Smoothed Particle Hydrodynamics in Flood Simulations

Michal Chládek*
Comenius University in Bratislava

Roman Ďurikovič†
Comenius University in Bratislava

Abstract

Water simulation and water effects are nowadays an evolving field. In recent years we have seen a big advance in fluid simulations. In this paper we present a method for flood simulations of cities and complex models. SPH method is used for the fluid simulation by which we gain less expensive computation than by using other known methods of simulation. Because of complex models, collision handling takes an important role in the simulation and becomes a computational burden. In this paper we propose a new and fast approach of collision handling in particle methods of fluid simulation by using the distance from the surface.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modelling—Physically based modelling; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

Keywords: SPH, collision handling, flood simulation

1 Introduction

Physically based animations have been an active research area in computer graphics. Besides Eulerian approaches, where physical values are stored in a grid, Lagrangian approaches have been adopted. In contrast to Eulerian methods, Lagrangian methods do not use a grid. Instead they use particles, which are not connected, and store physical values in them.

Two main aproaches are used for particle fluid simulation. Namely smoothed particle hydrodynamics (SPH) [Monaghan 1992] and MPS [Koshizuka and Oka 1996]. SPH is often used even if it computes compressible flow, because of its lower computation cost compared to MPS. SPH solves incompressible flow only approximately. The incompressibility is achieved either by small time steps used in the integration or by using correcting methods like in [Solenthaler and Pajarola 2009].

In this paper, we aim at proposing a solution for flood simulation of cities. We concentrated more on visual realism than physical correctness. SPH was used to run the simulation faster. In particle methods boundaries are often represented by particles. Thus, the collision handling and the fluid simulation has a unified approach. With the size and the complexity of a model, the number of particles used to represent boundaries increases rapidly and becomes an issue. Therefore we use alternative ways how to solve the collision.

*e-mail: mychalch@gmail.com
†e-mail: durikovic@fmph.uniba.com

The paper is organized as follows: In the next section we describe SPH method that we used in particle simulation. In Section 3, we show collision handling based on intersections as well as our proposed method. In Section 4, we briefly describe our visualisation method. In the end, we present our results.

2 SPH

Smoothed particle hydrodynamics is a Lagrangian fluid simulation method originally developed for simulating astrophysical phenomena. Later, it was adopted in other fields like highly deformable solids ([Desbrun and Gascuel 1996]) or computational fluid dynamics ([Müller et al. 2003]).

SPH is an Lagrangian i.e. interpolation method. It approximates values and derivatives of a continuous fluid by interpolated discrete samples represented by particles. Each particle carries properties like mass, position, or velocity. These values can be computed at any point in space by interpolating the properties of neighbouring particles. This is done by so called kernel functions or smoothing kernel:

$$A_s = \sum_b m_b \frac{A_b}{\rho_b} W(\mathbf{r} - \mathbf{r}_b, h), \qquad (1)$$

where we sum over all particles, m_b is mass of particle with index b, ρ_b is its density, and A_b is the property A of particle b at position $\mathbf{r}_b$. W is a kernel function with support radius h. The kernel function must be even i.e.

$$W(\mathbf{r}, h) = W(-\mathbf{r}, h) \qquad (2)$$

and normalized

$$\int W(\mathbf{r} - \mathbf{r}', h) d\mathbf{r}' = 1. \qquad (3)$$

We use kernel function suggested by [Müller et al. 2003].

For example, to compute density of particle s, we substitute A in Eq. 1 by density and get

$$\rho_s = \sum_b m_b \frac{\rho_b}{\rho_b} W(\mathbf{r} - \mathbf{r}_b, h) = \sum_b m_b W(\mathbf{r} - \mathbf{r}_b, h). \qquad (4)$$

We can see that it corresponds with the physical definition of density because it represents mass around a particle.

Unlike in Eulerian approaches with particles in a grid, the derivatives in SPH can be computed at an arbitrary point in the fluid. The advantage of Eq. 1 is that the derivation affects only the kernel functions. The gradient of A is

$$\nabla A_s = \sum_b m_b \frac{A_b}{\rho_b} \nabla W(\mathbf{r} - \mathbf{r}_b, h) \qquad (5)$$

and the Laplacian of A is

$$\nabla^2 A_s = \sum_b m_b \frac{A_b}{\rho_b} \nabla^2 W(\mathbf{r} - \mathbf{r}_b, h). \qquad (6)$$

2.1 SPH and Navier-Stokes Equations

Because we are working with isothermal fluids, the Navier-Stokes equations can be used in a simplified form. Fluid is expressed by a velocity field $\mathbf{v}$, a density field ρ and a pressure field p. Thus by expressing the evolution of the fluid in the time, we get one equation for conservation of mass

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0, \tag{7}$$

and another one for conservation of momentum

$$\rho \left(\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v} \right) = -\nabla p + \rho \mathbf{g} + \mu \nabla^2 \mathbf{v}. \tag{8}$$

This equation can be further simplified. Because we use a particle method, where the particle mass and the particles number is constant, conservation of mass is guaranteed. Thus the Eq. 7 can be omitted. The expression $\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v}$ in Eq. 8 is the convective derivative, which is a derivative taken with respect to the Lagrangian coordinate system. The Lagrangian coordinate system is a coordinate system, which moves together with the fluid flow. Because particles move with the flow we can use time derivative of the velocity instead of convective derivative.

$$\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v} = \frac{D\mathbf{v}}{Dt} \tag{9}$$

After the simplifications of Navier-Stokes equations we get:

$$\mathbf{a}_i = \frac{\partial \mathbf{v}_i}{\partial t} = \frac{-\nabla p_i + \mu \nabla^2 \mathbf{v}_i + \mathbf{f}_i}{\rho_i}. \tag{10}$$

Now we have to calculate forces resulting from pressure $(-\nabla p_i)$ and viscosity $(\mu \nabla^2 \mathbf{v}_i)$. This is done using Eq. 1. By substituting p into this equation we get

$$f_i^{pressure} = -\nabla p(\mathbf{r}_i) = -\sum_b m_b \frac{p_b}{\rho_b} \nabla W(\mathbf{r}_i - \mathbf{r}_b, h). \tag{11}$$

Analogically, we derive the equation for viscosity

$$f_i^{viscosity} = \mu \nabla^2 \mathbf{v}(\mathbf{r}_i) = \mu \sum_b m_b \frac{v_b}{\rho_b} \nabla^2 W(\mathbf{r}_i - \mathbf{r}_b, h). \tag{12}$$

To calculate the pressure we use approach suggested by [Desbrun and Gascuel 1996]

$$p = k(\rho - \rho_0), \tag{13}$$

where k is the fluid stiffness and ρ_0 is the rest density. By using this equation instead of ideal fluid law ($p = k\rho$), the fluid tends to hold density near the rest density.

The problem with Eqs. 11 and 12 is that the forces resulting from them, are not necessarily symmetric. When we compute this forces for two particles which don't have the same pressure or velocity, the forces differ, and the third Newton's law is violated. Many approaches exist to make this equations symmetric. We use the approach from [Müller et al. 2003]:

$$f_i^{pressure} = -\sum_b m_b \frac{p_b + p_i}{2\rho_b} \nabla W(\mathbf{r}_i - \mathbf{r}_b, h) \tag{14}$$

and

$$f_i^{viscosity} = \mu \sum_b m_b \frac{v_b - v_i}{\rho_b} \nabla^2 W(\mathbf{r}_i - \mathbf{r}_b, h). \tag{15}$$

Except of these forces gravitation and surface tension are used. We implemented surface tension as in [Müller et al. 2003]:

$$c_s(\mathbf{r}) = \sum_j m_j \frac{1}{\rho_j} W(\mathbf{r} - \mathbf{r}_j, h), \tag{16}$$

$$\mathbf{n} = \nabla c_s, \tag{17}$$

$$\mathbf{f}^{surface} = -\sigma \nabla^2 c_s \frac{\mathbf{n}}{\|\mathbf{n}\|}, \tag{18}$$

where σ is the surface tension coefficient. Leap-frog integration is used for advancing position and velocity every time step.

3 Collision Handling

Collision handling makes a relevant part of our simulation. It has to work properly with complex geometry models. It is also important to make it computationally less expensive in order to accelerate the computation.

One of widely used methods for collision handling is sticking the surface with particles or representing the solid objects with particles. We have decided to not handle the collisions in this way because the number of particles increases rapidly with the size of the model. By simulating a flood of a city, this would be a great computational burden. We implemented two ways of collision handling. One method is based on intersections and the other one on distances.

3.1 Intersection Method

Handling of collisions can be divided into two parts. Collision detection and collision response. First we have to detect a collision and then properly respond to it. Consider that a collision occurred in time between t_i and t_{i+1}. Define a segment by position of collided particle in time t_i and t_{i+1}. There has to be an intersection of this segment with the mesh. Thus the collision detection is just finding an intersection of a segment with a mesh. In response to the collision we move the particle back to the intersection point and reflect the velocity along the surface normal:

$$\mathbf{v}_i = \mathbf{v}_i - 2(\mathbf{v}_i \cdot \mathbf{n})\mathbf{n}. \tag{19}$$

The equation describes elastic collisions, unfortunately, collisions are not fully elastic in real world. In order to prevent it, we want to control how much of the kinetic force will be preserved. We also want to make the collisions dependent on the exact time of the collision in the time interval between t_i and t_{i+1}. To do this, we use impulse based collision like in [Kelager 2006]. The velocity of the particle is updated to

$$\mathbf{v}_i = \mathbf{v}_i - (1 + c_R \frac{d}{\Delta t \|\mathbf{v}_i\|})(\mathbf{v}_i \cdot \mathbf{n})\mathbf{n}, \tag{20}$$

where $0 \leq c_R \leq 1$ is a constant controling preservation of kinetic energy and d is the penetration depth. Value $c_R = 0.45$ seems to work well. The main drawback of this method is a relatively slow calculation compared to the approach based on distances.

3.2   Distance Method

Our approach is inspired by [Harada et al. 2007a]. If the distance of a particle from the surface is smaller than d then we presume that a collision with the surface can occur. We used $d = h$, where h is support radius of the kernel functions, in our simulation. The collision will occur only when the particle is moving toward the surface. So we have to test if $\mathbf{v}_i \cdot \mathbf{n}(\mathbf{r}_i) < 0$. If this test is positive we handle the collision by reflecting the velocity of the particle along the surface normal (Fig. 1), where we use greater angle of reflection than the angle of incidence. We do the response before the collision actually happens (as soon as the distance is smaller the d).
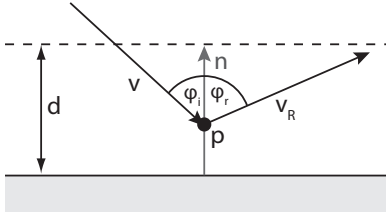

Figure 1: Collision handling in the distance method. Velocity v of particle p is reflected along the surface normal, where $\phi_i \neq \phi_r$, because it is moving towards surface.

Model used in the simulation does not change in time, thus we can use this fact to accelerate the computation of the distance. Surface of the model is represented by a mesh. Thus the distance of a particle from the surface corresponds to the distance of the particle to the closest face of the mesh. We precompute a regular grid of pointers to nearest faces. If the distance is above a certain threshold the pointer is NULL. When computing distance of a particle from surface we find the closest pointer in the grid. If the pointer is NULL, we do not have to handle a collision. If pointer is not NULL, we compute the distance of the particle to the pointed face.

If gravitation or pressure force is pushing a particle towards the surface, in some cases, reflecting of the velocity is not enough to get the particle farther from the surface than d. Such particle will gradually come closer to the surface and eventually go through it. To avoid this problem, we use a force that pushes particles away from the surface, if they come closer to the surface than d. We apply this force by increasing velocity in the direction of surface normal.

$$\mathbf{v} = \mathbf{v} + c(2d - dist)\mathbf{n}, \tag{21}$$

where $\mathbf{v}$ is the velocity of a particle, c is a user defined constant and $dist$ is the distance of a particle from the surface.

4   Visualisation

Fluid surface reconstruction is done using marching cube algorithm, where density is used as isofunction. Obtained surface is then fitted to the model like in [Harada et al. 2007a]. When a vertex

of the fluid surface is closer to the surface than user defined distance d_s, it is moved to the surface. It can be described as

$$\bar{\mathbf{x}} = \mathbf{x} - d(\mathbf{x})\mathbf{n}(\mathbf{x}), \tag{22}$$

where $\bar{\mathbf{x}}$ is a fitted vertex, $\mathbf{x}$ is the vertex before fitting, $d(\mathbf{x})$ is the distance of thevertex $\mathbf{x}$ from the surface, and $\mathbf{n}(\mathbf{x})$ is the normal in the closest point of the model surface from the vertex $\mathbf{x}$. We use $d_s = h$, where h is the support radius of the the kernel functions used in SPH. The acceleration method of distance finding from the collision handling is used here to accelerate the surface fitting. The results are than exported into a COLLADA file. The file is imported into 3ds Max and rendered.

5   Results

We implemented described method on a PC with Intel P9400 2.4 GHz processor with 4.0GB RAM, but we did not fully exploited the multicore potential. We tested the presented methods with two scenarios. Collision handling based on distances produces similar visual results as the method based on intersections, but with a higher speed.

scene	number of particles	distance	intersections	time of the precomputation
cuboid	5510	17-20 fps	5-6.5 fps	5s
city	8892	10-13 fps	2-3.5 fps	2m 18s

Table 1: Comparison of frames per second calculated using the collision handling based on distances and the collision handling based on intersections.

OpenSceneGraph was used for importing of the 3D models used in collision handling. Best result were achieved for 3DS and OBJ files. GNU triangulated surfaces library was used in surface reconstruction and COLLADA DOM was used for exporting the water surface and geometry of the model into a COLLADA file.

6   Conclusions and Future Work

In this paper, we presented a method of water simulation, which can be used in flood simulations. To do that, we used the SPH particle method. We implemented collision handling using intersections, as well as approach based on the distance from the surface. We showed that the collision handling based on the distance from the surface produces visually realistic results and runs faster than collision handling based on intersections.

We intend to further accelerate the simulation by using adaptive particle size like in [Adams et al. 2007] and so reduce the number of particles needed in the simulation. We want also implement a faster neighbour search algorithm as proposed in [Onderik and Durikovic 2008].

References

ADAMS, B., PAULY, M., KEISER, R., AND GUIBAS, L. J. 2007. Adaptively sampled particle fluids. *ACM Transactions on Graphics 26*, 3 (July), 48:1–48:7.
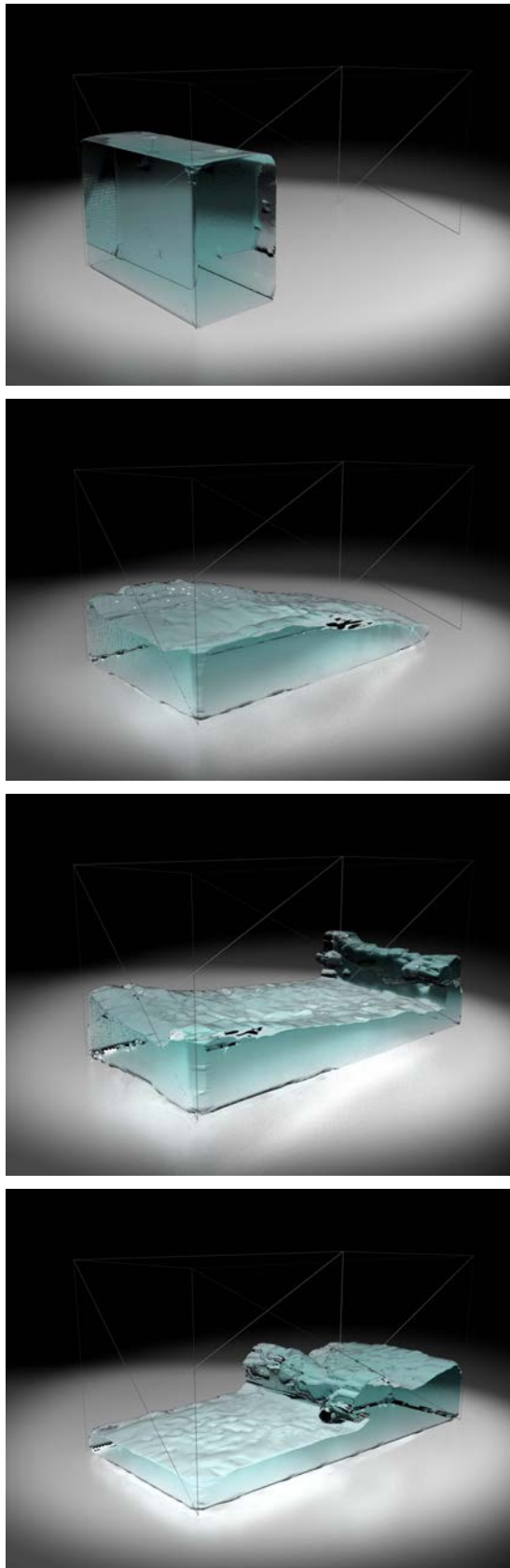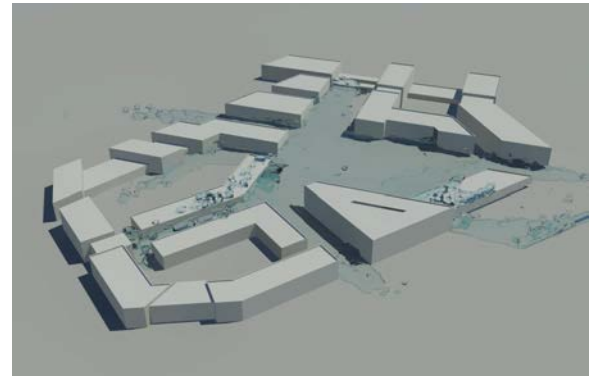
Figure 3: Flood simulation of a city rendered in 3ds Max.

DESBRUN, M., AND GASCUEL, M.-P. 1996. Smoothed particles: A new paradigm for animating highly deformable bodies. In *In Computer Animation and Simulation 96 (Proceedings of EG Workshop on Animation and Simulation)*, Springer-Verlag, 61–76.

HARADA, T., KOSHIZUKA, S., AND KAWAGUCHI, Y. 2007. Smoothed particle hydrodynamics in complex shapes. In *Proceedings of the 23st spring conference on Computer graphics*.

HARADA, T., KOSHIZUKA, S., AND KAWAGUCHI, Y. 2007. Smoothed particle hydrodynamics on GPUs. In *Proceedings of Computer Graphics International*, 63–70.

JONES, M. W. 1995. 3D distance from a point to a triangle. Tech. rep., Department of Computer Science, University of Wales.

KELAGER, M. 2006. *Lagrangian Fluid Dynamics Using Smoothed Particle Hydrodynamics*. Master's thesis, Department of Computer Science, University of Copenhagen, Copenhagen.

KOSHIZUKA, S., AND OKA, Y. 1996. Moving-particle semi-implicit method for fragmentation of incompressible fluids. *Nucl. Sci. Eng. 126*, 421–434.

MONAGHAN, J. 1992. Smoothed particle hydrodynamics. *Annual review of astronomy and astrophysics 30*, 543–574.

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.

ONDERIK, J., AND DURIKOVIC, R. 2008. Efficient neighbor search for particle-based fluids. *Journal of the Applied Mathematics, Statistics and Informatics 4*, 29–43.

SOLENTHALER, B., AND PAJAROLA, R. 2009. Predictive-corrective incompressible sph. In *SIGGRAPH '09: ACM SIGGRAPH 2009 papers*, ACM, New York, NY, USA, 1–6.

SOLENTHALER, B., SCHLFLI, J., AND PAJAROLA, R. 2007. Pajarola r.: A unified particle model for fluid solid interactions: Research articles. *Comput. Animat. Virtual Worlds 18*, 69–82.

Figure 2: Fluid surface rendered in 3ds Max.