Real Time Fluid Simulation

using Smoothed-Particle Hydrodynamics and OpenGL

Computer Graphics CS 488

Matthias Untergassmair
munter2@uic.edu

Michael Berg
mberg4@uic.edu

ABSTRACT

This paper explores the field of Smooth Particle Hydrodynamics (SPH), starting at its beginnings as a tool to simulate astrophysical phenomena and following its evolution and implementation as a way to simulate fluids such as water. We will then give a brief mathematical background and ensuing algorithm of our SPH simulation followed by a detailed explanation of it.

Author Keywords

SPH, Smoothed Particle Hydrodynamics, OpenGL, Fluid Simulation, Real Time

Download code at

`https://github.com/munter2/RealTimeFluid`

INTRODUCTION

Smoothed Particle Hydrodynamics (SPH) successfully simulates fluids by breaking up a fluid body into individual parts, or particles. These particles together form a particle system that simulates various gravitational forces. Fluid movement is simulated in the system by moving particles around any particle moved, simulating a rippling, wave effect.

The ripple effect is created by first calculating which particles are surrounding a particle moved, and these surrounding particles are moved according to the movement of the first particle moved. But SPH wasn't originally intended to simulate liquid substances, but astrophysical phenomena.

HISTORY

Given all the different applications for Smoothed Particle Hydrodynamics (SPH), it was first used to simulate interstellar phenomena. Conceived in 1977 by Gingold and Monaghan was an improvement to the Standard Finite Difference Method, which until their breakthrough, was the method to use to simulate astrophysical phenomena. They improved on this method by making "use of Lagrangian description of fluid flow which automatically focuses attention on fluid elements" [3]. In this implementation, particles "move according to the Newtonian equations with forces due to the pressure gradient and other body forces: gravity, rotation and magnetic" [3].

The set of equations utilized in Gingold and Monaghan's SPH implementation is a density distribution calculated at each point

$$\rho_s(r) = \int W(r - r')\rho(r')dr' \tag{1}$$

where W is a function satisfying the condition

$$\int W(r)dr = 1 \tag{2}$$

This density distribution equation is used to calculate the density of the particles surrounding the current particle the equation it's being applied to. A Gaussian function is used for the kernel function, with the form

$$(\frac{1}{\pi h^2})^{\frac{3}{2}} \exp{(\frac{-r^2}{h^2})} \tag{3}$$

and h is

$$h = b(< r^2 > - < r >^2)^{\frac{1}{2}} \tag{4}$$

and *b* is adjustable. The gravitational potential of the particles is denoted by

$$\phi = -G \int \frac{\rho_N(r')}{|r - r'|} dr' \tag{5}$$

with G being the gravitational constant. Substituting in eq. 1 turns the equation of gravitational potential into

$$\phi = -\frac{GM}{N} \sum_{j=1}^{N} \int \frac{W(r - r')}{|r - r'|} dr' \tag{6}$$

To finish the equation, starting with

$$\nabla^2 I_j = -4\pi W(r - r_j) \tag{7}$$

substituting gravitational potential for I

$$\nabla\rho = -\frac{GM}{N} \sum_{j=1}^{N} \left\{ -\frac{4\pi}{u_j^2} \int_0^{u_j} W(u)u^2 du \right\} \nabla u_j \tag{8}$$

where

$$u_j = r - r_j \tag{9}$$

Adding in the Gaussian Function in eq. 3 for W yields

$$\nabla\phi = -\frac{GM}{N} \sum_{j=1}^{N} \frac{2}{u_j} \left(\frac{f}{\pi}\right)^{\frac{1}{2}} \left[\exp{(-fu_j^2)} - \frac{1}{u_j} \int_0^{u_j} \exp{(-fu^2)} du \right] \nabla \tag{10}$$

This equation is the full solution to calculating the gravitational potential of all particles involved in the simulation.

The result of Gingold and Monaghan's theory and implementation was a robust and extendable idea that could easily made more accurate "by increasing the number of particles and by using the devices known to improve Monte Carlo integration methods" [3].

SMOOTHED PARTICLE HYDRODYNAMICS OVERVIEW

Smoothed-particle hydrodynamics (SPH) method works by dividing the fluid into a set of referred to as particles. These particles have a spatial distance, or smoothing length, over which their properties are "smoothed" by a kernel function, such as a Gaussian function

$$(\frac{1}{\pi h^2})^{\frac{3}{2}} \exp{(\frac{-r^2}{h^2})} \tag{11}$$

This means that the physical quantity of any particle can be obtained by summing the relevant properties of all the particles which lie within the range of the kernel. For example, using Monaghan's popular cubic spline kernel the temperature at position r depends on the temperatures of all the particles within a radial distance 2h of r [5].

The contributions of each particle to a property are weighted according to their distance from the particle of interest, and their density A property for any one particle is calculated by discerning how many particles lay within smoothing length of a given particle and what their respective properties are, and how dense that smoothing length is around the particle of interest. Mathematically, this is governed by a kernel function, like the Gaussian function given in eq. 1. This Gaussian function gives large numbers for particles whose smoothing lengths are close to the particle of interest, and thus effect the calculation of a given particles properties, but as particles get far away, and their smoothing length becomes extremely small, although never reaches 0. This is due to intrinsic properties of the Gaussian function itself, but it is easy to account for this problem by testing for extremely small values close to zero, and disregarding those particles that are far away from the particle of interest as a result.

The equation for any quantity A at any point r is given by

the equation

$$A(r) = \sum_j m_j \frac{a_j}{p_j} W(|r - r_j|, h) \tag{12}$$

where m is the mass of particle j, A is the value of the quantity A for particle j, r denotes the position, p is the particle's density and W is a kernel function, such as the Gaussian function used by Gingold and Monaghan shown in eq. 1. The density of the particle can be computed in a variety of ways, such as the example equation

$$p_i = p(r_i) = \sum_j m_j \frac{p_j}{p_j} W(|r_i - r_i|, h) = \sum_j m_j W(r_i - r_j, h) \tag{13}$$

Similarly, the spatial derivative of a quantity can be computed as follows

$$\nabla A(r) = \sum_j m_j \frac{a}{p_j} \nabla W(|r - r_j|, h) \tag{14}$$

Although the size of the smoothing length can be fixed in both space and time, this does not take advantage of the full power of SPH. By assigning each particle its own smoothing length and allowing it to vary with time, the resolution of a simulation can be made to automatically adapt itself depending on local conditions. For example, in a very dense region where many particles are close together the smoothing length can be made relatively short, yielding high spatial resolution. Conversely, in low-density regions where individual particles are far apart and the resolution is low, the smoothing length can be increased, optimising the computation for the regions of interest. Combined with an equation of state and an integrator, SPH can simulate hydrodynamic flows efficiently [5].

Given the principles of smoothing length, calculating properties based on the area around a particle, and increasing or decreasing a property given the density of particles in an area, we applied some of these concepts in our implementation of SPH.

MATHEMATICAL BACKGROUND

$$a_i^n = \frac{F_i^n}{m_i} = \dots \tag{15}$$

THE ALGORITHM

In the following, we denote the position for the particle i at time t as x_i^t, its velocity as v_i^t and its acceleration as a_i^t. We omit the vector notation $(\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{a})$ for these quantities, since the following equations are valid for the vectors as well as for each component individually.

References

1. Akenine-Möller, T., Haines, E., and Hoffman, N. Real-time rendering 3rd edition. Natick, MA, USA: A. K. Peters, Ltd., 2008, 1045. ISBN: 987-1-56881-424-7.
2. Erleben, K. Smoothed particle hydrodynamics - a short introduction to principles and ideas. *University of Copenhagen*, 2010:
3. R.A. Gingold, J. M. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Mon. Not. R. Astron. Soc.*", (375-389), 1977:
4. S. Adami X. H., N. A. A generalized wall boundary condition for smoothed particle hydrodynamics. *Journal of Computational Physics*, (231), 2012:
5. Smoothed-particle hydrodynamics. **http://en.wikipedia.org/wiki/Smoothed_particle_hydrodynamics**. Accessed: 2014-12-09.

Several resources point out the algorithms to be used for applying SPH to Fluid Dynamics, the algorithm is described particularly clearly in [2] and can be summarized as follows:

Algorithm 1: SPH simulation

for $i = 1, \ldots, N_{Particles}$ **do**

Compute the density of the ith particle

$$\rho_i = \sum_j m_j W(\|x_i - x_j\|, h)$$

Compute the pressure of the ith particle

$$p_i = k(\rho_i - \rho_0)$$

Compute the interaction forces acting on the ith particle, consisting of pressure forces, viscosity and gravity

$$f_i^p = -\sum_j m_j \frac{p_i + p_j}{2\rho_j} \nabla W(\|x_i - x_j\|, h)$$

$$f_i^v = \mu \sum_j m_j \frac{v_j + v_i}{\rho_j} \nabla^2 W(\|x_i - x_j\|, h)$$

$$f_i^{\text{total}} = f_i^p + f_i^v + g$$

Solve the differential equation

$$\begin{aligned} \dot{x}_i &= v_i \\ \dot{v}_i &= \frac{f_i}{\rho_i} \end{aligned} \tag{16}$$

In our implementation, we use the Velocity Verlet (or Leap-Frog) scheme to solve the ODE 16, as suggested in [4]. The scheme looks as follows:

Algorithm 2: Single Timestep with Velocity Verlet Algorithm

Data: x_i^t, $v_i^{t-\frac{\Delta t}{2}}$, a_i^t, Δt

Result: $x_i^{t+\Delta t}$, $v_i^{t+\frac{\Delta t}{2}}$, $a_i^{t+\Delta t}$

$v_i^{t+\frac{\Delta t}{2}} = v_i^{t-\frac{\Delta t}{2}} + \Delta t a_i^t$;

$x_i^{t+\Delta t} = x_i^t + \Delta t v_i^{t+\frac{\Delta t}{2}}$;

$a_i^{t+\Delta t} = a_i^{t+\Delta t}(x_i^{t+\Delta t}, m_i)$ from equation 15 ;

As a kernel function we used the standard kernel, the *cubic spline*

$$W_3(r,h) := \frac{1}{\pi h^3}\begin{cases} 1-\frac{3}{2}\xi^2+\frac{3}{4}\xi^3, & \text{for } 0\le\xi<1 \\ \frac{1}{4}(2-\xi)^3, & \text{for } 1\le\xi<2 \\ 0, & \text{otherwise} \end{cases}$$

where $\xi := \frac{r}{h}$. It can be seen clearly that the support of this function is $2h$.
The derivative is

$$\nabla W_3(r,h) = \frac{\partial}{\partial r}W_3(r,h) + \frac{\partial}{\partial h}W_3(r,h)$$

- for $0 \le \xi < 1$

$$\frac{\partial}{\partial r}W_3(r,h)|_{0\le\xi<1} = \frac{1}{\pi h^3}\frac{1}{h}\left(3\xi+\frac{9}{4}\xi^2\right)$$

$$\begin{aligned}\frac{\partial}{\partial h}W_3(r,h)|_{0\le\xi<1} &= \frac{-3}{\pi h^4}\left(1-\frac{3}{2}\xi^2+\frac{3}{4}\xi^3\right) \\ &\quad+\frac{1}{\pi h^3}\frac{1}{r}\left((-2)\frac{3}{2}\xi^3+(-3)\frac{3}{4}\xi^4\right) \\ &= -\frac{3}{\pi h^4}\left(1-\frac{3}{2}\xi^2+\frac{3}{4}\xi^3\right) \\ &\quad+\frac{1}{\pi h^3}\frac{1}{r}\left(-3\xi^3-\frac{9}{4}\xi^4\right)\end{aligned}$$

$$\begin{aligned}\Rightarrow \nabla W_3(r,h)|_{0\le\xi<1} &= \frac{1}{\pi h^4}\left(3\xi+\frac{9}{4}\xi^2\right) \\ &\quad-\frac{3}{\pi h^4}\left(1-\frac{3}{2}\xi^2+\frac{3}{4}\xi^3\right) \\ &\quad+\frac{1}{\pi h^3}\frac{1}{r}\left(-3\xi^3-\frac{9}{4}\xi^4\right) \\ &= \frac{3}{\pi h^4}\left(-1+\xi+\frac{9}{4}\xi^2-\frac{3}{4}\xi^3\right) \\ &\quad-\frac{1}{\pi r h^3}\xi^3\left(3+\frac{9}{4}\xi\right)\end{aligned}$$

- for $1 \le \xi < 2$

$$\begin{aligned}\frac{\partial}{\partial r}W_3(r,h)|_{1\le\xi<2} &= \frac{1}{4}3(2-\xi)^2\cdot\left(-\frac{1}{h}\right) \\ &= -\frac{3}{4h}(2-\xi)^2\end{aligned}$$

$$\begin{aligned}\frac{\partial}{\partial h}W_3(r,h)|_{1\le\xi<2} &= \frac{1}{4}3(2-\xi)^2\cdot\left(+\frac{\xi}{h}\right) \\ &= \frac{3\xi}{4h}(2-\xi)^2\end{aligned}$$

$$\begin{aligned}\Rightarrow \nabla W_3(r,h)|_{1\le\xi<2} &= -\frac{3}{4h}(2-\xi)^2+\frac{3\xi}{4h}(2-\xi)^2 \\ &= \frac{3}{4h}(2-\xi)^2(\xi-1)\end{aligned}$$

- otherwise

$$\nabla W_3(r,h)|_{\text{otherwise}} = 0$$

The following two code snippets show how these formulas are implemented in our simulation:

Listing 1: Cubic Spline

```
// Cubic spline function
inline float W3(float r, float h) {
  float W = 0;
  float zeta = r/h;
  if(0 <= zeta && zeta <= 2) {
    W = (zeta < 1 ?
        1 - .75*zeta*zeta*(2+zeta) :
        .25*(2-zeta)*(2-zeta)*(2-zeta)
    );
  }
  W /= (M_PI*h*h*h);
  return W;
}
```

Listing 2: Derivative of Cubic Spline

```
Hello World
```