

# Particle-based Realtime Fluid Dynamics

Florian Reiterer 0527025\*

Institute of Computer Graphics and Algorithms, TU Vienna, Austria

## Abstract

This paper discusses particle-based approaches in fluid simulation with a special focus on realtime simulation. After a short introduction to Navier-Stokes equations and particle systems in general some simple fluid simulation approaches are presented. The more accurate Smoothed Particle Hydrodynamics (SPH) technique is explained and applied to the Navier-Stokes equations. Optimizations for improved speed are discussed, as well as alternative methods (MPS-MAFL). Different rendering methods are compared and finally a short outlook on fluid simulation in (future) games is made.

**Keywords:** fluid dynamics, simulation, particles, Smoothed Particle Hydrodynamics

## 1 Introduction

The movement of fluids (gases and liquids) is an important field of research. Computational fluid dynamics (CFD) is one of the branches of fluid mechanics that uses numerical methods and algorithms to solve and analyze problems that involve fluid behavior. CFD is used in different areas, e.g. in engineering (aerodynamics, hydrodynamics), in meteorology or river regulation. In the last decades it has found his way into computer graphics for realistic animation and visualization of fluids.

### 1.1 Navier-Stokes equations

The motion of fluids can be described by a set of equations called Navier-Stokes equations, named after Claude-Louis Navier and George Gabriel Stokes.

The Navier-Stokes equations exist in many different formulations. The most general form is:

$$\rho \left( \frac{\partial v}{\partial t} + v \cdot \nabla v \right) = -\nabla p + \nabla T + f \quad (1)$$

with  $\rho$  being the density,  $v$  being the velocity,  $t$  being time,  $p$  being pressure,  $\nabla T$  being shear stress and  $f$  being other forces. This is a statement of the conservation of momentum in a fluid. For an incompressible Newtonian fluid the equation is:

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

\*e-mail: me@florianreiterer.com

$\mu$  is the viscosity. The meaning of the single terms is:

$$\rho \left( \underbrace{\frac{\partial v}{\partial t}}_{\text{unsteady acceleration}} + \underbrace{v \cdot \nabla v}_{\text{convective acceleration}} \right) = \underbrace{-\nabla p}_{\text{pressure gradient}} + \underbrace{\mu \nabla^2 v}_{\text{viscosity}} + \underbrace{f}_{\text{other forces}} \quad (3)$$

In addition, conservation of mass is necessary. In general form:

$$\frac{\partial \rho}{\partial t} + \nabla(\rho v) = 0 \quad (4)$$

For incompressible fluids the density is constant, therefore the equation simplifies to:

$$\nabla v = 0 \quad (5)$$

### 1.2 Fluid simulation

There exist two basic types of fluid simulations:

- **Grid-based or Eulerian** approaches solve the fluid equations at fixed locations in space. For a survey of these methods see [Marek 2007].
- **Particle-based or Lagrangian** approaches solve the fluid equations at locations moving with the fluid, so called particles. This document only covers this kind of algorithms.

## 2 Particle-based methods

Particles were first introduced by [Reeves 1983]. Since then they have been used for a variety of effects, like fire and smoke, water and even cloth and other deformable objects. But what are particles and what makes them so flexible?

### 2.1 Introduction to particles

Particles are objects that have mass, position and velocity and respond to forces, but have no spatial extent. Because of their simplicity, large numbers of them can be used to simulate various effects. Their behavior is mainly determined by the forces and constraints applied to them. In contrast to grids (see [Marek 2007]), where the points (samples) are fixed in space, particles move through space.

#### 2.1.1 A simple particle

Particles move according to certain laws. For a physical correct simulation, they follow the laws of classical mechanics (see e.g. [Goldstein 1980]). So each particle needs a number of properties:

- **Mass:** Each particle has a certain mass (if not it is not moved by forces)

- **Position** vector: Each particle has a certain position in space
- **Velocity** vector: The velocity (and its direction) of each particle
- **Force** accumulator: Each particle can be affected by a number of forces. The (vector) sum of them is the force that moves the particle.

The connection between those properties is made by Newton's second law:

$$F = ma \quad (6)$$

where  $F$  is the applied force,  $m$  the particle's mass and  $a$  its acceleration. We know the particle's mass and the forces applied to it. Its acceleration is:

$$a = \frac{F}{m} \quad (7)$$

The acceleration is the first time derivative of the velocity  $v$ :

$$\dot{v} = \frac{\partial v}{\partial t} = a \quad (8)$$

and the velocity is the time derivative of the position  $x$ :

$$\dot{x} = v \quad (9)$$

therefore

$$\ddot{x} = \frac{F}{m} \quad (10)$$

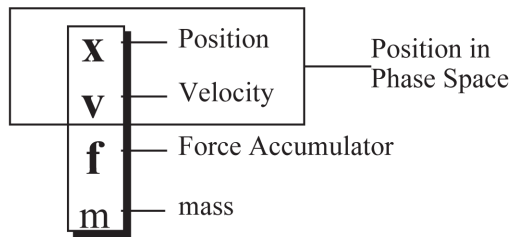


Figure 1: Structure of a simple particle (image: [Witkin and Baraff 1997])

## 2.1.2 The particle system

A particle system is a group of particles which (can) interact with each other. It holds pointers to the single particles and manages the forces acting on them.

## 2.1.3 Phase Space

In order to calculate the position of a particle at a certain time, the second order differential equation  $\ddot{x} = F/m$  must be solved. In order to simplify this often the concept of *phase space* [Goldstein 1980] is used. A phase space is a space in which all possible states of a system are represented, with each possible state of the system corresponding to one unique point in the phase space. The position vector  $x$  and the velocity  $v$  are concatenated to form a 6-vector (in three-dimensional space) in phase space. In components, the phase space equation for motion is:

$$[\dot{x}_1, \dot{x}_2, \dot{x}_3, v_1, v_2, v_3] = [v_1, v_2, v_3, f_1/m, f_2/m, f_3/m] \quad (11)$$

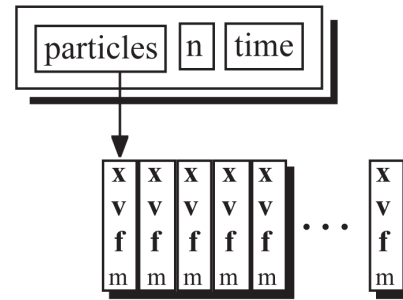


Figure 2: Structure of the particle system (image: [Witkin and Baraff 1997])

This equation can be solved numerically using e.g. Euler's method.

A system of  $n$  particles can be described by  $n$  copies of the equation, concatenated to form a  $6n$ -long vector. This can be regarded as a point moving through  $6n$ -space.

## 2.1.4 Forces

Forces can be grouped in three categories:

- **Unary forces** act independently on each particle. That can be a constant force, or it can be dependent on one or more of the particle's properties, like its position, velocity or mass. It can also change in time. Examples are gravity or wind.
- **$n$ -ary forces** act on a fixed set of particles, like e.g. springs.
- **Forces of spatial interaction** can act on any or all pairs of particles, depending on their position, e.g. attraction or repulsion.

The last category is especially interesting for the simulation of fluid dynamics.

## 2.1.5 Collisions with Rigid Bodies

The detection of collisions is one of the more difficult problems in physical simulation. Here, only a short insight can be given. The collision problem can be divided into two parts: **detecting collisions** and **responding to them**.

The collision detection basically means determining if a point in space (here the particle location) is inside or outside of an object. This is simple for a plane, but can get quite complicated with more complex objects, e.g. models with thousands of polygons.

After a collision has been detected, a proper response has to be made. The right thing to do is to solve for the time of contact, and bring back the whole system to that time. A less accurate, but easier alternative is to displace the point that has collided back to the surface.

Additionally, the particle must react somehow to the collision, e.g. bounce. In order to do this, the velocity and force vectors are partitioned into two orthogonal components, one normal to the surface, and the other parallel to it. In a fully elastic collision the normal

component of the velocity is negated, which means the velocity is mirrored with respect to the surface. In an inelastic collision, the normal velocity component is additionally multiplied by a constant between 0 and 1, called the coefficient of restitution. Furthermore, friction could be simulated.

## 2.2 Uncoupled particle systems

Uncoupled particles have only unary forces applied. They have been used in games for many years, for effects like smoke, explosions, splattering blood, falling rain and others. Because there is no connection between the particles, uncoupled particle systems are not well suited for simulation of substances, where internal forces have a big effect on their behavior. Flowing water or viscous fluids like e.g. honey can not be simulated this way.

But effects where water dissolves to drops, like waterfalls or fountains can be reproduced very well this way. Since the update of each particle's position is very simple and fast, large numbers of particles can be used. This leads to very detailed simulation results, but does not represent a solution to fluid dynamics.



Figure 3: A fountain using uncoupled particles (image: [Andersson 2005])

## 2.3 Simple particle coupling

To simulate the interaction between particles, some kind of interparticle force has to be applied. A simple form would be the Lennard-Jones potential [Lennard-Jones 1931].

The Lennard-Jones potential describes the forces between neutral atoms or molecules. There is a long range attracting force (van der Waals force) and a short range repulsive force (Pauli repulsion, the result of overlapping electron orbitals). The force is zero at a certain ideal distance (where attracting and repulsive forces are in equilibrium) and falls off to zero with increasing distance. The Lennard-Jones potential is a simple mathematical model that represents this behavior:

$$V(r) = 4\varepsilon \left[ \left( \frac{\sigma}{r} \right)^{12} - \left( \frac{\sigma}{r} \right)^6 \right] \quad (12)$$

where  $r$  is the particle distance,  $\varepsilon$  is the depth of the potential well and  $\sigma$  is the (finite) distance at which the interparticle force is zero.

Murta and Miller [Murta and Miller 1999] take this approach to simulate the motion of dripping and splashing fluids. They use the Lennard-Jones potential for the particle-particle interaction and the Euler method with small time steps to integrate the resulting particle accelerations. By defining different particle families having different forces within and between them, they simulate multi-fluid interaction, like e.g. the separation of oil and water, or air bubbles inside water.



Figure 4: Splashing fluid coupled by Lennard-Jones potential (image: [Andersson 2005])

## 2.4 Smoothed Particle Hydrodynamics

Smoothed Particle Hydrodynamics, SPH in short, is another form of coupled particle system. It was invented to simulate astrophysical phenomena such as star formation and galaxy collisions ([Lucy 1977], and [Gingold and Monaghan 1977]). A good introduction to SPH can be found in [Monaghan 1992].

The fluid is made up by a large number of particles with properties such as mass, velocity, and new: density. In SPH the fluid is a continuous field in space. The particles represent samples or interpolation points of this continuous function. The properties of the fluid can be evaluated at any point in space by interpolating between the sample points. This is done by using a radially symmetric weighting function, called interpolation kernel or smoothing kernel. The integral interpolant of a property  $A$  at location  $r$  is defined by:

$$A_I(r) = \int A(r')W(r-r',h)dr' \quad (13)$$

where the integration is over the entire space and  $W$  is the smoothing kernel with core radius  $h$ .  $W$  must be even, meaning

$$W(r,h) = W(-r,h) \quad (14)$$

and normalized, which means

$$\int W(r-r',h)dr' = 1 \quad (15)$$

Since we have a discrete number of samples (the particles) the integral can be replaced by a summation.

$$A_S(r) = \sum_b m_b \frac{A_b}{\rho_b} W(r-r_b,h) \quad (16)$$

where the summation is over all particles. Particle  $b$  has the mass  $m_b$ , the density  $\rho_b$  and the position  $r_b$ .  $A_b$  is the property  $A$  of the particle  $b$ .

For example, the density at location  $r$ ,  $\rho_S(r)$  can be calculated by

$$\rho_S(r) = \sum_b m_b \frac{\rho_b}{\rho_b} W(r - r_b, h) = \sum_b m_b W(r - r_b, h) \quad (17)$$

As we see, the density depends on the mass around the location. More particles, more mass in less space means more density. That matches the physical definition of density.

In most fluid equations, derivatives of field properties appear and need to be evaluated. Using integration by parts the derivatives only affect the smoothing kernel. The smoothing kernel can normally be differentiated analytically. The gradient of  $A$  (the spatial derivative) is simply:

$$\nabla A_S = \sum_b m_b \frac{A_b}{\rho_b} \nabla W(r - r_b, h) \quad (18)$$

Similarly the Laplacian of  $A$  is:

$$\nabla^2 A_S = \sum_b m_b \frac{A_b}{\rho_b} \nabla^2 W(r - r_b, h) \quad (19)$$

## 2.5 SPH and the Navier-Stokes equations

Many forms of the Navier-Stokes equations appear in the literature. For liquids or even air at room temperature a simplification can be used. Incompressible Newtonian fluid has two equations of conservation that have to be guaranteed:

- Conservation of mass:

$$\nabla \cdot v = 0 \quad (20)$$

- and conservation of momentum:

$$\rho \left( \frac{\partial v}{\partial t} + v \cdot \nabla v \right) = -\nabla p + \mu \nabla^2 v + f \quad (21)$$

The use of particles has the advantage of simplifying these equations compared to a stationary (Eulerian) grid:

Because the number of particles is constant and each particle has a constant mass, mass conservation is guaranteed automatically and therefore equation (20) can be omitted.

On the left hand side of equation (21) there is the expression:

$$\frac{\partial v}{\partial t} + v \cdot \nabla v \quad (22)$$

This expression is called the convective derivative of  $v$  and can be written as

$$\frac{\partial v}{\partial t} + v \cdot \nabla v = \frac{Dv}{Dt} \quad (23)$$

The convective derivative, or Lagrangian derivative is the derivative taken with respect to the Lagrangian coordinate system. The Lagrangian coordinate system is a coordinate system moving with the fluid flow (a fixed reference frame is called Eulerian). For more information see [Goldstein 1980]. Since the particles move with the flow, the convective derivative of the velocity is simply the time derivative of the velocity. So the convective term  $v \cdot \nabla v$  is not needed.

With those simplifications, for the acceleration of particle  $i$  we get:

$$a_i = \frac{\partial v_i}{\partial t} = \frac{-\nabla p_i + \mu \nabla^2 v_i + f_i}{\rho_i} \quad (24)$$

We still have to do the calculation of the forces resulting from pressure ( $-\nabla p_i$ ) and viscosity ( $\mu \nabla^2 v_i$ ).

### 2.5.1 Pressure force

By substituting the pressure  $p$  into equation (18), we get the force from pressure at particle  $i$ :

$$f_i^{pressure} = -\nabla p(r_i) = -\sum_b m_b \frac{p_b}{\rho_b} \nabla W(r_i - r_b, h) \quad (25)$$

However this force is not symmetric: The gradient of the smoothing kernel is zero at its center. Therefore the pressure of the particle  $i$  is not taken into account. If e.g. there are only two particles, particle  $i$  only uses the pressure of particle  $b$  for computing its pressure force and vice versa. If the two have different pressures Newton's third law is violated. To symmetrize this equation, different approaches exist. [Monaghan 1992] discusses

$$f_i^{pressure} = -\nabla p(r_i) = -\sum_b m_b \frac{p_b - p_i}{\rho_b} \nabla W(r_i - r_b, h) \quad (26)$$

but finally suggests a more complicated solution. [Müller et al. 2003] suggest the simple and stable solution of using the arithmetic mean of the two pressures:

$$f_i^{pressure} = -\sum_b m_b \frac{p_b + p_i}{2\rho_b} \nabla W(r_i - r_b, h) \quad (27)$$

The pressures in the equation can be calculated using the ideal gas law [Fermi 1937]

$$p = k\rho \quad (28)$$

where  $k$  is a constant depending on the kind of gas and its temperature. [Müller et al. 2003] suggest the use of the equation

$$p = k(\rho - \rho_0) \quad (29)$$

where  $\rho_0$  is the rest density. This leads to a higher numerical stability. More details on the calculation of pressure forces can be found in [Monaghan 1992] and [Müller et al. 2003].

### 2.5.2 Viscosity force

Similarly, by using equation (19), we get the force from viscosity at the particle  $i$ :

$$f_i^{viscosity} = \mu \nabla^2 v(r_i) = \mu \sum_b m_b \frac{v_b}{\rho_b} \nabla^2 W(r_i - r_b, h) \quad (30)$$

This force is not symmetric either, but can be made symmetric like this:

$$f_i^{viscosity} = \mu \sum_b m_b \frac{v_b - v_i}{\rho_b} \nabla^2 W(r_i - r_b, h) \quad (31)$$

More details in [Monaghan 1992] and [Müller et al. 2003].

## 2.6 Simulating surface tension

Surface tension is a property of liquids not present in the Navier-Stokes equations. It is caused by the attraction between the molecules of the liquid. Inside liquid each molecule is pulled equally in all directions by neighboring liquid molecules, resulting in a net force of zero. At the surface of the liquid, the molecules are pulled inwards by other molecules inside the liquid but they are not attracted as intensely by the molecules in the neighboring medium (e.g. air).

The Surface tension force acts in the direction of the surface normal towards the fluid. It tends to minimize the liquid's surface area

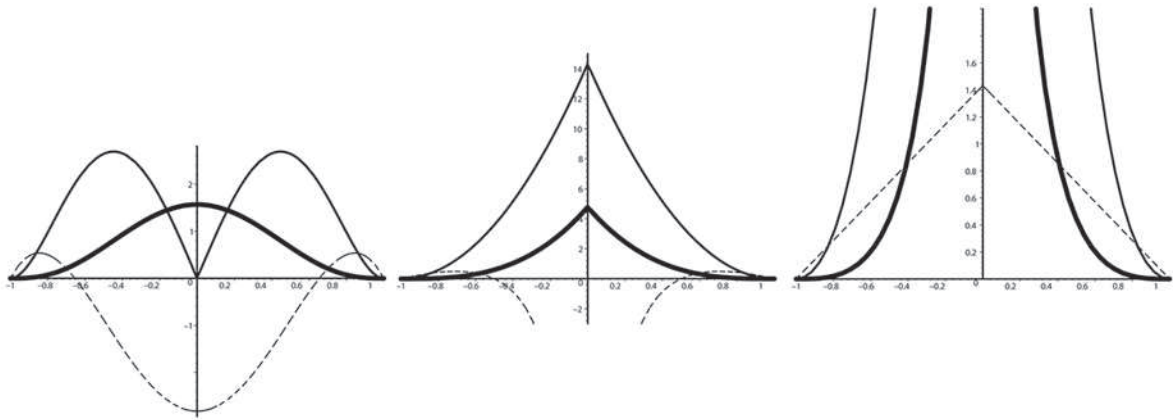


Figure 5: Optimized filter kernels (thick line), their gradients towards the center (thin lines) and their Laplacian (dashed lines) (image: [Müller et al. 2003])

and its curvature. In order to simulate surface tension, surface particles must be identified and their curvature must be minimized. To accomplish those tasks different approaches have been suggested.

One possible approach is the use of the so called *color field*. It is a quantity that is 1 at particle locations and 0 everywhere else. This field is interpolated like the other particle properties. Its gradient corresponds to the surface normal of the fluid. The divergence of the surface normal measures the curvature of the surface. Those properties allow the computation of the surface tension force.

## 2.7 Optimizations

Since the introduction of SPH many optimizations have been suggested in order to improve its speed and accuracy. The most important techniques are briefly reviewed in the following sections.

### 2.7.1 Adaptive kernel size

Because of its interpolating approach, details in the fluid that are below the size of the filter kernel are smoothed out and get lost. In order to avoid this, early experiments with SPH used a filter kernel radius ( $h$ ) varying in time.

The resolution can be increased even more by using a spatially varying kernel size. In areas where there are many particles, a smaller kernel is used, and the details of the higher local resolution are preserved. If there are few particles in an area, a larger filter size guarantees smooth results. But with this approach additional calculations are needed in order to guarantee the conservation of mass and momentum (e.g. symmetric filter kernels). For more details see [Monaghan 1992].

### 2.7.2 Special filter kernels

Stability, accuracy and speed of the SPH method highly depend on the choice of the smoothing kernel. Initially a Gaussian kernel was used. Later the use of kernels based of spline functions proved to provide more stable results. [Müller et al. 2003] suggested a new

set of kernels designed for improved speed and stability, shown in Figure 5:

- A simplified bell shaped kernel which can be evaluated without computing square roots in distance computations
- A special spiky kernel for the computation of the pressure force with the advantage that its gradient does not get zero at the center.
- A special kernel for the calculation of the viscosity force whose Laplacian is positive everywhere. This increases the stability significantly.

Those optimized filter kernels offer more stability and therefore allow bigger time steps to be used, resulting in a faster simulation. Together with a Leap-Frog integration scheme it allows time steps up to 10 milliseconds. More details and formula can be found in [Müller et al. 2003].

### 2.7.3 Adaptive time steps

A problem for all differential solvers lies in the determination of a good step size. It should be as large as possible, but not large enough to introduce a too big error or worse, to induce instability. A fixed step size must be small enough to consider the worst case.

A better solution is to make the time steps adaptive. This can be done by estimating the error (e.g. by using the Courant-Friedrichs-Lewy condition, [Courant et al. 1967]) and choose an appropriate step size based on it.

### 2.7.4 Acceleration through search structures

Theoretically, in a coupled particle system like the SPH approach, each particle interacts with all others. That would be very inefficient. Normally a smoothing kernel with finite size is chosen, which is zero after a certain distance. This reduces the interaction to the particles that lie within the smoothing filter's radius.

The search for the particles which lie within the filter radius can be sped up by using search structures. Often, a grid structure is suggested: The room taken by the particles is divided into equally

sized cubes (cells), whose side length matches the filter radius. So the particles which may interact must be searched only in the current and the directly adjacent cells. This reduces the complexity from  $O(n^2)$  to  $O(nm)$ ,  $m$  being the average number of particles per grid cell.

[Müller et al. 2003] suggest an additional optimization, which can speed up the simulation by a factor of 10. Instead of storing references to the particles in the search grid, they store copies of the particle objects in each cell. This doubles the memory consumption, but because the information needed for the SPH interpolation lies very near in memory the cache hit rate is dramatically increased. This brings the drastic speed increase.

By choosing even more sophisticated search structures additional speed improvements could be made. E.g. Hilbert space filling curves could be used for indexing [Moon et al. 2001]. Space-filling curves are curves (self-similar fractals) whose ranges contain the entire 2-dimensional unit square (or the 3-dimensional unit cube). Algorithms based on them can significantly speed up nearest neighbor searches.

The [Müller et al. 2003] approach with all its optimizations performs quite well: A simulation involving 2200 particles and collisions with solid objects runs at 20 frames per second on a 1.8 GHz Pentium IV PC with GeForce 4 graphics card (used just for displaying the particles).

## 2.8 MPS-MAFL

**MPS** stands for **M**oving-**P**article **S**emi-**I**mplicit and represents another particle based fluid simulation method besides SPH. It was introduced by [Premzoe 2003]. MPS uses a semi-implicit method to solve the Navier-Stokes equations in two steps:

1. Temporary particle locations and velocities are computed from the previous time step. These violate the incompressibility of the fluid.
2. A correction velocity is calculated in order to fulfill conservation of mass, momentum and incompressibility.

**MAFL** stands for **M**eshless **A**dvection using **F**low-directional **L**ocal-grid. It is an extension of the MPS method which alleviates two problems of MPS and any Lagrangian method:

- Inflow and outflow of fluid can not be handled (in contrast to grid-based methods)
- Because the particles move around, local resolution may be weak. MAFL allows to introduce new particles to improve resolution.

The MAFL method consists of three steps:

1. A Lagrangian Phase: the MPS method
2. A Reconfiguration Phase: the particle positions are reconfigured
3. An Eulerian Phase: the particle convection is computed on a "one-dimensional grid"

This method produces very accurate results (see Figure 6 and Figure 7), but is not as efficient as SPH with optimizations. It allows the simulation of very complex effects such as e.g. the interaction of multiple fluids with different properties.

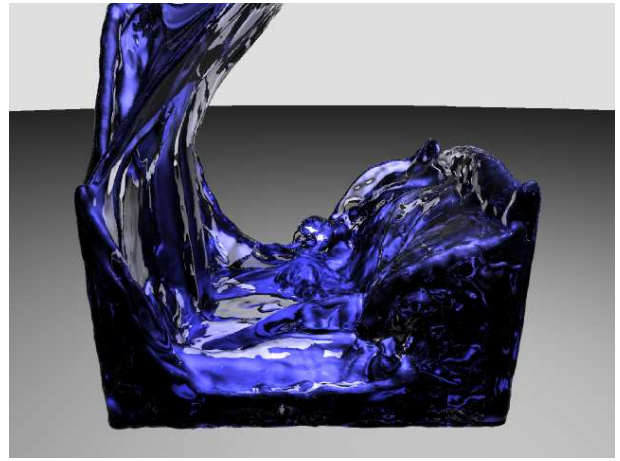


Figure 6: MPS-MAFL simulation using 150,000 particles (image: [Premzoe 2003])



Figure 7: MPS-MAFL simulation of a corridor getting flooded using 100,000 fluid particles. The simulation time is about 3 minutes per frame. (image: [Premzoe 2003])

## 2.9 SPH on the GPU

There exist several techniques for speeding up Eulerian fluid simulations by using the GPU (Graphics Processing Unit). Furthermore, uncoupled particle systems have been GPU-accelerated for years. But running an SPH-based fluid simulation offers some difficulties.

In general, to perform a particle simulation on the GPU, all the properties of the particles have to be stored in textures. Each pixel corresponds to one particle. A texture can hold one vector or 4 scalar properties (in the 4 color channels RGBA). A fragment program is used to update the properties at each time step. Because the input and output texture can not be the same on the GPU, two textures have to be used for the same property. One (the input texture) represents the previous time step, the other (the output texture) represents the current time step. After each time step the two textures are exchanged.

For a SPH particle system the particles in the neighborhood of a 3D position have to be determined. This step does not map well to the GPU. In order to avoid it, [Kolb and Cuntz 2005] have developed a method called Dynamic Particle Coupling. It solves SPH without the need for global sorting or an explicit solution of the n-nearest neighbor problem.

Dynamic Particle Coupling uses 3D textures for storing the properties. The position in the 3D texture corresponds to a real 3D space

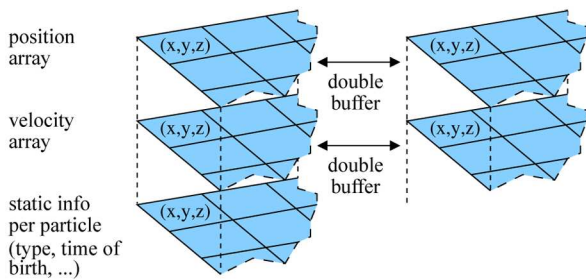


Figure 8: Textures hold the particle properties (image: [Kolb and Cuntz 2005])

position; the 3D texture is a grid discretizing the 3D space. Each particle's contribution is accumulated in those textures. After having performed this for all particles, the resulting properties can be retrieved by sampling in the 3D textures. For computing the properties separable summands are used.

The computation of the properties is done using the GPU. Since current (up to Shader Model 3) GPUs can only render to 2D textures, the 3D texture has to be build using a stack of 2D textures, called slices. For each slice all particles that belong to it are drawn as point sprites. So the contribution of each particle is spread in 3D space (this is somewhat similar to point splatting, used for volume and point primitive rendering). Particles can contribute to more than one slice. Those that do not belong to the current slice are clipped by properly set near and far clipping planes.

In order to reduce the needed number of passes and therefore improve speed, multiple render targets (MRT) are used to compute 4 slices in parallel. Additionally, several slices can be put in one 2D texture as subregions.

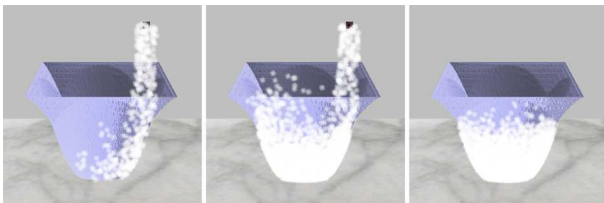


Figure 9: A cup is filled with water (running at 12 fps on the GPU) (image: [Kolb and Cuntz 2005])

This fluid simulation technique is able to deliver satisfactory results at interactive frame rates. A simulation involving 2400 particles in a  $32^3$  force field and a kernel radius of 20% of the force field dimension runs at 12 frames per second. This result is generally not bad, but quite disappointing, when compared with the CPU-based approach by [Müller et al. 2003], which runs faster and is more flexible. But Kolb and Cuntz' GPU-based approach is just a proof of concept and has much room for improvement. E.g. now (two years later) with DirectX 10 and Shader Model 4 graphics hardware is able to render directly to 3D textures. When sampling them trilinear interpolation in hardware can be used. These and some other improvements, together with the increased fragment program processing power, could make this approach much faster. Maybe even faster than the before mentioned CPU-based approach.

## 2.10 Fluid rendering

All the previous sections dealt with how to simulate the motion of fluids. But to visualize a realistic fluid, also it's rendering is important. Different techniques are needed depending on what kind of fluid is being visualized.

For gaseous phenomena such as fire, smoke, steam or clouds:

- **Raycasting:** A common volume rendering technique, where rays are sent through the volume to calculate its reaction to light. It is better suited for grid-based simulations, since it needs a volume expressed as voxels.
- **Sprites/volumetric sprites:** A simple form of rendering particles is to place a semitransparent image (sprite) at each particle location. Volumetric sprites use three-dimensional volumes instead of the image planes.
- **Splatting:** Another technique for volume rendering, where instead of sampling the volume, the volume is "splatted" onto the image plane.

Liquids usually have a well defined surface which separates them from the surrounding material (like e.g. air). This surface has to be determined and rendered. Surface particles can be identified by the color field and it's gradient. If the gradient of a particle's color field exceeds a threshold, it is regarded a surface particle. In literature other surface tracking methods exist.

Once the surface particles are identified, a continuous surface has to be constructed. Different approaches exist, e.g.:

- **Point Splatting:** The particles are rendered directly, building a smooth surface in image space. However this method is designed for very dense point clouds (like e.g. from 3D laser scanners). But it still delivers acceptable results.
- **Marching Cubes:** This is a technique to triangulate the fluid surface, so it can be rendered like any other polygon object.

## 2.11 Conclusion

Particle-based methods are a good attempt to bring fluid simulation to realtime applications. They are fast, straightforward to understand and implement and deliver good results. There even exists an implementation on the GPU. So are we soon going to see games with massive photorealistic fluid simulations? Probably not. To achieve interactive frame rates particle numbers still need to be quite low. Because of this lack of resolution realtime fluids still tend to look somewhat clumpy and too viscous. And for sure such particle counts are not enough to simulate ships sinking spectacularly in a turbulent ocean or monster waves flooding whole cities. Such effects still belong to the cinema and take hours per frame to simulate.

Still, there are first steps in the direction of realtime fluid in games. Both of the two leading physics engines - Havok and PhysX - offer simple fluid simulation in their new versions. Demos show water flowing off a car, descending fog or a bursting water tank (See Figure 11 and Figure 12). It's only a matter of time until first games use those effects. And in the future growing hardware capabilities will bring even more sophisticated simulations.



Figure 10: Three different rendering modes: the particles, the surface using point splatting and the iso-surface triangulated via marching cubes. (image: [Müller et al. 2003])

## References

- ANDERSSON, L. 2005. *Real-Time Fluid Dynamics for Virtual Surgery*. Master's thesis, CHALMERS UNIVERSITY OF TECHNOLOGY, Department of Computer Engineering, Göteborg.
- COURANT, R., FRIEDRICHS, K., AND LEWY, H. 1967. On the partial difference equations of mathematical physics. *IBM Journal (March)*, 215–234.
- FERMI, E. 1937. *Thermodynamics*. Dover.
- GINGOLD, R. A., AND MONAGHAN, J. J. 1977. Smoothed particle hydrodynamics - theory and application to non-spherical stars. *Royal Astronomical Society, Monthly Notices 181* (nov), 375–389.
- GOLDSTEIN, H. 1980. *Classical Mechanics*, 2<sup>nd</sup> ed. Addison-Wesley, Reading, MA, U.S.A. 672 pages.
- KOLB, A., AND CUNTZ, N. 2005. Dynamic particle coupling for gpu-based fluid simulation. *Proc. 18th Symposium on Simulation Technique*, 722–727.
- LENNARD-JONES, J. 1931. Cohesion. *Proceedings of the Royal Society of London 43*, 240, 461–482.
- LUCY, L. B. 1977. A numerical approach to the testing of the fission hypothesis. *Astronomical Journal 82* (dec), 1013–1024.
- MAREK, S. 2007. Grid-based realtime fluid dynamics. Tech. rep., Institute of Computer Graphics and Algorithms, TU Vienna, Austria.
- MONAGHAN, J. J. 1992. Smoothed particle hydrodynamics. *Annual review of astronomy and astrophysics 30*, 543–574.
- MOON, B., JAGADISH, H. V., FALOUTSOS, C., AND SALTZ, J. H. 2001. Analysis of the clustering properties of the hilbert space-filling curve. *IEEE Transactions on Knowledge and Data Engineering 13*, 1, 124–141.
- MUELLER, M., SCHIRM, S., TESCHNER, M., HEIDELBERGER, B., AND GROSS, M. 2004. Interaction of fluids with deformable solids. *Computer Animation and Virtual Worlds 15*, 3-4 (June), 159–171.
- MÜLLER, M., CHARYPAR, D., AND GROSS, M. 2003. Particle-based fluid simulation for interactive applications. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 154–159.
- MURTA, A., AND MILLER, J. 1999. Modelling and rendering liquids in motion. In *WSCG'99 Conference Proceedings*, V. Skala, Ed.
- PREMZOE, S. 2003. Particle-based simulation of fluids. 401–410.
- REEVES, W. T. 1983. Particle systems - a technique for modeling a class of fuzzy objects. *ACM Trans. Graph.* 2, 2, 91–108.
- WITKIN, A., AND BARAFF, D., 1997. Physically based modeling: Principles and practice. Siggraph '97 Course notes.



Figure 11: Screenshot from the Ageia PhysX Car Wash demo showing flowing water (*image: ageia*)



Figure 12: Screenshot from another Ageia PhysX demo showing a liquid flowing out of a broken water tank (*image: ageia*)