**Fluidsimulation in Computergraphics:
Weakly Compressible Smoothed Particle Hydrodynamics
and Position Based Fluids**

Abstract

Introduction

> Reading maketh a full man; conference a ready man; and writing an exact man.
>
> *Sir Francis Bacon*

Methods

In the following section we will give an overview of our implementation of a fluidsolver based on the SPH and PBF frameworks. We will take a look at the general structure of the implementation and discuss specific parts that are different from the given assignment. For a thorough discussion of the SPH and PBF framework we refer the reader to the review of SPH fluids by Ihmsen et al. [1] and to the original paper about position based fluids by Macklin et al. [2].

The implementation uses a C-style imperative approach with the exception of a few object oriented paradigms. Data encapsulation in the form of C++ classes is applied to ensure data consistency during computation. Basic inheritance is used to be able to reuse code for different parts of the program. This style of programming allows us to be flexible enough to incrementially implement the practical assignment tasks while sacrificing little to no performance due to design overload.

The implementation is divided into three general categories. The solvers, the emitter and particlesystems and the category of auxilary functions. A particlesystem is a collection of particles which are represented by basic attributes, such as mass, density, position, velocity and the accumulated external forces on the particles. Since it is vital to keep all attributes of a particlesystem consistent during computation a particlesystem object is only allowed to be created by the emitter object. The emitter object is a singleton, which means that there is always one and only one single instance of the object during the runtime of the program. During runtime a particlesystem is present in two forms, either as fluid particles or as boundary particles. These subtypes posess attributes and methods responsible for the behaviour of fluid or boundary particles in a system. For example, boundary particles possess a seperate variable volume attribute and a method to compute the volume for each particle depending on the density, e.g. the number of boundary particles in close neighbourhood.

Boundary particles are used to realise the interaction between the fluid and rigid boundaries. As boundaries in a scene are usually represented by triangular mesh models, we create a boundary particlesystem by sampling each triangle of a given input mesh. To this end, it is reasonable to dissolve a higher resolution mesh into a low resolution mesh which still contains the shape information and use this low resoltion mesh for the simulation and keep the original high resolution mesh for rendering the final simulation. This will reduce the overall number of triangles and therefore the number of sampled particles.

Figure 1: High resoltuion mesh for rendering (left) and low resolution mesh for simulation (right). The mesh on the left maintains the qualitative shape of the mesh.

To sample the particles on a triangle surface we perform a variation of the Pineda algorithm [3] in the local coordinate system of the triangle surface. Given a triangle specified by the three vertex points $a, b, c \in \mathbb{R}^3$, we construct the orthonormal base vectors $u, v, n \in \mathbb{R}^3$ defining the triangle surface.

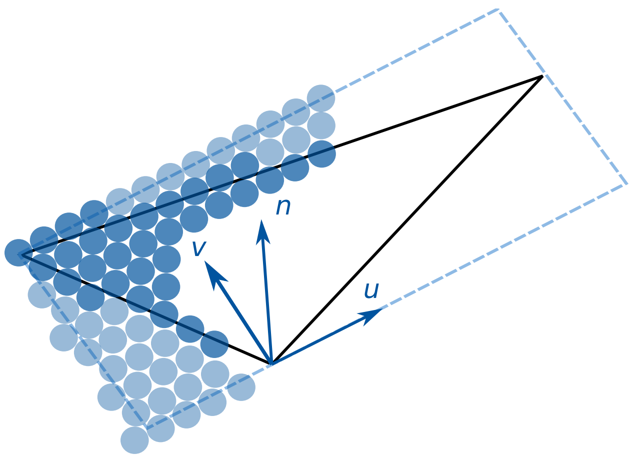

Figure 2: Hexagonal sampling using pineda algorithm in local triangle space.

The boundary particle radius should optimally be the same as the fluid particle radius. This is due to the timestep being determined by the Courant-Friedrich-Levy (CFL) condition, e.g. $\Delta t \leq \lambda \frac{h}{v^{max}}$ which dictates, that a particle moves a maximum distance of its own radius h per time-stamp Δt. If the boundary particle radius is much smaller. The fluid particles may tunnel through the boundary if the timestep is too high. If the boundaryparticle radius is too high, the boundary shape information is lost and the interaction therefore incorrect. The constant factor λ in the CFL is set around 0.4 in all further simulations.

Results

Conclusion

Outlook

References

[1] M. Ihmsen, J. Orthmann, B. Solenthaler, A. Kolb, and M. Teschner, "SPH Fluids in Computer Graphics," in *Eurographics 2014 - State of the Art Reports* (S. Lefebvre and M. Spagnuolo, eds.), The Eurographics Association, 2014.

[2] M. Macklin and M. Müller, "Position based fluids," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 4, pp. 1–12, 2013.

[3] J. Pineda, "A parallel algorithm for polygon rasterization," in *Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pp. 17–20, 1988.