of Figures

1 Introduction

Simulation of different physical phenomena is increasingly used within several fields including various scientific researches, software and games developement, and the movie industry. Consequently, we found that the simulation of fluids would be an exciting area to explore. The aim of this project is to program an interactive application which simulates fluids in 3D.

1.1 Background

The description and simulation of different physical phenomena has always been an important part of science studies and a fundament in physics. Fluids are no exception and numerous approaches to visualise fluids has been presented. Two frequently used methods are grid-based and particle-based visualisation. The use of computer calculations and computer graphics has revolutionized the prospects of Computational Fluid Dynamics (CFD). This project employs the particle-based method of CFD which is based on the Navier-Stokes equations.

1.2 Usage

1.2.1 Computer games

In todays' computer games the player is likely to encounter some sort of fluid being simulated, whether it be water, smoke, among a few. In order to improve rendering speed and visual quality of the simulation, some physical details are sacrificed. To interact with fluids in games the simulation is usually done by graphic card calculations, because of its large amount of cores compared to a processor (CPU).

1.2.2 Movies

On the contrary to computer games, where the focus lies in rendering speed, the most important factor of fluid simulations in movies is its visual appearance. Real-time rendering is not necessary in movies, thus enabling heavier calculations, should that be required for the visual appearance.

2 Metoder

Nedan följer en beskrivning av de metoder som har använts i projektet.

2.1 Navier-Stokes ekvationer

En välanvänd metod för att beskriva fluider är Navier-Stokes ekvationer, som är framtagna av Claude-Louis Navier och George Gabriel Stokes. Navier-Stokes tillämpar Newtons andra lag för att beskriva hur stabila fluider flödar och genererar en serie av differentialekvationer.

Navier-Stokes ekvationer gör det möjligt att illustrera hur storheter som kraft, densitet och viskositet påverkar fluiden.

$$\rho\left(\frac{\partial v}{\partial t} + v \cdot \nabla v\right) = -\nabla p + \rho g + \mu\nabla^2 v \tag{1}$$

Ekvationen förenklas enligt Müller03:

$$\mathbf{a}_i = \frac{d\mathbf{v}_i}{dt} = \frac{-\nabla p_i + \rho_i \mathbf{g} + \mu\nabla^2 \mathbf{v}_i}{\rho_i} \tag{2}$$

där $-\nabla p_i$ är kraften från tryck, $\rho_i \mathbf{g}$ är yttre krafter såsom gravitation och $\mu\nabla^2 \mathbf{v}_i$ är viskositetskraften.

2.2 SPH, Smoothed Particle Hydrodynamics

Smoothed Particle Hydrodynamics (SPH) utvecklades av Lucy och Gingold och är till början en metod för att simulera astrofysiska problem, men lämpar sig även till att simulera vätskor. I en artikel skriven av Müller, Charypar and Gross (2003) så utvecklas denna metod ytterligare. Müller med andra kommer fram till att för att bestämma hur varje partikel rör sig är det nödvändigt att inkludera en "smoothing kernel"-metod, som regulerar stabiliteten, noggrannheten och hastigheten av SPH. Enligt SPH är en skalärkvantitet *A* interpolerad vid plats *r* med en vägd summa av bidrag från alla partiklar.

$$A_S(\mathbf{r}) = \sum_j m_j \frac{A_j}{\rho_j} W(\mathbf{r} - \mathbf{r}_j, h) \tag{3}$$

där m_j är massan och ρ_j är densiteten av partikel j. Funktionen W är en viktfunktion kallad "smooth kernel".

2.3 Particles

2.4 Marching Cubes

Marching Cubes är en algoritm utveckld av Cline och Lorensen år 1987, som smälter samman partiklar för att skapa en yta, en så kallad "mesh". Marhing Cubes använder 256 kubinställningar för att representera alla möjliga kombinationer som en mesh kan korsa kuben. Genom att utnyttja symmetri är det möjligt att reducera kombinationerna till 15 unika mönster.

Marching cubes is an algorithm developed by Cline and Lorensen in 1987, which "melts" particles together to construct a surface, or so called "mesh". Marching cubes uses 256 cube configurations to represent all possible variations that a mesh can cross the cube. By utilizing symmetry it is possible to reduce the variations to 15 unique patterns.

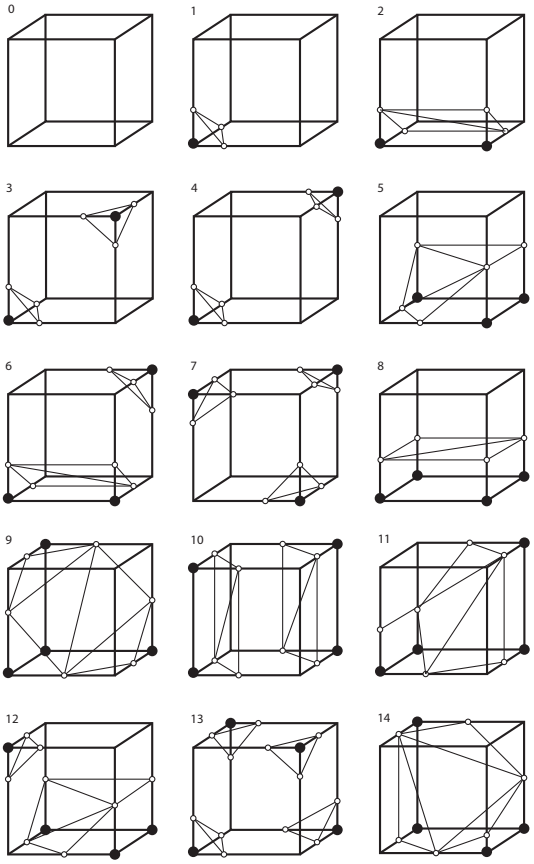

Figure 1: Marching cubes

2.5 C#/.Net och XNA

Den exekverbara filen för det här projektet skrevs i programmspråket C# och ramverket Microsoft .NET. Microsoft XNA används för att underlätta det grafiska utseendet.

3 System description

HÄR BEHÖVS TEXT

3.1 System

The system built in this project is a fluid represented by particles. Different forces act on the particles which causes them to move the way they do. Fluids are materials that are being deformed under forces and are commonly known to be liquids, yet fluids are also gases. Our model is capable simulating a gas like substance, however, the focus was mainly on creating a liquid. Normally, a simulation system is described with a bond graph or block diagram but this is difficult in our case, because we are creating a particle based fluid simulation, which requires quite a lot of particles to look like liquid.

3.2 Physical history

One method to describe the behavior of fluids is the Navier-Stokes equations, which is used for this project. As seen in equation 1, the physical quantities used are pressure, density and viscosity.

3.3 Density

The first step in the simulation is to compute the particle densities. By substituting A in equation x with the particle density ρ, we get:

$$\rho_S(\mathbf{r}) = \sum_j m_j W(\mathbf{r} - \mathbf{r}_j, h) \tag{4}$$

The smoothing kernel used for calculating the density in this simulation was the W_{poly6} kernel, designed by Müller03.

$$\rho_S(\mathbf{r}) = \sum_j m_j W(\mathbf{r} - \mathbf{r}_j, h) \tag{5}$$

3.4 Pressure

The pressure is calculated using the forumla described below and is a simplified version of the original SPH formula. [1]

$$p = k(\rho - \rho_0) \tag{6}$$

where ρ is the density of the current particle, ρ_0 is the rest density, and k is the gas constant of the current substance being simulated, which depends on the temperature. The higher the temperature, the lighter the substance.

3.5 Viscosity

Viscosity is caused by friction which converts the particle's kinetic energy into heat. The viscosity is calculated by looking at the force in the gradient of the neighbors of each particle.

3.6 Force simulation

The sum of all forces that act on the particles is used for the final calculation which determines how the particles movements repectively. Since the particle mass is known the acceleration is calculated using Newtons 2^{nd} law. The current acceleration is multiplied with a predefined timestep and added to the current velocity. This new velocity is multipled with the same predefined timestep, which gives the movement and therefore the new position of the particle.

4 Implementation

As mentioned in the method section, the code for this simulation was written in C# using Microsoft Visual Studion Pro 2010.

4.1 Neighbors and Forces

The main functions of the software is to calculate the forces which affect the particles. The amount of forces acting on each particle depends on the surrounding particles, and therefore an important step is to find the neighbors of each particle. A particle *i* is considered as neighbour to a particle *j* if the distance betweem them is less than a preset maximum distance. The distance between neighbours found decides how they will affect eachother.

4.2 Collision Handling

The collision handling performed in this simulation is fairly simple. A bound is defined in which the fluid can exist. After a particle's new position is calculated, the program checks if it is within the acceptable bounds or not. If the particle's position is out of bounds, the velocity of this particle is inverted and its new position is set to the position on the bounds where it probably would have been before it went out of bounds, otherwise nothing changes.

4.3 Changing Simulation Behaviour

To make the simulation interactive, sliders were added to the simulation software GUI, which changes the values of constants used in the equations calculating the forces of the simulation.

4.4 Rendering

The simulation is rendered using the benefits of XNA. A "camera" is created to capture the visual implementation of the simulation. XNA Studio already provides with shaders which really adds to the visual appearance without being coded manually.

5 Results

The result of this project is an executable .NET-application. This means it can be run on any computer as long as that computer has the .NET framework installed, along with XNA Studio. Screenshots from the resulting application of this project is presented the appendix.

Figure 2 demonstrates the general appearance of the program. In the middle there are a number of blue particles sealed in a transparent cube. (An orange grid is passing under the cube). In the top left the user will find statistics from the calculation of

the program as well as tips on how to control the program. In the top right there are six different sliders and one button. Figure 3 shows a close-up of the top left of the program. The label FPS stands for frames per second and is a measurement of how many frames per second the CPU manages to calculate. A higher FPS-total will give a more fluent appearance then a low. Performance of the application may vary as different computers have different specifications and as a result get a higher or lower FPS when the particles are in motion. StopWatch. The label named Particles shows the number of particles. Camera Position has the three variables X, Y and Z, measuring the cameras position from the centre of the cube. The user can control the camera using the arrows on the keyboard. The user can also toggle to fullscreen, pressing <F6> as well as toggle grid on and off, pressing <tab>. Pressing <shift> will toggle marching cubes on and off. Figure 4 shows a close-up of the top right. The user can drag the three first sliders (from the top) to control the fluid parameters. Particles control the number of particles. Time step controls the speed of the simulation. Dragging the Rotation speed slider will rotate the cube clockwise or counter clockwise. There is also a Restart-button that will restart the simulation. Pressing restart will not restore the changes the user have done on any of the sliders or keyboard commands.

6 Conclusion

- Producing realistic simulation of fluids is very difficult. It is also hard to define what a "good" simulation is. There are several other simulations, often published as short demonstration films that are pre-rendered and not interactive. An example of that is the result of Beaudoin, Clavet and Poulin (2005). Beaudoin et al produces stunning visuals and the clip has gotten 23 956 views on youtube.com (2011-03-09) which certainly could be rubricated as "good". With that in mind we think that our own result is good. We have succeeded in reaching the aim of our project.

What could have been done differently? - One could consider if the same, or better, result could be accomplished if another environment had been chosen as a platform for the project. For example if we have had chosen OpenGL instead of XNA. It was also possible to have used other methods to reach our goal. One of the most important reasons why we chose to use SPH was that there were quite a lot of resources available on the subject. At one point we were considering using point splatting (Müller at alt, 2003) instead of marching cubes. But we quickly realized that point splatting generally uses 10.000 to 100.000 particles, which would have demanded significantly more CPU-power, since we only used up to 2000 particles in our system. If we wanted to continue developing the software, what would we have done? - In an ongoing development of the project we would concentrate our efforts to implement the NVIDIA engine CUDA, Compute Unified Device Architecture, to the project. CUDA uses the inbuilt graphics card of the computer to calculate simple computations and unburden the CPU. Doing this will improve the "speed" of the simulation, i.e. the number of frames/second. We would also do more collision handling. We could for example have a static object in the glass cube that the fluid would have to interact with.

Rewritten!

As a conclusion, producing realistic fluid simulations is difficult. It is also hard to define what is considered a correct simulation. Our refence point as to how a good fluid simulation should look like is the simulation done by Beaudoin, Clavet and Poulin in 2005. Moreover, we could have taken other approaches to achieve different results,

better or worse. We used XNA as a platform for our project, however there are other options that we could have chosen, OpenGL for instance.

7 References

References

[1] Müller M., Charypar D., Gross M. *Particle based fluid simulation for interactive applications* 2003.

[2] Lucy L. B. *A numerical approach to the testing of the fission hypothesis.* The Astronomical Journal, 82:1013-1024, 1977. http://adsabs.harvard.edu/full/1977AJ.....82.1013L

[3] Gingold R. A., Monaghan J. J. *Smoothed Particle hydrodynamics: theory and application t non-spherical stars.* Monthly Notices of the Royal Astronomical Society, 181:375-389, 1977. http://adsabs.harvard.edu/full/1977MNRAS.181..375G(2011-03-09)

[4] Cline H. E., Lorensen W.E. *Marching Cubes: A high resolution 3D surface construction algorithm.* SIGGRAPH, 1987. http://kucg.korea.ac.kr/seminar/2001/src/PA-01-16.pdf(2011-03-09)

[5] Beaudoin P., Clavet S., Poulin P. *Particle-based Viscoelastic Fluid Simulation.* http://www.iro.umontreal.ca/labs/infographie/papers/Clavet-2005-PVFS/pvfs.pdf(2011-01)

A Screen captures