Journal of the Applied Mathematics, Statistics and Informatics (JAMSI), **4** (2008), No. 1

Efficient Neighbor Search for Particle-based Fluids

JURAJ ONDERIK AND ROMAN ĎURIKOVIČ

Abstract

Lagrangian particle-based animation is a popular strategy for simulating complex phenomena as fluids. Due to its inherent mesh-less nature the set of neighbor particles within a specified range must be efficiently found.

In this paper we propose *Cell Indexing* a novel approach for searching approximate neighbor particles necessary for efficient fluid simulation using SPH. Instead of storing particles into a fixed 3D grid or a hash map, we encode their index and coordinates into a *key*. The list of keys is then sorted using linear time radix sort. A simple traversal using H-mask can quickly accumulate approximate neighbors without problematic cache misses of Spatial Hashing, large memory requirements of full 3D grids or $O(n \log n)$ time complexity of kd-trees. Furthermore we can achieve sub-cell precision by using larger H-masks, while having only constant factor slowdown. Using H-mask can substantially increase the precision of Spatial Hashing or 3D grids, however more cache misses or larger memory requirements arise.

We have demonstrated our approach within a standard SPH based fluid simulation.

Mathematics Subject Classification 2000: I.3.5, I.3.7
Additional Key Words and Phrases: Neighbor Search, Cell Indexing, H-Mask, Multi-Phase Smoothed Particle Hydrodynamics

1. INTRODUCTION

Physically based animation of complex natural phenomena is an attractive topic among the computer graphics research. Simulation methods for rigid and deformable solids are being coupled with various fluid simulation strategies. For both Eulerian [13; 4; 24] and Lagrangian approaches [20; 31] unified solid-fluid simulation techniques has been proposed. However, still a number of issues related to stability, accuracy, realistic boundary conditions, performance etc. arise. Recent graphics hardware allows huge parallelization of many time consuming problems, thus simulation algorithms has to be adapted.

Generally fluid simulation techniques solving full 3D Navier-Stokes equations can be categorized as *Eulerian* and *Lagrangian*. In Eulerian approaches governing equations are evaluated on a fixed mesh (usually a 3D grid), whereas Lagrangian methods can be both *mesh-based* or *mesh-less*. In both Lagrangian strategies mesh or particles are not fixed to the domain, but are advected with the fluid. This can simplify governing equations, see section (3) and allows virtually unlimited simulation space. Beside these advantages, it is usually complex to extract smooth boundary representation of interfaces, correctly handle interface tension in small features as bubbles and droplets, perform complex remeshing (mesh-based techniques) or alternatively efficiently find neighbor (closest) particle pairs (mesh-less techniques). Beside full Navier-Stokes simulations, several "shallow water" techniques exists, where fluid interface is simulated using wave equations. However, in combination with Navier-Stokes equation they can achieve quite realistic results.[36; 17]

In this paper we focus on Lagrangian particle-base simulation of viscous fluids using *Smoothed Particle Hydrodynamics* (SPH), see section (3). It is a mesh-less approach, where all physical quantities are sampled on particle locations. The influence of each particle is only local[1] thus it is essential to find the set of neighbor particles. Since this highly affects further calculations of fluid dynamics it becomes usually the bottleneck of the overall animation. In section (4) we have extended the well known *Spatial Hashing* for varying particles support length, see subsection (4.1) and developed a novel neighbor search algorithm *Cell Indexing*, see subsection (4.2), fixing several issues of previous methods.

2. RELATED WORK

It is beyond the scope of this article to give an extensive overview of the fluid simulation problematics, therefore we focus here only to related Eulerian and mesh-less Lagrangian works.

2.1 Eulerian Grid-based Methods

Since introducing to graphics community the full 3D simulation of Navier-Stokes equations, intensive research has been done to improve their the famous Eulerian-based *MAC-grid* method [12].[2] Later Stam proposed the popular extension of basic MAC by using a semi-Lagrangian integration scheme and iterative solver of the pressure equation.[32] Fedkiw et al. developed a *Particle Level Set* method for accurate interface tracking.[11; 10]

Advanced simulation of melting and flowing of highly viscous, non-Newtonian fluids were presented in [4]. Later Carlson et al. used distributed Lagrangian multipliers [5] for animating the interplay between rigid bodies and viscous incompressible fluids. Direct two way coupling between mesh-based thin solids and Eulerian-based fluids was done in [13; 24].

Hong and Kim [16] use a *Volume Of Fluid* (VOF) indicator function to simulate a two-phase fluid flow and bubbles with surface tension forces. Losasso et al. extended the particle Level Set Method for interface tracking of multiple interacting fluids [25] and presented a cutting-edge fluid simulator. A generalized solution to modeling hydraulic erosion is presented by Beneš et al. in [3]. Recently Chentanez [6] developed a method for animating incompressible liquids with detailed free surfaces using Lattice-Based Tetrahedral Meshes.

Octrees has been used to speed-up grid-based simulations [23]. Further combing 2D height-fields with a full 3D Navier-Stokes simulation near the interface allowed efficient animation of large bodies of water.[17]. Recently controlling fluid motion while preserving details was successfully applied to both grid-based *Lattice-Boltzmann Method* (LBM) and Lagrangian *Smoothed Particle Hydrodynamics* (SPH)[35].

[1]Within a support distance from particles location

[2]Nevertheless a full 3D fluid simulation in CFD engineering has been already well established

2.2 Lagrangian Particle-based Methods

Particle-based simulations were introduced to graphics community by Reeves. These animations of simple fountains and sprays were presented by uncoupled particles, which is insufficient for realistic fluid simulation.

Later Desbrun et al. [9] used Smoothed Particle Hydrodynamics[3] (SPH) to couple particles and simulate highly deformable bodies including viscous fluids. Müller et al. further popularize SPH technique by animating a pouring water into a glass at interactive rates [27]. By averaging viscosity, they came with a simple model of multi-phase fluids. [28].

Since computing pressure forces only locally andusing explicit integration, their model is suitable only for compressible and near incompressible fluids. For stiff fluids incompressibility is achieved by taking very small time steps.[4] Premože et al. gained incompressibility by solving the pressure globally (using Poisson equation) with an iterative linear equation solver. [29] This technique is referred as *Moving Particle Semi-implicit* (MPS).

Clavet et al. describe a SPH based interactive technique for simulating visco-elastic and plastic fluids by connecting particles with temporary springs [7]. Unified particle-based SPH model for coupling solids and fluids has been presented in [20] and later in [31]. Recently a successful approach of coupling SPH and Particle Level Set has been done by Losasso.[22]. Promising results are shown by Harada when moving the entire SPH calculation to GPU and precalculate the influence of solid boundary particles.[5]. Their were simulating 60k particles at nearly interactive rates (17 fps)[14; 15].

Further simplifications has been proposed by Adams et. al [1] by adaptively sampling particles and allowing them to have various sizes. For sparse particles Kipfer et al. developed a novel technique for computing neighbor particles [21].

3. MULTI-PHASE SMOOTHED PARTICLE HYDRODYNAMICS

Inspired by Monte-Carlo integration *Smoothed Particle Hydrodynamics* (SPH) was initially developed by Lucy and Monaghan for simulating flow of interstellar gas within Astrophysics[26]. Unlike traditional mesh-based methods (FEM, FDM) where physical quantities are evaluated on a fixed Mesh (grid), SPH belongs to mesh-less methods, where calculated values are approximated from neighboring points. Particles have no fixed topology, thus are suitable for complex dynamic phenomena as fluids. Due to the particle-based Lagrangian nature of SPH, mass conservation is trivially satisfied and convection of the substance is inherent.

3.1 Smoothing with Particles

Given a set of particles (interpolation points) $\mathbf{r}_i$ carrying values $A_i = A(\mathbf{r}_i)$ of field quantity A we can approximate $A(\mathbf{r})$ at arbitrary point $\mathbf{r}$ by a convolution with a

[3]originally developed by Lucy and Monaghan

[4]Mainly due to stability issues

[5]Commonly solid-fluid boundary conditions are solved by fixing particles to the solid volume near the surface. This increase the number of particles and lower down performance

radial symmetric *smoothing kernel* $W(\mathbf{r},h)$ as [26; 28]

$$A(\mathbf{r}) \approx A \star W = \int_{\mathbf{r}'} A(\mathbf{r}')W(|\mathbf{r}-\mathbf{r}'|,h)d\mathbf{r}' \approx \sum_j V_j A_j W(\mathbf{r}-\mathbf{r}_j,h) \tag{1}$$

and further replaced by summation over neighbor particles having finite *volume* $V_j = m_j/\rho_j$. Mass of particles is a constant property, thus volume is defined as the ratio between *mass* m_j and *density* ρ_j. The convolution smoothing kernel must have the following properties

$$\int_{\mathbf{r}} W(\mathbf{r},h)d\mathbf{r} = 1 \quad \wedge \quad \lim_{h\to 0} W(\mathbf{r},h) = \delta(\mathbf{r}) \tag{2}$$

where h is the *support length* of the kernel, i.e. the distance to which particle affects other particles. Notice in limit case $h \leftarrow 0$ kernel $W(\mathbf{r},h)$ must be the Dirac delta function $\delta(\mathbf{r})$. Making the kernel second order differentiable we can further express gradient $\nabla A(\mathbf{r})$ and laplacian $\nabla^2 A(\mathbf{r})$ of field function $A(\mathbf{r})$ at arbitrary point as

$$\begin{aligned} A(\mathbf{r}) &= \textstyle\sum_j V_j A_j W(\mathbf{r}-\mathbf{r}_j,h) \\ \nabla A(\mathbf{r}) &= \textstyle\sum_j V_j A_j \nabla W(\mathbf{r}-\mathbf{r}_j,h) \\ \nabla^2 A(\mathbf{r}) &= \textstyle\sum_j V_j A_j \nabla^2 W(\mathbf{r}-\mathbf{r}_j,h) \end{aligned} \tag{3}$$

This properties greatly simplify further calculations of Lagrangian-based fluid dynamics.

3.2 Evaluating Fluid Properties using SPH

The well known *Navier-Stokes* governing equations of a simple Newtonian isothermal and incompressible fluid can by expressed in Eulerian form by two conservation laws, i.e. the conservation of mass (*continuity equation*) and the conservation of momentum (*momentum equation*)

$$\begin{aligned} \frac{\partial \rho}{\partial t} &= -\rho \nabla \cdot \mathbf{v} && \text{(continuity eq.)} \\ \rho\left(\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v}\cdot\nabla\mathbf{v}\right) &= -\nabla p + \mu \nabla^2 \mathbf{v} + \rho \mathbf{g} && \text{(momentum eq.)} \end{aligned} \tag{4}$$

Lagrangian (particle based) formulation can be obtained using the material derivative $\frac{D\mathbf{q}}{Dt} = \frac{\partial \mathbf{q}}{\partial t} + \mathbf{v}\cdot\nabla\mathbf{q}$.[6] as

$$\begin{aligned} \frac{D\rho}{Dt} &= -\rho \nabla \cdot \mathbf{v} \\ \rho\frac{D\mathbf{v}}{Dt} &= -\nabla P + \mu\nabla^2\mathbf{v} + \rho\mathbf{g} \quad (= \mathbf{F}^{\text{press}} + \mathbf{F}^{\text{visco}} + \mathbf{F}^{\text{ext}}) \end{aligned} \tag{5}$$

Assuming constant mass of all particles total mass of fluid is always conserved. we can thus completely omit continuity equation from further calculations. Viewing the momentum equation as Newton's second law ($\mathbf{f} = m\mathbf{a}$), i.e. we only need to

[6]Since particles move with the fluid the convective term $\mathbf{v}\cdot\nabla\mathbf{q}$ is not present

calculate forces acting on particles. Notice the right hand side of the momentum equation can be thus expressed as a sum of *pressure* $\mathbf{F}^{\text{press}}$, *viscosity* $\mathbf{F}^{\text{visco}}$ and *external* $\mathbf{F}^{\text{ext}}$ force fields. This external force field further contains all other force fields acting on fluid (e.g. interface tension $\mathbf{F}^{\text{int}}$ or gravity $\mathbf{F}^{\text{grav}}$)

Assuming a finite volume of particles, we get the relation between total force $\mathbf{f}_i$ acting on i-th particle and respective force field, by integrating the force field over the particles volume, setting $\mathbf{F}_i = \mathbf{F}(\mathbf{r}_i)$ and using SPH approximation, see equation (3) as

$$\mathbf{f}_i = \int_{\mathbf{r}} \mathbf{F}(\mathbf{r}) d\mathbf{r} \approx \int_{\mathbf{r}} V_i \mathbf{F}_i W(\mathbf{r} - \mathbf{r}_j, h) = V_i \mathbf{F}_i \int_{\mathbf{r}} W(\mathbf{r} - \mathbf{r}_i, h) = V_i \mathbf{F}_i \qquad (6)$$

Fluid Properties. Assuming $\rho_j = m_j / V_j$ density can be approximated using SPH as well. Further pressure P_i occurred at i-th particle is described by Tait's equation [2][7]

$$\rho(\mathbf{r}_i) = \sum_j m_j W(\mathbf{r}_i - \mathbf{r}_j, h) \qquad P_i = k^{\text{gas}} \left(\left(\frac{\rho}{\rho_0} \right)^{\gamma} - 1 \right) \qquad (7)$$

Using blindly the SPH approximation for pressure force $\mathbf{f}_i^{\text{press}}$ and multiphase viscosity force $\mathbf{f}_i^{\text{visco}}$ violates the action-reaction principle, since the forces are not symmetric ($\mathbf{f}_i \neq -\mathbf{f}_j$). Various symmetrization approaches exists among the SPH literature (see [30; 26]), however we adapt here the simplified formulation by Müller [28],[8] where the pressure and viscosity is only averaged.[9]

$$\begin{aligned}
\mathbf{f}_i^{\text{press}} &= V_i \mathbf{F}_i^{\text{press}} = -\sum_j V_i V_j \frac{P_i + P_j}{2} \nabla W^{\text{press}}(\mathbf{r}_i - \mathbf{r}_j, h) && (a) \\
\mathbf{f}_i^{\text{visco}} &= V_i \mathbf{F}_i^{\text{visco}} = \sum_j V_i V_j \frac{\mu_i + \mu_j}{2} (v_j - v_i) \nabla^2 W^{\text{visco}}(\mathbf{r}_i - \mathbf{r}_j, h) && (b) \\
C_i^{\text{int}} &= \sum_j V_j c_j^{\text{int}} \nabla^2 W^{\text{poly}}(\mathbf{r}_i - \mathbf{r}_j, h) && (c)
\end{aligned} \qquad (8)$$

Notice particle volume $V_i = m_i / \rho_i$ uses approximate density ρ_i computed in equation (7).

Interface tension force $\mathbf{f}_i^{\text{int}}$ acts along the normalized gradient $\mathbf{n} = \frac{\nabla C_i^{\text{int}}}{|\nabla C_i^{\text{int}}|}$ of the interface color field $C^{\text{int}}(\mathbf{r})$ and is proportional to its curvature $\kappa_i = \nabla^2 C_i^{\text{int}}$. Interface color field values C_i^{int} at particle locations are approximated from particles interface color values c_i^{int} using SPH, see equation (8) (c). Interface tension force is thus defined as

$$\mathbf{f}_i^{\text{int}} = -\sigma^{\text{int}} \kappa_i \mathbf{n}_i = -\sigma^{\text{int}} \nabla^2 C_i^{\text{int}} \frac{\nabla C_i^{\text{int}}}{|\nabla C_i^{\text{int}}|} \qquad (9)$$

[7]An simpler alternative is the ideal gas equation $P_i = k(\rho - \rho_0)$
[8]See [9] for alternative formulations
[9]This gives more stable simulation, with the cost of some energy dissipation

Smoothing Kernels. In our computations we use the following smoothing kernels adapted from [27] and its derivatives (see appendix (6)).

$$\begin{aligned}
W^{\text{poly}}(\mathbf{r},h) &= \frac{315}{64\pi h^9}\begin{cases}(h^2-r^2)^3 & 0\le r\le h \;\wedge\; r=|\mathbf{r}|\\ 0 & \text{otherwise}\end{cases}\\
W^{\text{press}}(\mathbf{r},h) &= \frac{15}{\pi h^6}\begin{cases}(h-r)^3 & 0\le r\le h \;\wedge\; r=|\mathbf{r}|\\ 0 & \text{otherwise}\end{cases}\\
W^{\text{visco}}(\mathbf{r},h) &= \frac{15}{2\pi h^6}\begin{cases}-\frac{r^3}{2h^3}+\frac{r^2}{h^2}+\frac{h}{2r}-1 & 0\le r\le h \;\wedge\; r=|\mathbf{r}|\\ 0 & \text{otherwise}\end{cases}
\end{aligned} \tag{10}$$

3.3 Simulation Overview

Our simulation loop contains three main steps, namely the *Neighbor Search*, *Force Computations* and the *Time Integration.* We use explicit time integration, thus the overall simulation can be categorized as explicit and force-based, see algorithm (1).

```
In: support length $h$, subdivision factor $H$ and delta time $\Delta t$
function SPH($h, H, \Delta t$)
1:  NEIGHBOURS ← SEARCHNEIGHBORS($h, H$)
2:  foreach $\mathcal{P}_i$ in PARTICLES do
3:      $\rho_i \leftarrow 0$;  $\nabla C_i \leftarrow \mathbf{0}$;  $\nabla^2 C_i \leftarrow 0$;  $\mathbf{f}_i \leftarrow \mathbf{f}_i^{\text{ext}}$        /* initialize */
4:      foreach $\mathcal{P}_j$ in NEIGHBOURS($\mathcal{P}_i$) do    /* accumulate density */
5:          $\rho_i \leftarrow \rho_i + m_j W^{\text{poly}}(\mathbf{r}_i - \mathbf{r}_j, h)$
6:      end
7:      $P_i \leftarrow k^{\text{gas}}\left(\left(\frac{\rho}{\rho_0}\right)^\gamma - 1\right)$                /* calculate pressure */
8:      foreach $\mathcal{P}_j$ in NEIGHBOURS($\mathcal{P}_i$) do     /* accumulate forces */
9:          $\mathbf{f}_i \leftarrow \mathbf{f}_i - V_i V_j \frac{P_i+P_j}{2} \nabla W^{\text{press}}(\mathbf{r}_i - \mathbf{r}_j, h)$          /* $(= \mathbf{f}_i^{\text{press}})$ */
10:         $\mathbf{f}_i \leftarrow \mathbf{f}_i + V_i V_j \frac{\mu_i+\mu_j}{2}(v_j - v_i)\nabla^2 W^{\text{visco}}(\mathbf{r}_i - \mathbf{r}_j, h)$ /* $(= \mathbf{f}_i^{\text{visco}})$ */
11:         $\nabla C_i \leftarrow \nabla C_i + V_j c_j^{\text{int}} \nabla W^{\text{poly}}(\mathbf{r}_i - \mathbf{r}_j, h)$         /* $(= \nabla C_i^{\text{int}})$ */
12:         $\nabla^2 C_i \leftarrow \nabla^2 C_i + V_j c_j^{\text{int}} \nabla^2 W^{\text{poly}}(\mathbf{r}_i - \mathbf{r}_j, h)$     /* $(= \nabla^2 C_i^{\text{int}})$ */
13:     end
14:     $\mathbf{f}_i \leftarrow \mathbf{f}_i - \sigma^{\text{int}} \nabla^2 C_i^{\text{int}} \frac{\nabla C_i^{\text{int}}}{|\nabla C_i^{\text{int}}|}$                    /* $(= \mathbf{f}_i^{\text{int}})$ */
15: end
16: foreach $\mathcal{P}_i$ in PARTICLES do                              /* Leap-Frog */
17:     $\mathbf{v}_i \leftarrow \mathbf{v}_i + \Delta t \frac{\mathbf{f}_i}{m_i}$
18:     $\mathbf{r}_i \leftarrow \mathbf{r}_i + \Delta t \mathbf{v}_i$
19: end
end
```

Algorithm 1: Our fluid simulation step using SPH

As shown in algorithm (1), searching for neighbor particles (i.e. building the list of close particle pairs) is done in SEARCHNEIGHBORS(h, H), where h is support length and H is the subdivision factor, see section (4). Next the force computation is done by iterating over all particles (lines 2-15). First density ρ_i is accumulated from neighbors (lines 4-6), then pressure P_i is calculated (line 7) and finally total force acting on particle is accumulated (lines 8-14). Time integration is performed using standard second order "Leap-frog" scheme (lines 16-19). This looks similar to the simple first-order explicit Euler scheme, however when the velocity is initialized[10] correctly as $\mathbf{v}_i = \mathbf{v}_i(t_0 - \frac{1}{2}\Delta t)$ the code perfectly mimic the following "Leap-frog" update rules (see equation (11))

$$\begin{aligned} \mathbf{v}_i(t + \frac{1}{2}\Delta t) &\leftarrow \mathbf{v}_i(t - \frac{1}{2}\Delta t) + \Delta t \frac{\mathbf{f}_i(t)}{m_i} \\ \mathbf{r}_i(t + \Delta t) &\leftarrow \mathbf{r}_i(t - \Delta t) + \Delta t \mathbf{v}_i(t + \frac{1}{2}\Delta t) \end{aligned} \tag{11}$$

Due to the explicit nature of our method, only small time steps are allowed to maintain stability and accuracy. Formally, i-th particle can be integrated with a maximal time step Δt_i according to the *Courant condition* of convergence[11]

$$\Delta t_i \le \Delta t^c = \alpha \frac{h}{c} \qquad\qquad \Delta t_i = \min\left(\Delta t^c, \alpha \frac{h}{|\mathbf{v}_i|}\right) \tag{12}$$

where c[12] is speed of sound in the fluid and $\alpha \approx 0.3$ is the Courant number. Stability (for fast moving particles) can be further increased by choosing Δt_i as in equation (12).

Choosing the global time step as $\Delta t = \min_i\{\Delta t_i\}$ is safe, but obviously inefficient. Since usually only a few particles will need this minimal time step, we can rather integrate each particle using its own time step Δt_i. Similarly as Desbrun [9] to synchronize integration of particles we choose a user defined simulation frame time $\Delta t^{\max}$ and find smallest positive number n_i satisfying $\Delta t'_i = \Delta t^{\max}/2^{n_i} \le \Delta t_i$ and set particles time step to $\Delta t'_i$. When the time step changes, we must further correct the position of the particle as

$$\mathbf{r}_i \leftarrow \mathbf{r}_i + \frac{(\Delta t'_{i,new})^2 - (\Delta t'_{i,old})^2}{8m_i}\mathbf{f}_i \tag{13}$$

Boundary Conditions. In SPH context solid-fluid boundary conditions are usually modeled by simply attaching *ghost* particles[13] near the boundary inside the solid object. this is usually a natural choice for deformable and melting solids. However for pure rigid objects one can precompute the influence of ghost particles (with respect to the body frame) and store it into a distance field omitting further calculations, thus speeding up whole simulation [14].

Interface Extraction. Realistic looking visualizations can be achieved by extracting a smooth interface representation. Beside using Marching Cubes to extract the

[10]The initialization routine trivial and is omitted here
[11]Combating numerical errors time step should be even smaller.
[12]Material property, i.e. user defined constant
[13]For static objects their positions are not integrated

zero-level of the implicit density function $\rho(\mathbf{r}) = 0$ several attempts has been done. We refer interested reader to [1] for further details.

4. NEIGHBOR SEARCH

Similarly to n-body problem, searching for close particle pairs is a crucial problem in almost any particle-based fluid simulation. Since this usually becomes the bottleneck of the simulation time, efficient algorithms are necessary. In the SPH context each particle affects only a set of neighbor particles which lie within its support distance h. Formally we define for each particle p_i its *set of neighbor particle indices* $N_i(h)$ as

$$N_i(h) = \{\ j \quad | \quad |\mathbf{r}_i - \mathbf{r}_j| \leq h\} \tag{14}$$

Beside the naive and inefficient $O(n^2)$ "all-pair-test", several algorithms usually based on spatial subdivision and fast approximate neighbor search has been proposed [34; 21; 18; 8]. Here we extend the usual *Spatial Hashing*, see subsection (4.1), technique for faster SPH simulation and propose a novel approach *Cell Indexing*, see subsection (4.2), based on indexing non-empty cells in a virtual subdivision grid.

4.1 Extended Spatial Hashing

A common approach to optimize the "all-pair-test" is to subdivide the smallest enclosing axis aligned bounding box (AABB) into a 3D grid of cells with size h. For each particle with position $\mathbf{r}_i = (x, y, z)$ we use its relative coordinates to AABBs *minimal corner* $\mathbf{c}_{\min} = (x_{\min}, y_{\min}, z_{\min})$ and calculate its (positive) cell coordinates $\text{cell}(x, y, z, h) = (i, j, k)$, see equation (15). Into each cell (i, j, k) we could store indices of particles with the same cell coordinates. Depending on the AABB and cell size ratio, such voxelization obviously lead to huge memory consumption.[14]

Teschner et. al [34] overcome this problem by introducing *spatial hash function* $\text{hash}(i, j, k)$ which naturally maps particle cell coordinates to a bucket in a hash map. This enables grid size to be virtually unlimited and does not store empty cells. The hash function is defined as

$$\begin{aligned} \text{cell}(x, y, z, h) &= \left(\left\lfloor \frac{x - x_{\min}}{h} \right\rfloor, \left\lfloor \frac{y - y_{\min}}{h} \right\rfloor, \left\lfloor \frac{z - z_{\min}}{h} \right\rfloor \right) = (i, j, k) \\ \text{hash}(i, j, k) &= (i \cdot p_1 \ \text{xor}\ j \cdot p_2 \ \text{xor}\ k \cdot p_3) \mod M \end{aligned} \tag{15}$$

where $p_1 = 73856093$, $p_2 = 19349663$ and $p_3 = 83492791$ are large prime numbers and M is the size of the hash map. Using cell and hash functions, insertion of a particle is $O(1)$ thus building the data structure for approximate neighbor search is linear. To find neighbors $N_i(h)$ for i-th particle we need examine only particles from all (26 in 3D) surrounding cells. Assuming the particle distribution is approximately even, each cell contains only a constant number of particles and thus overall neighbor search is near linear.

However in the SPH framework particles can cluster[15]. This leads usually to larger bucket sizes[16] or slower hashing. To reduce this problem we can naturally

[14]storing empty cells is not optimal

[15]simulating compressible (near incompressible) fluids with strong pressures

[16]We assume all buckets have fixed size as speed-up

hash particles into smaller cells but then we need to examine particles from more neighboring cells.

Formally let $h' = h/(2H+1)$ be the subdivided cell size and H the *subdivision factor*. When calculating approximate neighbors of particle in cell (i, j, k) we need to examine particles from all $(i+s, j+t, k+u)$ cells, where $(s, t, u) \in \mathcal{M}_H(h)$. The set H-mask $\mathcal{M}_H(h)$ contains relative coordinates of all neighbor cells which are not completely outside the support of processed cell, see figure (1) (d)

$$\mathcal{M}_H(h) = \{(s, t, u) \mid |(h's, h't, h'u)| < h \quad \wedge \quad s, t, u \in \{-H, \cdots, +H\}\} \tag{16}$$

Finally notice, that larger masks[17] completely contains smaller masks, i.e. $G < H \Rightarrow \mathcal{M}_G(h) \subset \mathcal{M}_H(h)$. This allows simulations where particles have different support length. Indeed, assuming cell size is h' and particles support is h we select H-mask for this particle according to $H = \lfloor h/h' \rfloor + 1$[18].

Hashing naturally introduce cache misses, thus examining all $\mathcal{M}_H(h)$ can even slow down the process[19]. With larger H size of hash map (M) must increase to avoid numerous collisions. It can be estimated experimentally.

4.2 Cell Indexing

Inspired by the *staggered grids* [21] we have developed a novel approach for searching approximate neighbor particles in a linear time, trying to avoid several disadvantages of the spatial hashing. Similarly to other grid methods we first calculate the AABB of all particles and use further only their relative coordinates to AABBs minimal corner $\mathbf{c}_{\min}$. For each particle we define it's $\text{key}(n, i, j, k)$ as

$$\text{key}(n, i, j, k) = n + 2^I i + 2^{I+J} j + 2^{I+J+K} k \qquad \text{where} \quad \begin{array}{rcl} 2^I &>& N \\ 2^J &>& \lfloor B_x/h \rfloor \\ 2^K &>& \lfloor B_y/h \rfloor \end{array} \tag{17}$$

where n is index of particle, N is the number of particles, B_x, B_y and B_z are dimensions of bounding box, $(i, j, k) = \text{cell}(x, y, z)$ are cell coordinates and I, J, K and are arbitrary constants [20] depending on the size of the AABB and cell size h. Notice, that function $\text{key}(n, i, j, k)$ encode its parameters to a unique key. Thus given a key q we can compute unique (n, i, j, k).[21]

1D Case. Suppose a 1D case show in figure (1) (a), where the keys reduce to $\text{key}(n, i) = n + 2^I i$. Given two arbitrary particles m and n with keys $\text{key}(m, i_m) \leq \text{key}(n, i_n)$ their cell coordinates also satisfy $i_m \leq i_n$ and vice-versa. After sorting all keys, we can thus efficiently access particles with increasing cell coordinates. Given a key q index of associated particle is $m = q \mod 2^I$.

When building the neighbor set $N(h)$ for particle p which lies in cell i we need to examine only particles in cells[22] $i-1$, i, $i+1$. This can be simply achieved

[17]with larger subdivision factor H

[18]Naturally this ration must be reasonable small

[19]Storing all indices of H-neighbor cells leads to more memory usage and slowdown

[20]usually 16 or 32

[21]Particle index is $n = q \mod 2^I$, cell coordinates are $i = (q/2^I) \mod 2^J, \cdots$

[22]Assuming size of cell is h

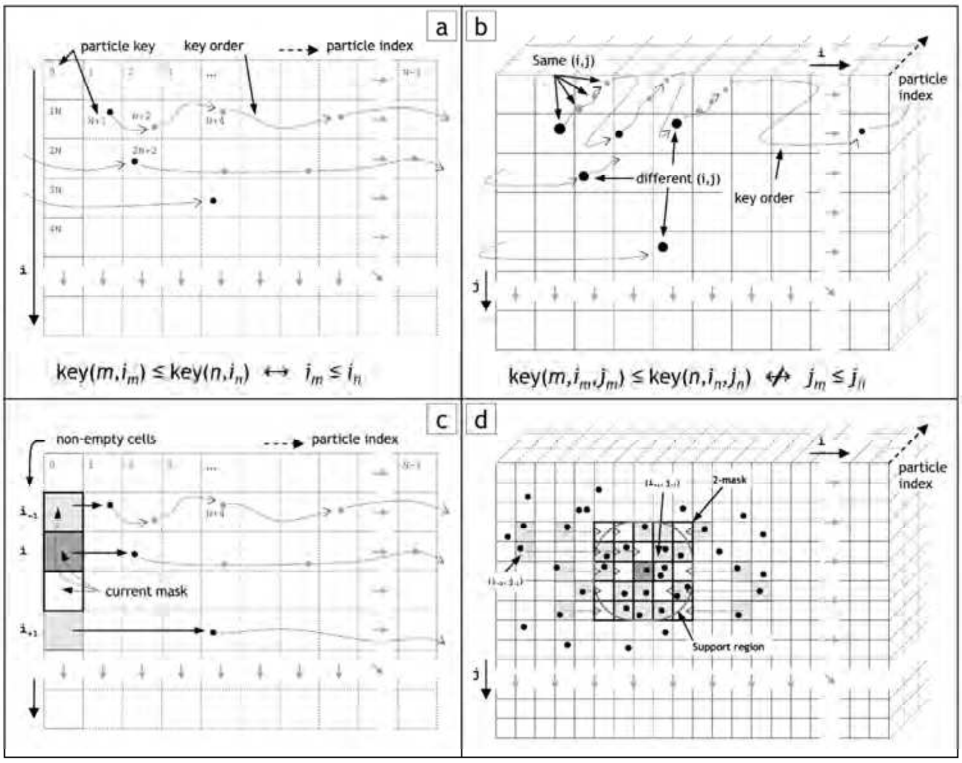

Fig. 1. *Cell Indexing principles. In (a) 1D case is shown. The left most column represents 1D space divided into cells. Dashed columns symbolize indices of particles. Each row thus contain all particles with equal i-coordinate. Dotted lines show the ordering of sorted keys. In (b) 2D extension is illustrated. Notice the third dimension depicts particle index, thus has no geometrical meaning. In (c) 1D searching principle is shown. The search follows key ordering, thus (in 1D) the i-coordinate as well. Simple 1-mask (cells with thick outline) is show and respective non-empty cells i_{-1}, i, i_{+1} (light-gray cells). In (d) H-mask is extended to 2D showing its non-empty cells and their corresponding location in the mask. Since the presented 2-mask is (in 2D) equal to the full 5x5 mask, the benefits of H-mask is not obvious from the picture, however for larger H (or in 3D) one can clearly see, that some "corner" cells of the full (2H+1)x(2H+1) square mask will be omitted in respective "spherical" H-mask.*

by storing indices $\mathcal{K}(i_{-1})$, $\mathcal{K}(i)$ and $\mathcal{K}(i_{+1})$ [23] of first[24] keys in *non-empty* cells $i_{-1} < i < i_{+1}$. We examine all keys (associated particles) in cell i_{-1} only when $i_{-1} = i - 1$ starting with key at index $\mathcal{K}(i_{-1})$. Similarly we examine cell i_{+1} only when $i_{+1} = i + 1$. We are processing particles in i-th cell until the respective key maps coordinates of this cell, i.e. for each key q must hold $\lfloor q/2^I \rfloor = \lfloor q_{\mathcal{K}(i)}/2^I \rfloor$.

Assuming the maximum number of particles inside one cell is bound, this traversal

[23] $\mathcal{K}(i)$ is index of key in list of all keys
[24] key in cell i with the smallest particle index

is $O(n)$. However when simulating particles with SPH this upper bound is *not* guaranteed, but fortunately in average case there are only few particles in one cell. To overcome particle clustering problem[25] we can choose the cell size $h' = h/(2H+1)$, where H is a *subdivision factor* depending on the fluid properties. When searching for neighbors we need now to examine cells $i_{-H} < i_{-H+1} < \cdots < i \cdots < i_{+H-1} < i_{+H}$ and thus store $2H+1$ key indices $\mathcal{K}(i_{-H}), \cdots \mathcal{K}(i), \cdots, \mathcal{K}(i_{+H})$, what slows down the traversal by a constant factor $O(H)$ per iteration. The algorithm thus stays linear $O(HN)$.

2D and 3D Case. The extension to 2D and 3D is not obvious, see figure (1) (b). In 2D when $\text{key}(m, i_m, j_m) \leq \text{key}(n, i_n, j_n)$ then $j_m \leq j_n$ while the order of i_m and i_n is non-decreasing only if $j_m = j_n$ (j-coordinates are equal). When calculating neighbors for particle in cell (i, j) we need to store all 9 key indices $\mathcal{K}(i_s, j_t) \mid s, t \in -1, 0, +1$ and examine non-empty cell (i_s, j_t) only when $(i_s, j_t) = (i + s, j + t)$. Furthermore in 3D we need 27 indices $\mathcal{K}(i_s, j_t, k_u) \mid s, t, u \in -1, 0, +1$ and examine non-empty cell (i_s, j_t, k_u) only when $(i_s, j_t, k_u) = (i + s, j + t, k + u)$. We could generalize this approach for cell subdivision taking $s, t, u \in -H, \cdots, +H$ or even better using H-mask where $(s, t, u) \in \mathcal{M}_H(h)$. The traversal is therefore in average case linear with complexity $O(|\mathcal{M}_H(h)|N)$.

Sorting Keys. To make the overall neighbor search algorithm linear, we must sort keys in linear time. As pointed by Terdiman [33] even a set of floating point numbers can be quickly sorted in $O(n)$ using radix sort. Since we have encoded indices of particles into their keys[26] we can use radix sort and have correct indices after sorting.

5. IMPLEMENTATION AND RESULTS

As shown in figure (2) three different simulations of the classical Dam-break test has been performed. Increasing the stiffness and surface tension coefficient, fluid behaves less compressible thus more realistic. In all three scenarios we used 1600 particles and set the rest density $\rho_o = 1000 kg/m^3$, mass $m_i = 0.0012 kg$, support length $h = 0.05m$ viscosity $\mu_i = 50 Ns/m^2$ and used a fixed time stepping with delta time $\Delta t = 0.002s$. The surface tension coefficient σ varied from 0.6 (top), 1.4 (mid) to 2.2 (bottom). We set stiffness k^{gas} to $20 Nm/g$ (top), $40 Nm/g$ (mid) and $70 Nm/g$ (bottom). Boundary conditions were handled by inelastic particle-plane collision resolution with wet friction. Simple particle-to-plane projection is used to prevent overshooting. State variables were integrated using explicit "Leap-Frog" integration scheme with fixed time stepping. The simulation has been performed on Mobile P4 1.7 GHz with GeForce 4 Go.

To test our approach we run each test case with three different neighbor search methods, namely the naive $O(n^2)$ "all-pairs-test", *Spatial Hashing* and our *Cell Indexing.*[27] and measured the average time spent for searching neighbors and integration of one time step. These results are summarized in table (I).

[25]Too many particles in one cell
[26]Keys are usually 64-bit numbers, 16 bits for each (n, i, j, k) component
[27]Other (tree) methods are left a the subject of future work

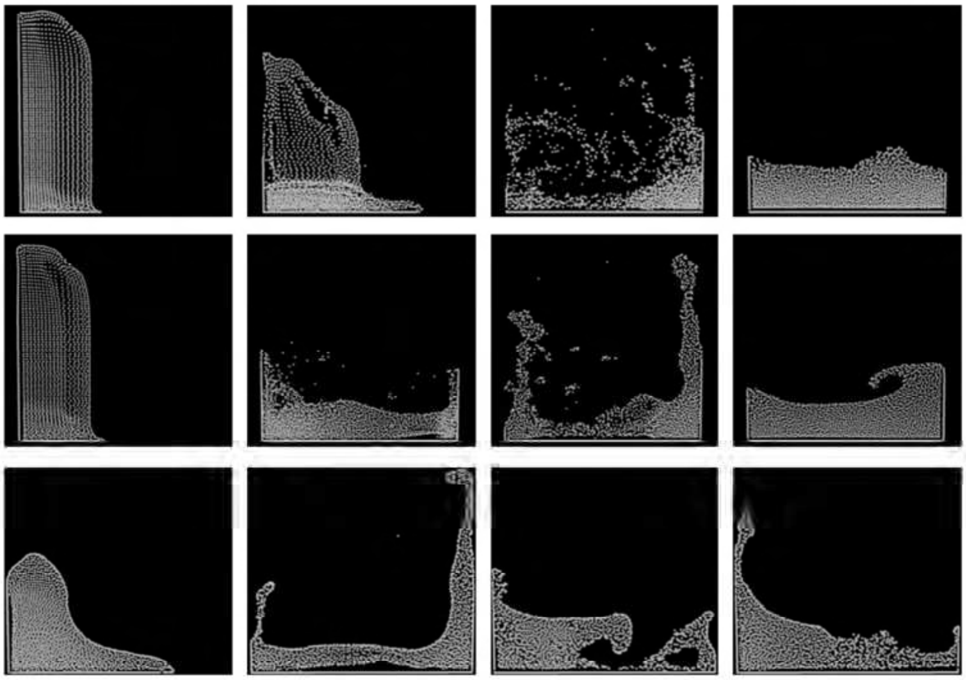

Fig. 2. *The classical Dam-break test within our SPH simulation environment. The three different test simulations demonstrate various fluid behavior with respect to changing stiffness and surface tension parameters. Top row describes the behavior of more compressible fluid with less attraction between particles, middle row corresponds to higher constants and finally the last row represents a stiffer fluid with stronger surface tension. Notice the snapshots are not taken in equal time steps.*

	All-Pairs	**Spatial Hashing**	**Cell Indexing**	**Dynamics**
Test Case 1	1,95 s	0,42 s	0,25 s	0,25 s
Test Case 2	1,97 s	0,45 s	0,27 s	0,24 s
Test Case 3	1,93 s	0,43 s	0,27 s	0,28 s

Table 1. Dam-break tests. 1600 Particles. Average time spent in one simulation time step in each test case.

6. CONCLUSION AND FUTURE WORK

We have proposed and demonstrated Cell Indexing as a novel approach for searching approximate neighbor particles within a SPH base fluid simulation. Due to cache misses, Spatial Hashing has been slightly outperformed by our method, without large memory requirements of Full 3D voxelization. Our approach is inherently linear, thus will outperform hierarchical methods[28] for larger data sets. We have introduced a H-mask to achieve speed-up by searching on a sub-cell resolution.

[28]They usually need $O(n \log n)$ to rebuild.

This seamlessly fits into Cell Indexing a extends even Spatial Hashing.[29]

Due to the immense computational power of current graphics hardware, we will investigate in the future the possibility of implementing our approach on GPU. Further we will try to involve spatial and temporal coherence into our method and allow it to use multi-resolutional fluid as been done in[19] The inherent compressibility of SPH can be decreased by solving the pressure implicitly. Therefore we will explore methods as MPS[29] and solve the pressure equation iteratively.

Appendix - Smoothing Kernel Derivatives

$$\begin{aligned}
\nabla W^{\text{poly}}(\mathbf{r},h) &= -\frac{945}{32\pi h^9}\begin{cases}(h^2-r^2)^2\mathbf{r} & 0\le r\le h \;\wedge\; r=|\mathbf{r}|\\ 0 & \text{otherwise}\end{cases}\\
\nabla^2 W^{\text{poly}}(\mathbf{r},h) &= \frac{315}{64\pi h^9}\begin{cases}(h^2-r^2)(7r^2-3h^2) & 0\le r\le h \;\wedge\; r=|\mathbf{r}|\\ 0 & \text{otherwise}\end{cases}\\
\nabla W^{\text{press}}(\mathbf{r},h) &= -\frac{45}{\pi h^6}\begin{cases}(h-r)^2\frac{\mathbf{r}}{r} & 0< r\le h \;\wedge\; r=|\mathbf{r}|\\ 0 & \text{otherwise}\end{cases}\\
\nabla^2 W^{\text{visco}}(\mathbf{r},h) &= \frac{45}{\pi h^6}\begin{cases}(h-r) & 0\le r\le h \;\wedge\; r=|\mathbf{r}|\\ 0 & \text{otherwise}\end{cases}
\end{aligned} \tag{18}$$

REFERENCES

Adams, B., Pauly, M., Keiser, R., and Guibas, L. J. Adaptively sampled particle fluids. In *ACM Transactions on Graphics (SIGGRAPH '07 papers)* (New York, NY, USA, 2007), vol. 26, ACM Press, pp. 48–48*.

Becker, M., and Teschner, M. Weakly compressible sph for free surface flows. In *SCA '07: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2007), Eurographics Association, pp. 209–217.

Benes, B., Tesínský, V., Hornys, J., and Bhatia, S. K. Hydraulic erosion. *Journal of Visualization and Computer Animation 17*, 2 (2006), 99–108.

Carlson, M., Mucha, P. J., R. Brooks Van Horn, I., and Turk, G. Melting and flowing. 167–174.

Carlson, M., Mucha, P. J., and Turk, G. Rigid fluid: animating the interplay between rigid bodies and fluid. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers* (New York, NY, USA, 2004), ACM Press, pp. 377–384.

Chentanez, N. Liquid simulation on lattice-based tetrahedral meshes. In *SIGGRAPH '07: ACM SIGGRAPH 2007 computer animation festival* (New York, NY, USA, 2007), ACM Press, p. 89.

Clavet, S., Beaudoin, P., and Poulin, P. Particle-based viscoelastic fluid simulation. In *Symposium on Computer Animation 2005* (july 2005), pp. 219–228.

Cohen, J. D., Lin, M. C., Manocha, D., and Ponamgi, M. I-collide: An interactive and exact collision detection system for large-scale environments. In *SI3D '95: Proceedings of the 1995 symposium on Interactive 3D graphics* (New York, NY, USA, 1995), ACM Press, pp. 189–196, 218.

Desbrun, M., and Cani, M.-P. Smoothed particles: A new paradigm for animating highly deformable bodies. In *Eurographics Workshop on Computer Animation and Simulation (EGCAS)* (Aug 1996), R. Boulic and G. Hegron, Eds., Springer-Verlag, pp. 61–76. Published under the name Marie-Paule Gascuel.

[29]However more cache misses or larger memory requirements arise

Enright, Losasso, and Fedkiw. A fast and accurate semi-lagrangian particle level set method. *Computers and Structures 83:* (2005), 479490.

Enright, D., Marschner, S., and Fedkiw, R. Animation and rendering of complex water surfaces. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2002), ACM Press, pp. 736–744.

Foster, N., and Metaxas, D. Realistic animation of liquids. In *GI '96: Proceedings of the conference on Graphics interface '96* (Toronto, Ont., Canada, Canada, 1996), Canadian Information Processing Society, pp. 204–212.

Guendelman, E., Selle, A., Losasso, F., and Fedkiw, R. Coupling water and smoke to thin deformable and rigid shells. *ACM Trans. Graph. 24*, 3 (2005), 973–981.

Harada, T., Koshizuka, S., and Kawaguchi, Y. Smoothed particle hydrodynamics in complex shapes. In *SCCG '07: Proceedings of the 23st spring conference on Computer graphics* (2007).

Harada, T., Koshizuka, S., and Kawaguchi, Y. Smoothed particle hydrodynamics on gpus. In *CGI 2007: Proceedings of the 2007 symposium on computer graphics* (2007).

Hong, J.-M., and Kim, C.-H. Animation of bubbles in liquid. *Computer Graphics Forum 22*, 3 (2003), 253–253.

Irving, G., Guendelman, E., Losasso, F., and Fedkiw, R. Efficient simulation of large bodies of water by coupling two and three dimensional techniques. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers* (New York, NY, USA, 2006), ACM Press, pp. 805–811.

Keiser, R. *Meshless Lagrangian Methods for Physics-Based Animations of Solids and Fluids.* PhD thesis, ETH Zurich, 2006.

Keiser, R., Adams, B., Dutre, P., Guibas, L. J., and Pauly, M. Multiresolution particle-based fluids. Tech. rep., Department of Computer Science, Katholieke Universiteit Leuven, 2006.

Keiser, R., Adams, B., Gasser, D., Bazzi, P., Dutre, P., and Gross, M. A unified lagrangian approach to solid-fluid animation. In *Symposium on Point-Based Graphics* (Zurich, Switzerland, 2005), M. Gross, H. Pfister, M. Alexa, and S. Rusinkiewicz, Eds., Eurographics Association, pp. 125–133.

Kipfer, P., and Westermann, R. Realistic and interactive simulation of rivers. In *Proceedings Graphics Interface 2006* (2006), S. Mann and C. Gutwin, Eds., Canadian Human-Computer Communications Society, pp. 41–48.

Losasso, F. *Algorithms for Increasing the Efficiency and Fidelity of Fluid Simulations.* PhD thesis, Stanford University, Department of Computer Science, 2007.

Losasso, F., Gibou, F., and Fedkiw, R. Simulating water and smoke with an octree data structure. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers* (New York, NY, USA, 2004), ACM Press, pp. 457–462.

Losasso, F., Irving, G., and Guendelman, E. Melting and burning solids into liquids and gases. *IEEE Transactions on Visualization and Computer Graphics 12*, 3 (2006), 343–352. Member-Ron Fedkiw.

Losasso, F., Shinar, T., Selle, A., and Fedkiw, R. Multiple interacting liquids. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers* (New York, NY, USA, 2006), ACM Press, pp. 812–819.

Monaghan, J. Smoothed particle hydrodynamics. *Reports on Progress in Physics 68* (2005), 1703–1759(57).

Müller, M., Charypar, D., and Gross, M. Particle-based fluid simulation for interactive applications. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2003), Eurographics Association, pp. 154–159.

Müller, M., Solenthaler, B., Keiser, R., and Gross, M. Particle-based fluid-fluid interaction. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation* (New York, NY, USA, 2005), ACM Press, pp. 237–244.

Premože, S., Tasdizen, T., Bigler, J., Lefohn, A., and Whitaker, R. Particle-based simulation of fluids. *Computer Graphics Forum 22*, 3 (2003), 401–410. citations 56 4/4/07.

Ritchie, B. W., and Thomas, P. A. Multiphase smoothed-particle hydrodynamics. *Monthly Notices of the Royal Astronomical Society 323* (2001), 743–756(14).

Solenthaler, B., Schläfli, J., and Pajarola, R. A unified particle model for fluidsolid interactions. *Comput. Animat. Virtual Worlds 18*, 1 (2007), 69–82.

Stam, J. Stable fluids. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1999), ACM Press/Addison-Wesley Publishing Co., pp. 121–128.

Terdiman, P. Radix sort revisited. Online Paper, 2000. URL: http://www.codercorner.com/RadixSortRevisited.htm.

Teschner, M., Heidelberger, B., Müller, M., Pomerantes, D., and Gross, M. H. Optimized spatial hashing for collision detection of deformable objects. In *VMV* (2003), pp. 47–54.

Thürey, N., Keiser, R., Pauly, M., and Rüde, U. Detail-preserving fluid control. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2006), Eurographics Association, pp. 7–12.

Thürey, N., Rüde, U., and Stamminger, M. Animation of open water phenomena with coupled shallow water and free surface simulations. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2006), Eurographics Association, pp. 157–164.

Juraj Onderik
Faculty of Mathematics, Physics and Informatics,
Comenius University,
842 48 Bratislava,
Slovak republic
email: juraj.onderik@fmph.uniba.sk

Roman Ďurikovič
Faculty of Natural Sciences,
University of Saint Cyril and Metod,
917 01 Trnava,
Slovak republic
email: roman.durikovic@fmph.uniba.sk

Received October 2007