[1 Introduction 2](#_Toc192496935)

[1.1 Motivation 2](#_Toc192496936)

[1.2 How to simulate fluids 2](#_Toc192496937)

[1.3 Related work 3](#_Toc192496938)

[1.4 Used techniques 4](#_Toc192496939)

[2 Fluid simulation 5](#_Toc192496940)

[2.1 Smoothed particle hydrodynamics 5](#_Toc192496941)

[3 Visualization 7](#_Toc192496942)

[3.1 Optical characteristics of water 7](#_Toc192496943)

[4 Conclusion 8](#_Toc192496944)

[4.1 Summary 8](#_Toc192496945)

[4.2 Improvements and alternatives 8](#_Toc192496946)

[Appendix 9](#_Toc192496947)

[References 9](#_Toc192496948)

[Glossary 9](#_Toc192496949)

# Introduction

## Motivation

Fluids like liquids and gases are ubiquitous parts of the environment we live in. For instance we all know how it looks like when milk gets filled into a drinking glass. In realtime computer graphics, where we traditionally try to reproduce parts of our world as visually realistic as possible, it’s unfortunately hard to simulate such phenomena. Computational fluid dynamics is a relatively old and well known research topic, but most applications (like i.e. in aerodynamics research) aim at results that are as accurate as possible. Therefore the simulations are mostly calculated offline and realtime visualization is mostly used only to render precomputed data sets, if at all.

|  |  |
| --- | --- |
| 2.png  3.png  Figure : Example for offline simulation Source:[APK07] | 1.png  Figure : Example for realtime simulation Source: [MCG03] |

Realtime applications that do allow the user to interact with authentically (but not necessarily accurately) simulated and rendered fluids (like i.e. water) are today rare. For all types of virtual realities, like surgical training environments or computer games, there’s always demand to cover more aspects of our world and so realtime simulation and rendering of fluids is an interesting field of study. In 2003 Müller, Charypar and Gross sparked additional interest in realtime fluid simulation, with a paper that proposed a relatively simple, particle based fluid-model, that fits well for realtime applications [MCG03]. Since then many different aspects of realtime particle based fluid simulation where covered in a couple of papers from authors around the world. This thesis gives an overview on the topic, as it discusses my implementation of a particle based fluid simulation and a suitable water renderer.

## How to simulate fluids

In the nineteenth century Claude Navier and George Stokes created the fundamentals of modern fluid dynamics as they formulated the well known Navier-Stokes equations. With these equations, which describe the conservation of momentum, together with two additional equations for mass and energy conservation, it’s possible to simulate the fluid flow. As the formulas tend to get very complicated for less common fluids, they are mostly written for Newtonian fluids which include a variety of common liquids and gases (water, air...).

Simulations apply numerically methods to solve the (in most cases) resulting nonlinear partial differential equations. One common way to do this is to treat the fluid as a continuum, discretize the spatial domain into a grid and use finite differences or the finite volume method. In the literature grid based fluid models are called Eulerian models. For the use within virtual environments grid based methods, as a matter of principle, have the drawback of a bounded simulation space.

Particle based methods (in literature: Lagrangian model, from Lagrangian mechanics) in contrast represent the fluid as a discrete set of particles and simulate the fluid flow through solving the particle dynamics. For realtime applications this brings some advantages over grid based methods:

* simpler calculation (mass conservation can be omitted, convective term can be omitted, see [MCG03])
* no numerical diffusions in the convection terms (diffusion directions are not influenced by the grid layout)
* surface reconstruction is likely to be easier
* fluid can spread freely in space (no boundary through the grid)

For those reasons (especially the last) this thesis focuses on a Lagrangian method based on smoothed particle hydrodynamics (SPH) [Mon05] which became very popular for this kind of applications. The idea behind SPH is that every particle distributes the fluid properties in its neighborhood using radial kernel functions. To evaluate some fluid property at a given point one must simply sum up the properties of the neighboring particles, weighted with the appropriate smoothing function.

## Related work

The first investigations in smoothed particle hydrodynamics where made in 1977 by Gingold and Monaghan (who coined the term) [GM77] and independently by Lucy [Luc77]. Its first usages took place mainly in the astronomy sector to simulate large scale gas dynamics, but later it also has been applied to incompressible flow problems like beach wave simulation, sloshing tanks and bow waves of ships.

While in realtime computer graphics first the Eulerian approach was favored, Müller, Charypar and Gross [MCG03] where one of the first who showed, that a SPH based Lagrange method also suits very well to interactive applications. Later many papers used SPH to simulate fluids (especially liquids) in realtime and brought adaptations and improvements both for the simulation as well as for the rendering of liquids.

Papers on realtime SPH simulation:

* [KC05] proposes to avoid the particle neighborhood problem by sampling the fluid properties from grids witch sum up the weighted properties from all particles
* [KW06] compares the performance of an octree based (linear time for neighbor search, but large costs for the update of the structure) versus a “staggered grid” based solution to the neighbor problem
* in [MST04] Müller et al. show how particle based fluids can interact with deformable solids
* [AIY04] sketches how to use a CPU generated neighbor map so that the property summation for each particle can be handled on the GPU which reaches twice the performance of their CPU only simulation
* [Hei07] uses the Ageia PhysX engine (one of its developers is Matthias Müller) for a SPH based simulation of smoke

Papers with relevance for realtime liquid rendering:

* [MCG03] suggests direct point splatting of the particles or marching cubes rendering [LC87] of the isosurface (which implies that an isovolume must be created for each frame)
* [KW03] presents a GPU executed isovolume raycaster; in combination with a efficient method for building the isovolume on the GPU this way the iso surface could be visualized
* [CHJ03] introduces iso-splatting, a point based isosurface visualization technique; same as with [KW03] applies here
* [Ura06] demonstrates a GPU version of the marching tetrahedra algorithm (variation of marching cubes); same as with [KW03]
* [KW06] uses a 2.5D “carped visualization” for the special case of rivers and lakes

## Used techniques

The goal with this thesis was to provide a realtime application that simulates a water-like liquid in a form that is “believable” in terms of movement behavior and optical appearance. The SPH simulation therefore focuses not on physical accuracy. It’s a straightforward implementation of the lightweight SPH model presented in [MCG03], optimized to run on actual multi-core consumer CPUs. To speed up the neighbor search it stores the particles according to their position in a dynamic grid, with a cell size equivalent to the maximal radius of support. The particle interactions are evaluated directly on pairs of particles (simultaneous for both particles). Chapter 2 discusses the theoretical foundations and the implementation details of the simulation.

For visualization three techniques are provided: The first directly renders the particles as point sprites, which is mainly useful for debug and tuning of the fluid behavior. The second, which is nearly entirely CPU-based, uses the marching cubes algorithm to construct a triangle mesh representing the isosurface. This technique was implemented to experiment with efficient isovolume construction methods and to test how well a marching cubes / triangle based approach fits for the purpose of liquid visualization. The last and most sophisticated technique uses the GPU to construct an isovolume within a 3D texture and renders the isosurface directly with a raycasting shader. The raycasting enables the visualization of effects like multiple refractions and reflections, which are characteristic for the optical appearance of liquids. Chapter 3 explains each visualization technique in detail.

|  |  |  |
| --- | --- | --- |
| 1.jpg  Figure : Sprite visualization | 2.jpg  Figure : Marching cubes visualization | 3.jpg  Figure : GPU raycasting visualization |

# Fluid simulation

## Chapter overview

## Basics of fluid mechanics

Fluid mechanics normally deals with macroscopic behavior at length and time scales where intermolecular effects do not matter. In this situation fluids can be treated as continuums where every property has a definite value at each point in space. Mathematically this can be expressed through functions that depend on position and time (i.e. vector or scalar fields). Properties are macroscopic observable quantities that characterize the state of the fluid. For the movement of fluids the most relevant properties are mass, density, pressure, viscosity and surface tension. The mass property specifies “how much matter there is” and is relevant for the inertia of the fluid. The mass density measures the mass per volume and is defined as:

|  |  |  |
| --- | --- | --- |
|  |  |  |

: very small length, but significant greater than the molecule spacing; : volume

Pressure is a scalar quantity that’s defined as the force acting in normal direction on a surface (normal stress): . The viscosity measures how much momentum is transferred between regions with different flow speeds and is therefore responsible for shear stress (tangential force on a surface). It’s stated as dynamic viscosity or kinematic viscosity . The surface tension at last characterizes the forces that act at the interface of two immiscible fluids (i.e. water and air). These forces try to minimize the area and curvature of the surface, because the molecules are more likely to group together with molecules of their own kind.

Now that we know for which properties the symbols stand, let’s see how the motion of a fluid could be described. We will start with Newton’s second law:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Note: vectors are written in bold (), scalars in italics ().

It states that the acceleration of an object depends on its mass and the force that acts on it. This could also be interpreted as conservation of momentum: Without external forces () there’s no change of velocity () and the momentum stays constant.

In classical (Newtonian) dynamics Newton’s second law is usually interpreted from the Lagrangian point of view, meaning that a moving object is observed. With fluids this would mean that the observation area follows the fluid flow, so that always the same “amount of fluid” is being watched. Alternatively in the Eulerian point of view the area of observation is locally fixed, so that the fluid passes by and the watched amount of fluid may be a different one at each moment. The Eulerian observer therefore not only sees changes due to variances in the currently watched amount of fluid, but also changes due to the fact that the watched amount of fluid may be a different one every moment.



Figure : Lagrangian versus Eulerian point of view

In an Eulerian description the acceleration therefore must be a special time derivative of the velocity, which takes the convection of properties by the fluid flow into account. It is called substantial derivative (synonyms: substantive d., convective d., material d.) and defined as follows:

|  |  |  |
| --- | --- | --- |
|  |  |  |

written for three dimensional Cartesian coordinates; : del operator; : components of velocity; : components of position; : an arbitrary quantity (vector or scalar)

By replacing the acceleration in (2.2) with the substantive derivative of the velocity we get:

|  |  |  |
| --- | --- | --- |
|  |  |  |

: gradient of the velocity (the Jacobian matrix)

The mass of the fluid inside the observed control volume depends on its density, therefore we write:

|  |  |  |
| --- | --- | --- |
|  |  |  |

## Basics of smoothed particle hydrodynamics

Smoothed particle hydrodynamics is a technique developed by Gingold and Monaghan [GM77] and independently by Lucy [Luc77] for the simulation of astrophysical gas-dynamics problems. As in other numerical solutions to fluid dynamic problems, the value of a physical quantity at a given position must be interpolated from a discrete set of points. SPH derives from the integral interpolation:

|  |  |  |
| --- | --- | --- |
|  |  |  |

is a radial symmetric smoothing function (also called kernel) with smoothing length (also called core radius). One could say that the interpolation uses the smoothing kernel to spread a quantity from a given position in its surroundings. In practice the kernel is even () and normalized () and tends to become the delta function for tending to zero (if would be the delta function, would reproduce exactly). This thesis follows the example of [MCG03] to treat as the radius of support, so all used smoothing functions will evaluate to zero for .

1.wmf

Figure : 1D example for a smoothing kernel

With SPH the interpolation points are small mass elements, which aren’t fixed in space (like the grid points in the Euler method) but move with the fluid. For each such fluid particle a position , velocity , mass and density is tracked. The value of a quantity at a given position can be interpolated from the particle values using the summation interpolant (derived from the integral form):

|  |  |  |
| --- | --- | --- |
|  |  |  |

The mass-density-coefficient appears because each particle represents a volume of . As an interesting example (2.2) applied to the density gives

|  |  |  |
| --- | --- | --- |
|  |  |  |

which shows that with SPH the density is estimated by smoothing the mass of the particles.

An advantage of SPH is that spatial derivatives (which appear in many fluid equations) can be estimated easily. When the smoothing kernel is differentiable the partial differentiation of (2.2) gives:

|  |  |  |
| --- | --- | --- |
|  |  |  |

The gradient therefore becomes:

|  |  |  |
| --- | --- | --- |
|  |  |  |

According to [MCG03] this could also be applied to the Laplacian:

|  |  |  |
| --- | --- | --- |
|  |  |  |

This formulations cause some problems when they are used to derive fluid equations for particles. The derivate does not vanish when is constant and a number of physical laws like symmetry of forces and conservation of momentum are not guaranteed. Therefore we will have to adjust the particle fluid equations slightly to ensure physical plausibility.

## Modeling fluid dynamics with particle systems

For simplification we will discuss only Newtonian fluids that satisfy the incompressible flow condition. A fluid is called Newtonian when the shear stress is proportional to the rate of strain. For Newtonian fluids the equation describes the relation between shear stress , viscosity and the velocity gradient perpendicular to the direction of share . This means in common words that in contrast to non-Newtonian fluids the viscosity is a constant and does not change under different shear rates. The fluid flow is called incompressible when the divergence of the velocity field is zero (), meaning that there are no sources or sinks in the velocity field. As a counter example think of air that expands because it’s heating up. Note that also flows of compressible fluids can satisfy the incompressible flow condition (i.e. air till ~ mach 0.3).

# Visualization

## Optical characteristics of water

# Conclusion

## Summary

## Improvements and alternatives

# Appendix

## References

**[APK07] Adams, et al. 2007.** Adaptively Sampled Particle Fluids. *Proceedings of the 2007 SIGGRAPH conference.* 2007.

**[AIY04] Amada, et al. 2004.** Particle-Based Fluid Simulation on GPU. *ACM Workshop on General-Purpose Computing on Graphics Processors.* 2004.

**[CHJ03] Co, Hamann and Joy. 2003.** Iso-splatting: A Point-based Alternative to Isosurface Visualization. *Computer Graphics and Applications, 2003. Proceedings.* 2003, pp. 325-334.

**[GM77] Gingold and Monaghan. 1977.** Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Royal Astronomical Society, Monthly Notices.* 181, 1977, pp. 375-389.

**[Hei07] Heinecke. 2007.** Physikalische Rauchsimulation auf Partikelbasis in Echtzeit mit der PhysX-Engine. [ed.] Technische Universität Dresden. [Minor thesis]. 2007.

**[KW06] Kipfer and Westermann. 2006.** Realistic and interactive simulation of rivers. *ACM International Conference Proceeding Series.* 2006, 137.

**[KC05] Kolb and Cuntz. 2005.** Dynamic Particle Coupling for GPU-Based Fluid Simulation. *Proc. 18th Symposium on Simulation Technique.* 2005, pp. 722-727.

**[KW03] Krüger and Westermann. 2003.** Acceleration Techniques for GPU-based Volume Rendering. *Proceedings of the 14th IEEE Visualization.* 2003, p. 38.

**[LC87] Lorensen and Cline. 1987.** Marching cubes: A high resolution 3D surface construction algorithm. *Proceedings of the 14th annual conference on Computer graphics and interactive techniques.* 1987, pp. 163-169.

**[Luc77] Lucy. 1977.** A numerical approach to the testing of the fission hypothesis. *Astronomical Journal.* 82, 1977, pp. 1013-1024.

**[Mon05] Monaghan. 2005.** Smoothed particle hydrodynamics. *Reports on Progress in Physics.* 2005, 8.

**[MCG03] Müller, Charypar and Gross. 2003.** Particle-Based Fluid Simulation for Interactive Applications. *Proceedings of 2003 ACM SIGGRAPH Symposium on Computer Animation.* 2003, pp. 154-159.

**[MST04] Müller, et al. 2004.** Interaction of Fluids with Deformable Solids. *Computer Animation and Virtual Worlds.* 2004, 15, pp. 159 - 171.

**[Pap99] Papanastasiou, Georgiou and Alexandrou. 1999.** *Viscous Fluid Flow.* s.l. : CRC Press, 1999. ISBN13: 9780849316067.

**[Ura06] Uralsky. 2006.** Practical Metaballs and Implicit Surfaces. *Game Developers Conference 2006 Presentation.* 2006. http://developer.nvidia.com/object/dx10-practical-metaballs.html.

**[WND]** wikipedia.org. *Navier-Stokes equations/Derivation.* [Online] [Cited: 03 10, 2008.] http://en.wikipedia.org/w/index.php?title=Navier-Stokes\_equations/Derivation&oldid=177609104.

## Glossary

## 

Eulerian model 3

Lagrangian model 3

Navier-Stokes equations 2

smoothed particle hydrodynamics 3