Simulation in Computer Graphics

Particle-based Fluid Simulation

Matthias Teschner

Computer Science Department
University of Freiburg

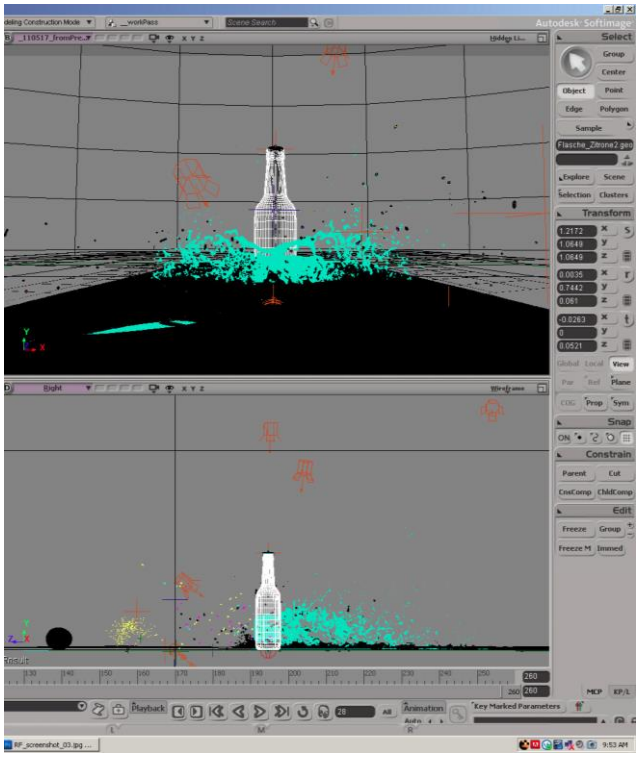
UNI FREIBURG

Application (with Pixar)

10 million fluid + 4 million rigid particles, 50 s simulated,
50 h computation time on a 16-core PC, www.youtube.com/cgfreiburg

UNI FREIBURG

Application (Commercials)



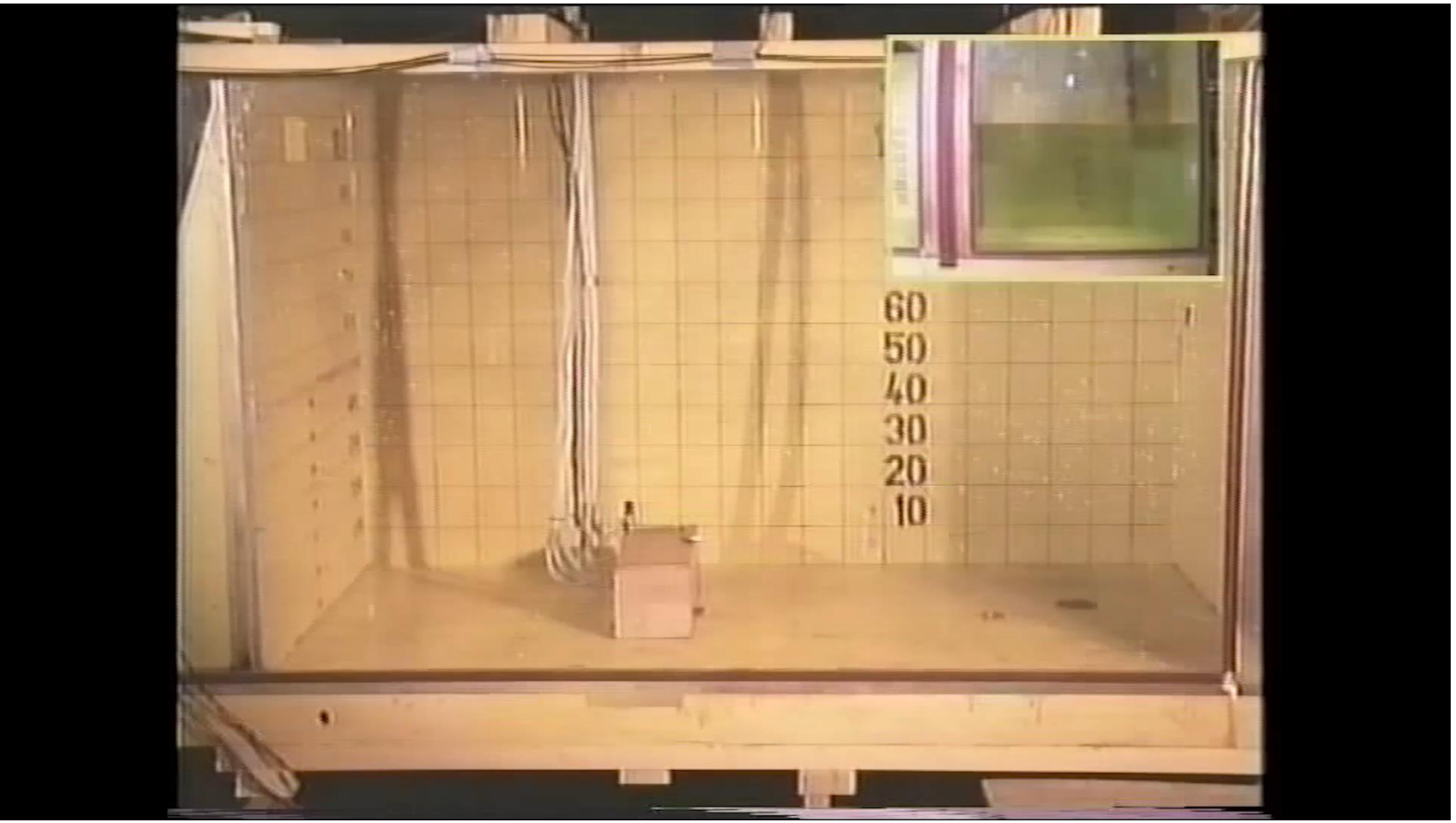
Copyright
NHB Studios,
Berlin,
Hamburg,
Dusseldorf

UNI FREIBURG

Application (with FIFTY2 Technology)

PreonLab, FIFTY2 Technology GmbH, www.youtube.com -> fifty2

Application (with FIFTY2 Technology)



PreonLab, FIFTY2 Technology GmbH, www.youtube.com -> fifty2

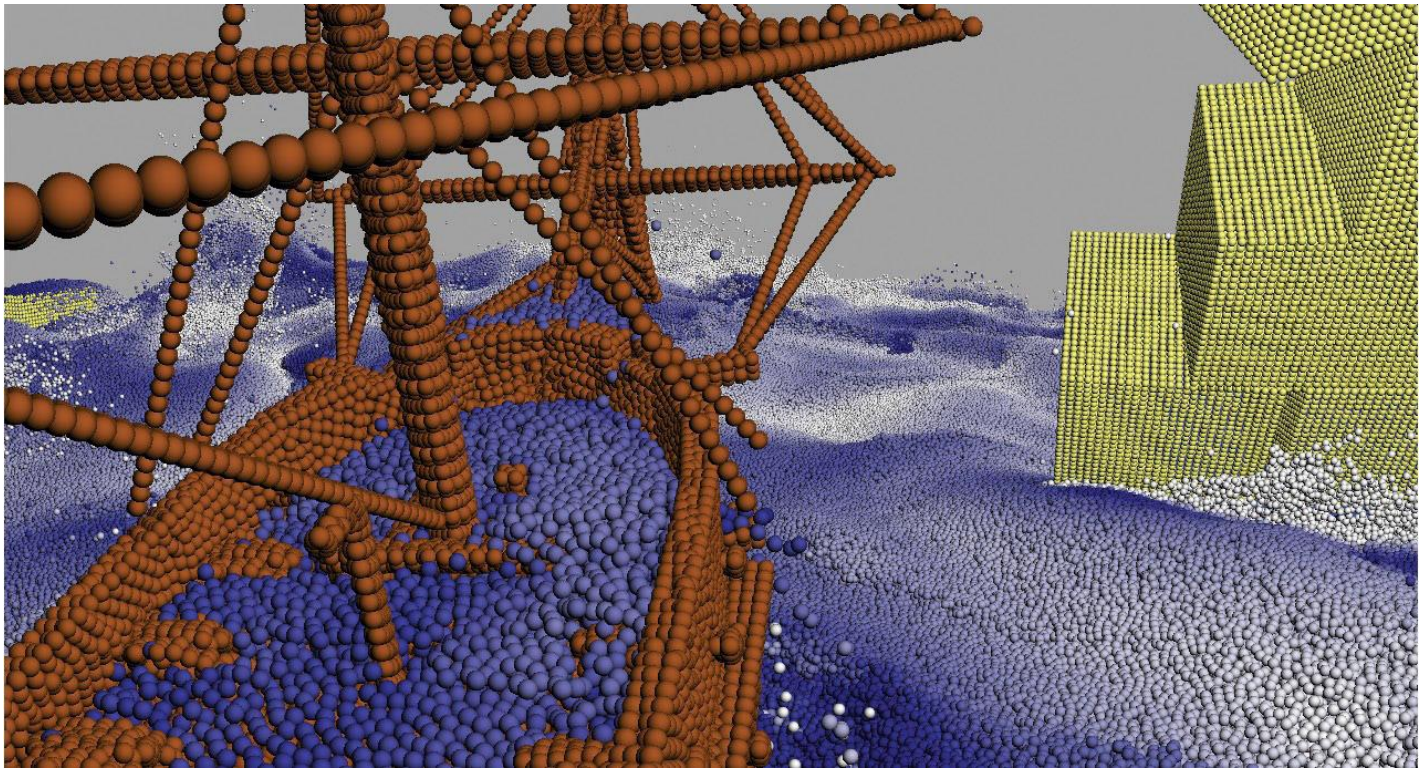
UNI FREIBURG

Outline

- concept of an SPH fluid simulator
- momentum equation
- SPH basics
- neighborhood search
- boundary handling
- incompressibility
- surface reconstruction

UNI FREIBURG

Concept

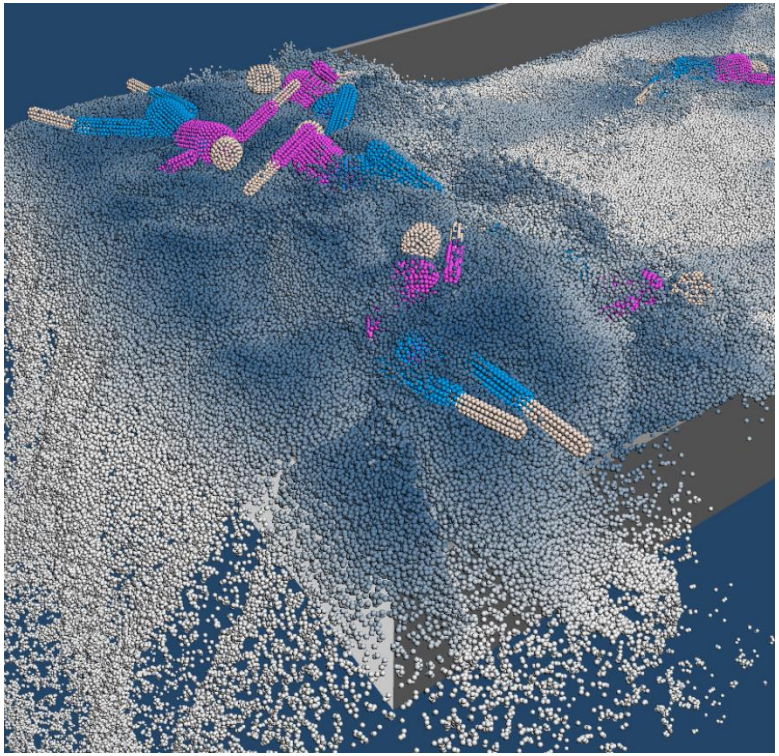


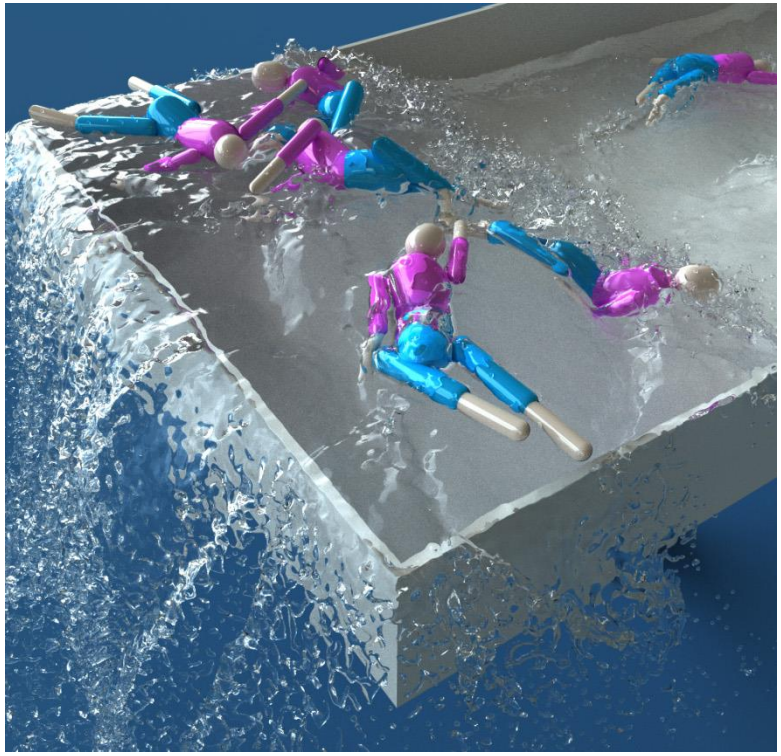
UNI FREIBURG

Concept

UNI FREIBURG

Fluid Representation

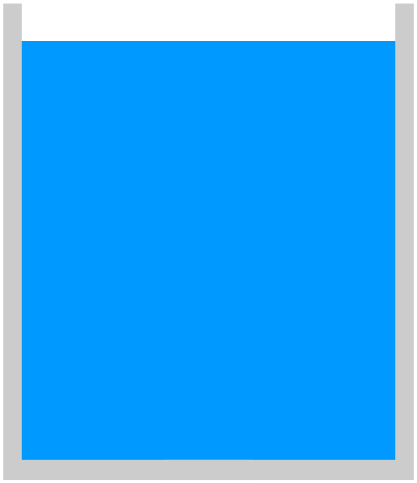




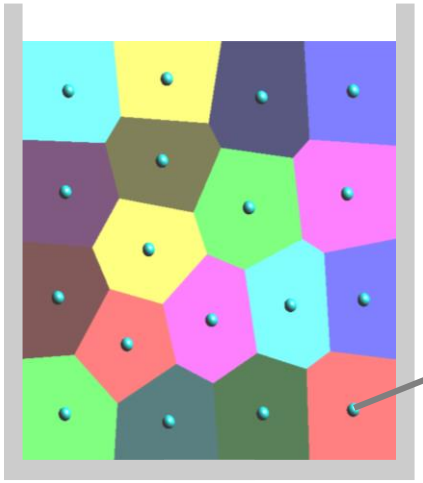
UNI FREIBURG

Fluid Representation

- fluid body is subdivided into small moving parcels, i.e. particles, with fluid properties



Fluid body

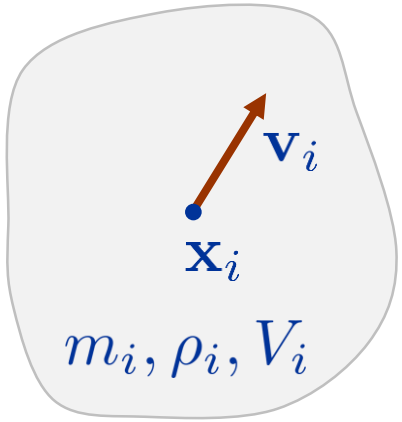


Set of fluid parcels

UNI FREIBURG

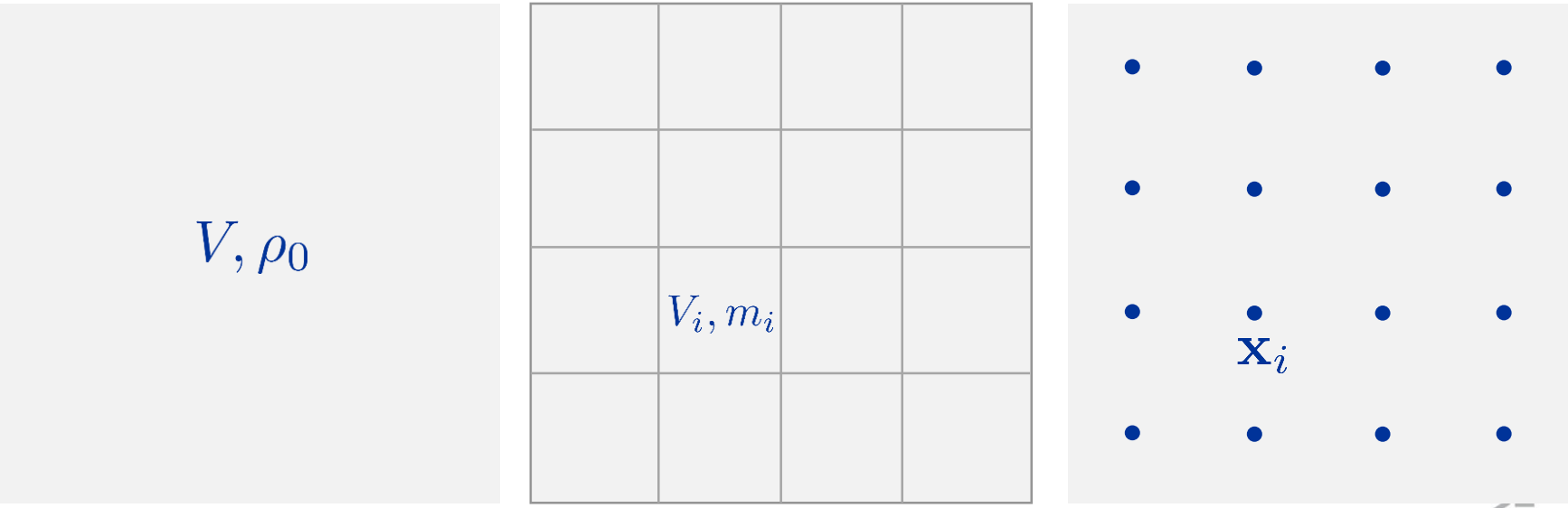
Particles / Fluid Parcels

- represent small fluid portions
- are represented by a sample position $\mathbf{x}_i$
- move with their velocity $\mathbf{v}_i$
- have a fixed mass m_i
- volume and density are related by $V_i = \frac{m_i}{\rho_i}$
  - preservation of density / volume over time is one of the challenges of a fluid simulator
- shape is not considered



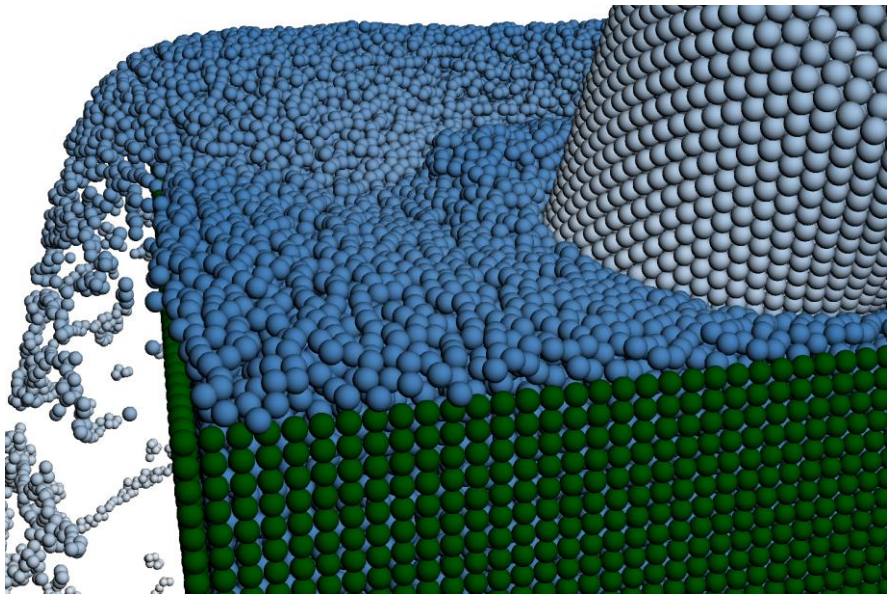
Typical Setup

- define overall fluid volume V and fluid density ρ_0
- define number n of particles
- assume particles of uniform size $V_i = \frac{V}{n}$
- compute particle mass as $m_i = \rho_0 \cdot V_i$
- sample $\mathbf{x}_i$ represents a particle in the simulation

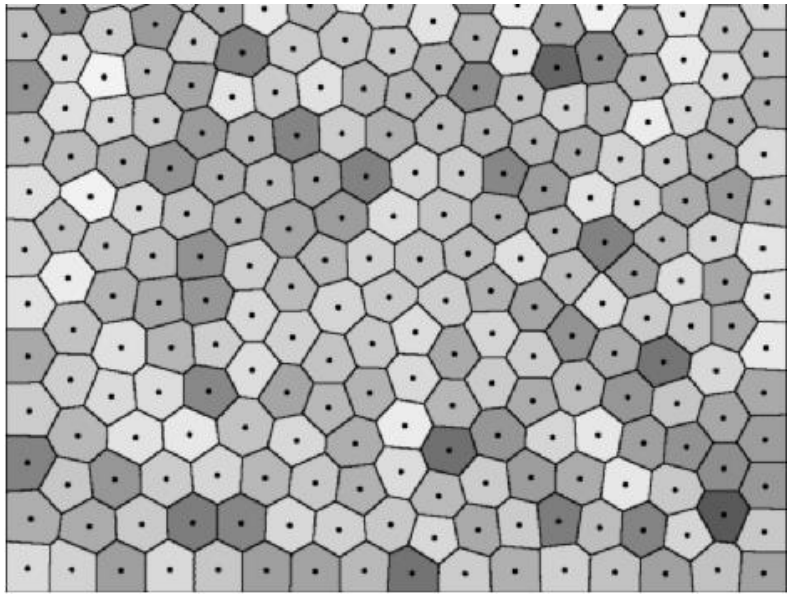


Particle Shape

- typically initialized as a cube
- typically visualized as a sphere
- implicitly handled as Voronoi cell by the simulation



PreonLab, FIFTY2 Technology GmbH



Adrian Secord: Weighted Voronoi Stippling, NPAR 2002.

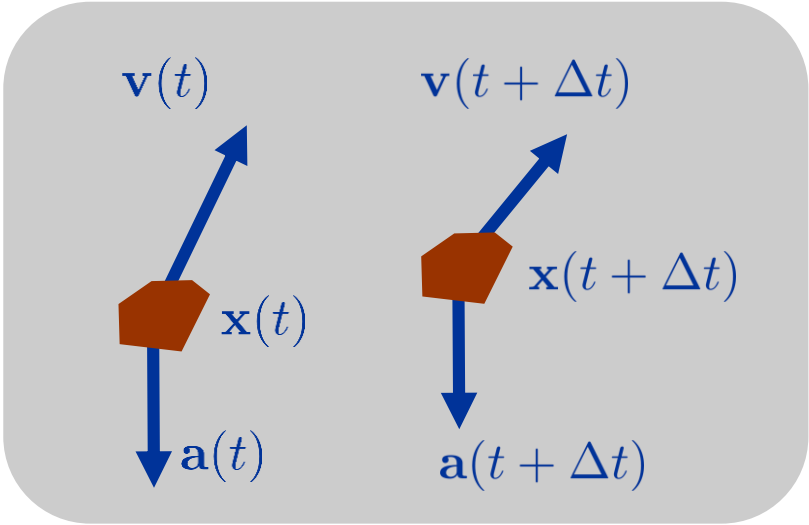
UNI FREIBURG

Fluid Simulation

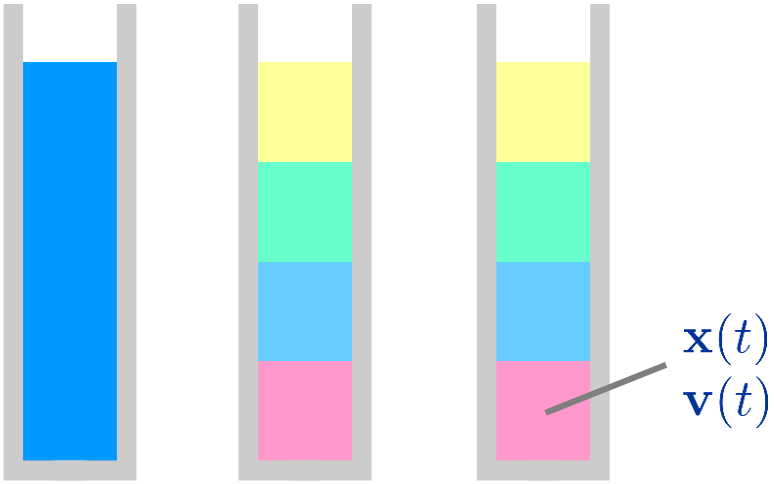
- computation of positions and velocities of fluid parcels over time
  - velocity change from current time t to subsequent time $t + \Delta t$

    $\mathbf{v}(t + \Delta t) = \mathbf{v}(t) + \Delta t \cdot \mathbf{a}(t)$
  - position change

    $\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \Delta t \cdot \mathbf{v}(t + \Delta t)$



Example



Fluid

Fluid parcels

Known current state

Unknown future state

$$\mathbf{v}(t+\Delta t) = \mathbf{v}(t) + \Delta t \cdot \mathbf{a}(t)$$
$$\mathbf{x}(t+\Delta t) = \mathbf{x}(t) + \Delta t \cdot \mathbf{v}(t+\Delta t)$$

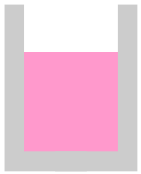
Accelerations

- gravity $\mathbf{g}$
- viscosity $\nu \nabla^2 \mathbf{v}$
  - friction
  - accelerate parcel towards the average velocity of adjacent fluid parcels
- pressure acceleration $-\frac{1}{\rho} \nabla p$
  - prevent fluid parcels from density / volume changes

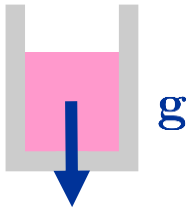
UNI FREIBURG

Simulation Step - Example

- gravity and viscosity would change the parcel volume

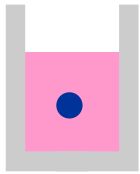


$$\mathbf{x}(t) = \mathbf{0}$$
$$\mathbf{v}(t) = \mathbf{0}$$



$\mathbf{g}$

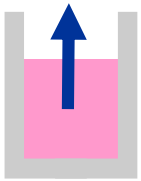
Gravity



$$\nu \nabla^2 \mathbf{v}(t) = \mathbf{0}$$

Viscosity

- pressure acceleration avoids the volume / density change



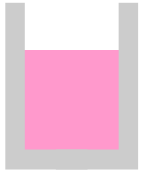
$$-\frac{1}{\rho}\nabla p = -\mathbf{g}$$

Pressure acceleration

UNI FREIBURG

Simulation Step - Example

- current state

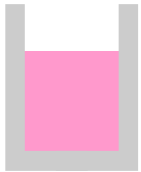


$$\mathbf{x}(t) = \mathbf{0}$$
$$\mathbf{v}(t) = \mathbf{0}$$

- overall acceleration

$$\mathbf{a}(t) = \mathbf{g} + \nu\nabla^2\mathbf{v}(t) - \tfrac{1}{\rho}\nabla p$$
$$= \mathbf{g} + \mathbf{0} - \mathbf{g} = \mathbf{0}$$

- subsequent state



$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \Delta t \cdot \mathbf{v}(t) = \mathbf{0}$$
$$\mathbf{v}(t + \Delta t) = \mathbf{v}(t) + \Delta t \cdot \mathbf{a}(t) = \mathbf{0}$$

UNI FREIBURG

Neighboring Parcels

- computations require neighboring parcels j
- density or volume

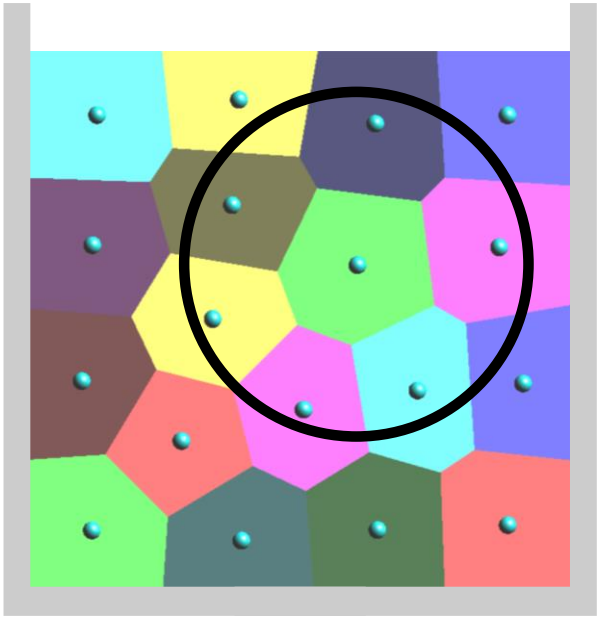
$$\rho_i = \sum_j m_j W_{ij} \quad V_i = \frac{V_i^0}{\sum_j V_j^0 W_{ij}}$$

- pressure acceleration

$$-\frac{V_i}{m_i}\nabla p = -\frac{V_i}{m_i}\sum_j (p_i + p_j) V_j \nabla W_{ij}$$

$$-\frac{1}{\rho_i}\nabla p_i = -\sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}\right) \nabla W_{ij}$$

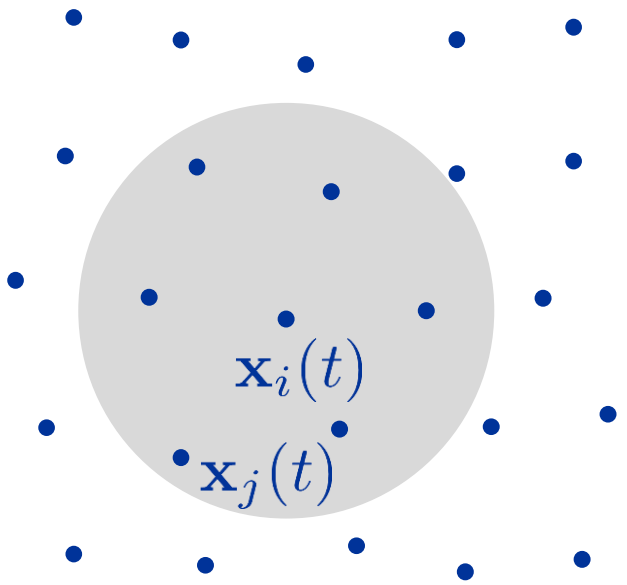
- Smoothed Particle Hydrodynamics
  - Gingold and Monaghan, Lucy



UNI FREIBURG

Simulation Step - Implementation

- determine adjacent particles / neighors
  $\mathbf{x}_j(t)$ of particle $\mathbf{x}_i(t)$ ($\mathbf{x}_i(t)$ is neighor of $\mathbf{x}_i(t)$!)
- compute forces $\mathbf{F}_i(t) = \sum_j \ldots$ as sums of neighbors
- advect the particles, e.g. Euler-Cromer
- determine neighbors
  of particle $\mathbf{x}_i(t + \Delta t)$
- ...



Governing Equations

- particles /sample positions $\mathbf{x}_i$ and the respective attributes are advected with the local fluid velocity $\mathbf{v}_i$

  $$\frac{\mathrm{d}\mathbf{x}_i}{\mathrm{d}t} = \mathbf{v}_i$$

- time rate of change of the velocity $\mathbf{v}_i$ is governed by the Lagrange form of the Navier-Stokes equation

  $$\frac{\mathrm{d}\mathbf{v}_i}{\mathrm{d}t} = -\frac{1}{\rho_i}\nabla p_i + \nu\nabla^2\mathbf{v}_i + \frac{\mathbf{F}_i^{other}}{m_i}$$

  - this form of the Navier-Stokes equation requires that the particle positions are advected with the flow
  - in contrast to the Eulerian form, it does not contain the convective acceleration $\mathbf{v}_i \cdot \nabla\mathbf{v}_i$, which is handled by the advection of the particles / sample positions
  - in Eulerian approaches, sample positions are not necessarily advected with the flow

UNI FREIBURG

Accelerations

- $-\frac{1}{\rho_i}\nabla p_i$: acceleration due to pressure differences
  - preserves the fluid volume / density
  - pressure forces act in normal direction at the surface of the fluid element
  - small and preferably constant density deviations are important for high-quality simulation
- $\nu\nabla^2\mathbf{v}_i$: acceleration due to friction forces between particles with different velocities
  - friction forces act in tangential (and normal) direction at the surface of the fluid element
  - kinematic viscosity $\nu \approx 10^{-6}\mathrm{m}^2 \cdot \mathrm{s}^{-1}$: larger friction is less realistic, but can improve the stability, dynamic viscosity $\eta = \mu = \nu \cdot \rho_0$
- $\frac{\mathbf{F}_i^{other}}{m_i}$: e.g., gravity, boundary handling

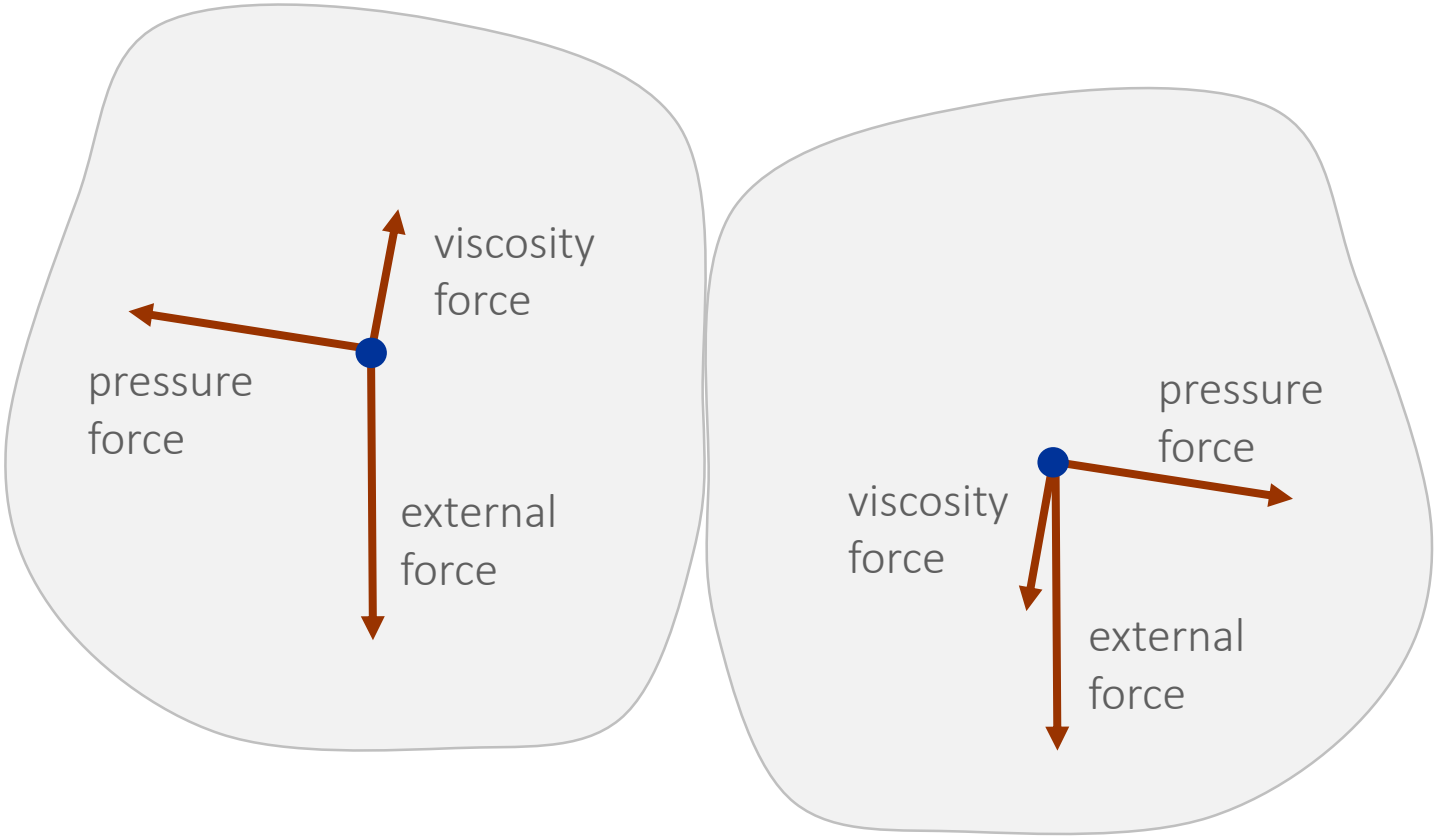
UNI FREIBURG

Accelerations

- $$-\frac{1}{\rho}\nabla p = -\frac{1}{\rho}\begin{pmatrix}\frac{\partial p}{\partial x_x}\\ \frac{\partial p}{\partial x_y}\\ \frac{\partial p}{\partial x_z}\end{pmatrix} = -\frac{1}{\rho}\nabla \cdot \begin{pmatrix}p & 0 & 0\\ 0 & p & 0\\ 0 & 0 & p\end{pmatrix}$$
- $$\nu\nabla^2\mathbf{v} = \nu\nabla\cdot(\nabla\mathbf{v}) = \nu\nabla\cdot\begin{pmatrix}\frac{\partial v_x}{\partial x_x} & \frac{\partial v_x}{\partial x_y} & \frac{\partial v_x}{\partial x_z}\\ \frac{\partial v_y}{\partial x_x} & \frac{\partial v_y}{\partial x_y} & \frac{\partial v_y}{\partial x_z}\\ \frac{\partial v_z}{\partial x_x} & \frac{\partial v_z}{\partial x_y} & \frac{\partial v_z}{\partial x_z}\end{pmatrix}$$
$$= \nu\begin{pmatrix}\frac{\partial^2 v_x}{\partial x_x^2} + \frac{\partial^2 v_x}{\partial x_y^2} + \frac{\partial^2 v_x}{\partial x_z^2}\\ \frac{\partial^2 v_y}{\partial x_x^2} + \frac{\partial^2 v_y}{\partial x_y^2} + \frac{\partial^2 v_y}{\partial x_z^2}\\ \frac{\partial^2 v_z}{\partial x_x^2} + \frac{\partial^2 v_z}{\partial x_y^2} + \frac{\partial^2 v_z}{\partial x_z^2}\end{pmatrix}$$

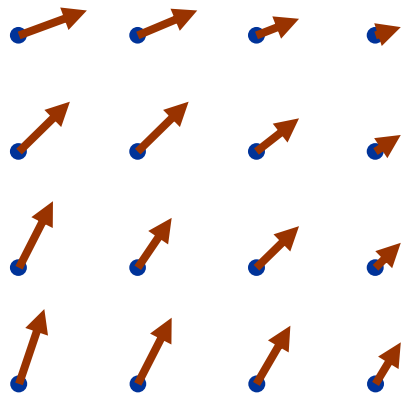
UNI FREIBURG

Forces


viscosity
force
pressure
force
external
force
pressure
force
viscosity
force
external
force

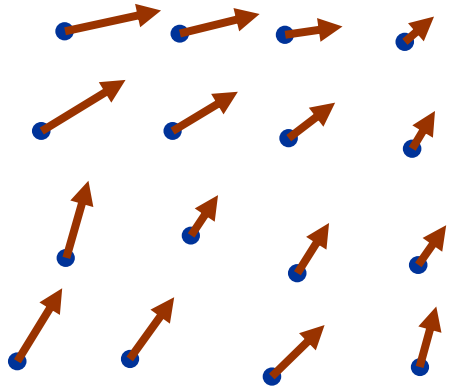
Lagrangian Fluid Simulation

- fluid simulators compute the velocity field over time
- Lagrangian approaches compute the velocities for samples $\mathbf{x}_i$ that are advected with their velocity $\mathbf{v}_i$



$$\mathbf{v}_i(x_i, y_i, z_i, t) = (u_i, v_i, w_i)$$

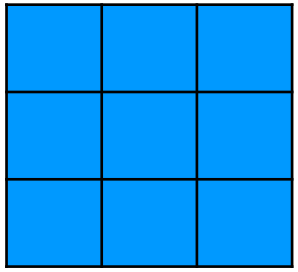
$$\mathbf{x}_i(t) = (x_i, y_i, z_i)$$



$$\mathbf{v}_i(x_i + \Delta t \cdot u_i, y_i + \Delta t \cdot v_i, z_i + \Delta t \cdot w_i, t + \Delta t)$$

$$\mathbf{x}_i(t + \Delta t) = (x_i + \Delta t \cdot u_i, y_i + \Delta t \cdot v_i, z_i + \Delta t \cdot w_i)$$

Moving Parcels vs. Static Cells



$$\frac{d\mathbf{v}}{dt} = \mathbf{g} + \nu\nabla^2\mathbf{v} - \frac{V}{m}\nabla p$$

Acceleration of a moving parcel

Smoothed Particle Hydrodynamics

- proposed by Gingold / Monaghan and Lucy (1977)
- SPH can be used to interpolate fluid quantities at arbitrary positions and to approximate the spatial derivatives in the Navier-Stokes equation with a finite number of samples, i.e., adjacent particles
- SPH in a Lagrangian fluid simulation
  - fluid is represented with particles
  - particle positions and velocities are governed by $\frac{\mathrm{d}\mathbf{x}_i}{\mathrm{d}t} = \mathbf{v}_i$ and $\frac{\mathrm{d}\mathbf{v}_i}{\mathrm{d}t} = -\frac{1}{\rho_i}\nabla p_i + \nu\nabla^2\mathbf{v}_i + \frac{\mathbf{F}_i^{other}}{m_i}$
  - ρ_i , $-\frac{1}{\rho_i}\nabla p_i$ and $\nu\nabla^2\mathbf{v}_i$ are computed with SPH
- SPH is typically used in Lagrangian, mesh-free approaches, but not limited to

UNI FREIBURG

SPH Interpolation

- quantity A_i at an arbitrary position $\mathbf{x}_i$ is approximately computed with a set of known quantities A_j at sample positions $\mathbf{x}_j$

$$A_i = \sum_j V_j A_j W_{ij} = \sum_j \frac{m_j}{\rho_j} A_j W_{ij}$$

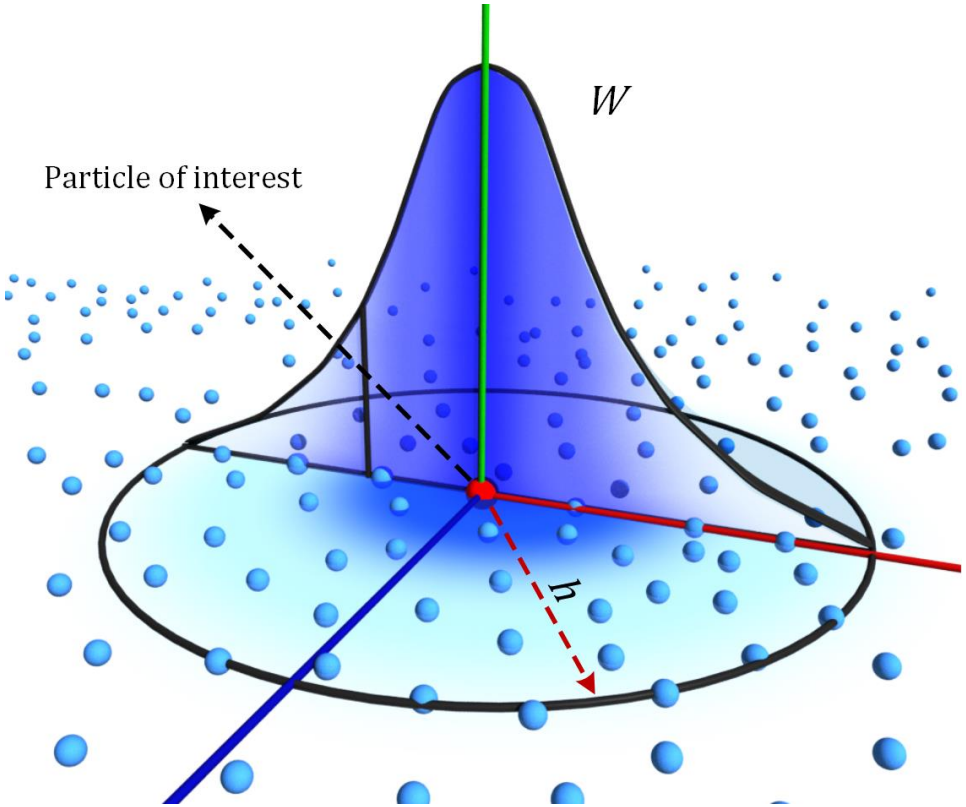
  - $\mathbf{x}_i$ is not necessarily a sample position
  - if $\mathbf{x}_i$ is a sample position, it contributes to the sum
- W_{ij} is a kernel function that weights the contributions of sample positions $\mathbf{x}_j$ according to their distance to $\mathbf{x}_i$

$$W_{ij} = W\left(\frac{\|\mathbf{x}_i - \mathbf{x}_j\|}{h}\right) = W(q)$$

  - d is the dimensionality of the simulation domain
  - h is the so-called smoothing length

UNI FREIBURG

SPH Interpolation – 2D



Neutrino Physics Guide

Kernel Function

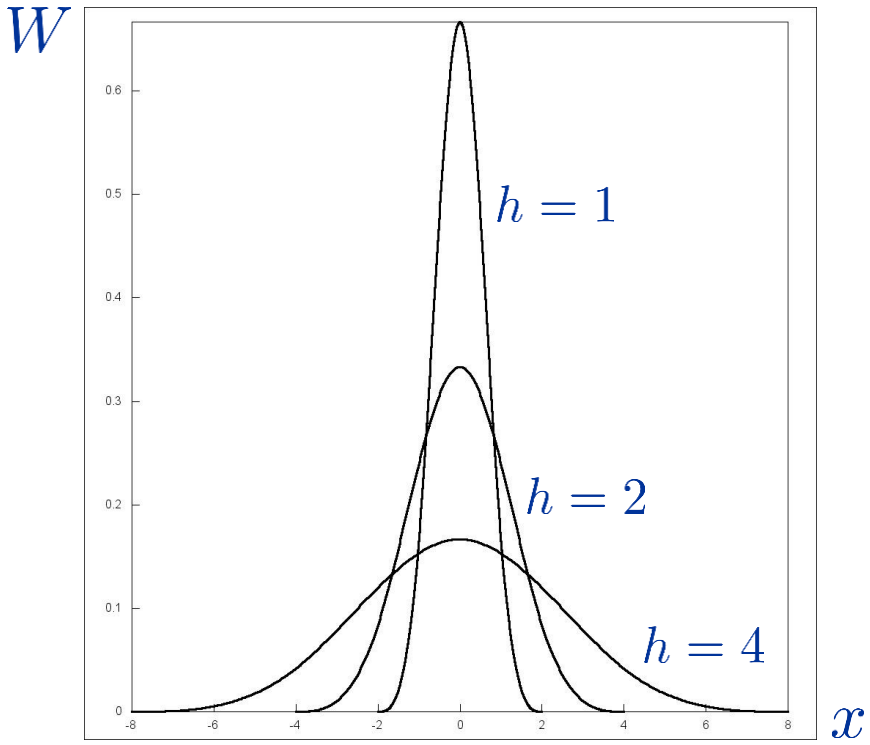
- close to a Gaussian, but with compact support
  - support typically between h and $3h$
- e.g. cubic spline (1D: $\alpha = \frac{1}{6h}$ 2D: $\alpha = \frac{5}{14\pi h^2}$ 3D: $\alpha = \frac{1}{4\pi h^3}$)

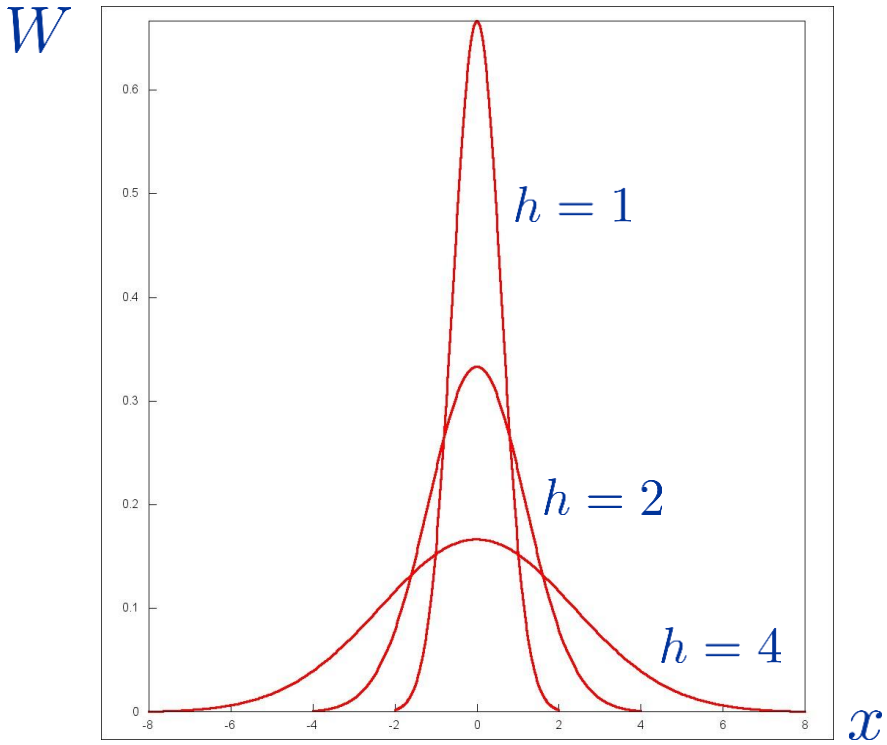
$$W(q) = \alpha \begin{cases} (2-q)^3 - 4(1-q)^3 & 0 \leq q < 1 \\ (2-q)^3 & 1 \leq q < 2 \\ 0 & q \geq 2 \end{cases} \qquad q = \frac{\|\mathbf{x}_i - \mathbf{x}_j\|}{h}$$

- number of particles / samples that are considered in the interpolation depends on
  - dimensionality, kernel support, particle spacing
  - e.g., 3D, cubic spline support $2h$, particle spacing h result in 30-40 neighboring particles
  - number of neighbors should not be too small to appropriately sample the kernel function

UNI FREIBURG

Kernel Function in 1D





$$W(x) = \frac{1}{6h} \begin{cases} (2 - \frac{|x|}{h})^3 - 4(1 - \frac{|x|}{h})^3 & 0 \leq |x| < h \\ (2 - \frac{|x|}{h})^3 & h \leq |x| < 2h \\ 0 & |x| \geq 2h \end{cases}$$

$$W(x) = \frac{2}{3h} e^{-\frac{x^2}{2 \cdot (0.59h)^2}}$$

Spatial Derivatives with SPH

- original approximations

$$\nabla A_i = \sum_j \frac{m_j}{\rho_j} A_j \nabla W_{ij}$$

$$\nabla^2 A_i = \sum_j \frac{m_j}{\rho_j} A_j \nabla^2 W_{ij}$$

- currently preferred approximations

$$\nabla A_i = \rho_i \sum_j m_j \left(\frac{A_i}{\rho_i^2} + \frac{A_j}{\rho_j^2} \right) \nabla W_{ij}$$

preserves linear and angular momentum, when used for pressure forces

$$\nabla^2 A_i = 2 \sum_j \frac{m_j}{\rho_j} A_{ij} \frac{\mathbf{x}_{ij} \cdot \nabla W_{ij}}{\mathbf{x}_{ij} \cdot \mathbf{x}_{ij} + 0.01 h^2}$$

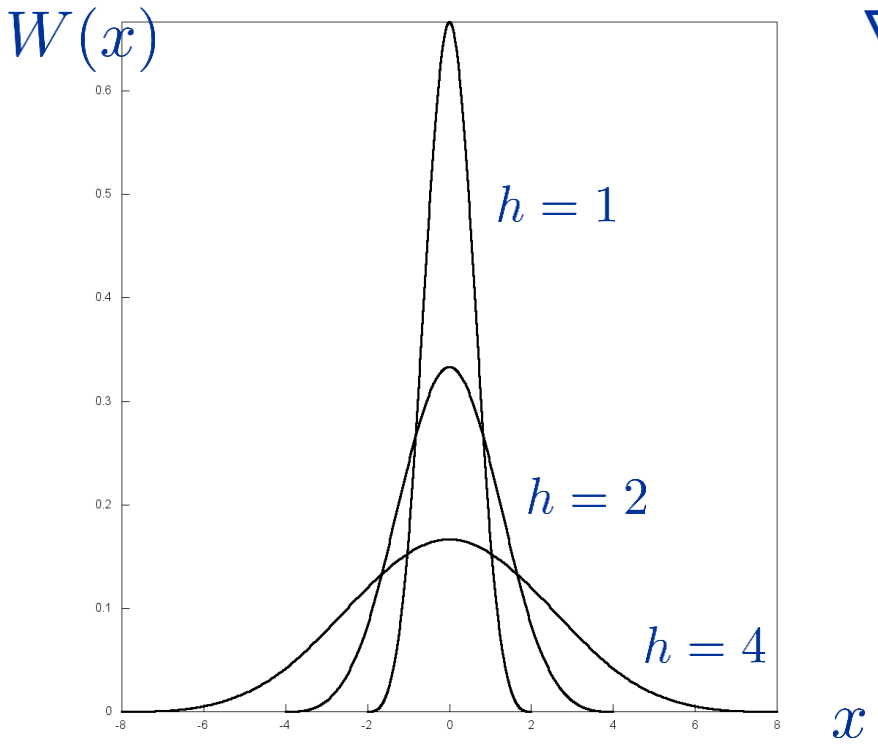
more robust as it avoids the second derivative of W

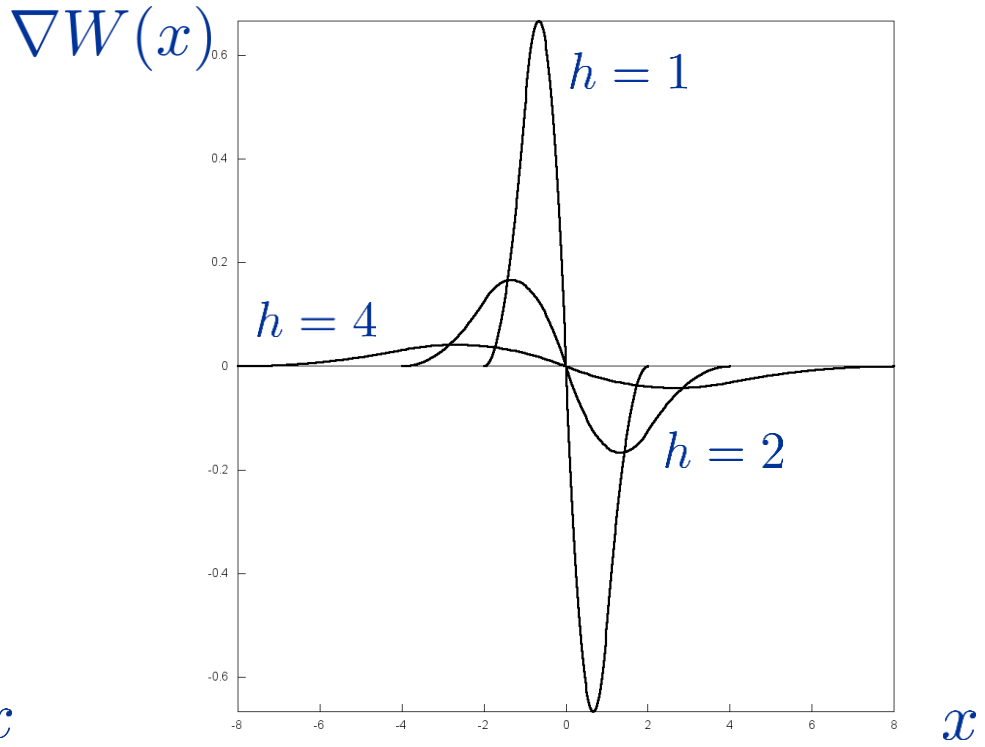
$$\nabla \cdot \mathbf{A}_i = -\frac{1}{\rho_i} \sum_j m_j \mathbf{A}_{ij} \nabla W_{ij}$$

$$A_{ij} = A_i - A_j \qquad \mathbf{A}_{ij} = \mathbf{A}_i - \mathbf{A}_j \qquad \mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j$$

UNI FREIBURG

Kernel Derivative in 1D





$$W(x) = \frac{1}{6h} \begin{cases} (2 - \frac{|x|}{h})^3 - 4(1 - \frac{|x|}{h})^3 & 0 \le |x| < h \\ (2 - \frac{|x|}{h})^3 & h \le |x| < 2h \\ 0 & |x| \ge 2h \end{cases}$$

$$\nabla W(x) = -\frac{x}{6h^2|x|} \begin{cases} -3(2 - \frac{|x|}{h})^2 + 12(1 - \frac{|x|}{h})^2 & 0 \le |x| < h \\ -3(2 - \frac{|x|}{h})^2 & h \le |x| < 2h \\ 0 & |x| \ge 2h \end{cases}$$

Density Computation

- explicit form

$$\rho_i = \sum_j \frac{m_j}{\rho_j} \rho_j W_{ij} = \sum_j m_j W_{ij}$$

  - comparatively exact
  - erroneous for incomplete neighborhood, e.g. at the free surface

- differential update
  - using the continuity equation
  - time rate of change of the density is related to the divergence of the velocity field $\frac{\mathrm{d}\rho_i}{\mathrm{d}t} = -\rho_i \nabla \cdot \mathbf{v}_i$

$$\frac{\mathrm{d}\rho_i}{\mathrm{d}t} = \sum_j m_j \mathbf{v}_{ij} \nabla W_{ij}$$

  - no issues for incomplete neighborhoods
  - drift, i.e. less accurate for large time steps

UNI FREIBURG

Simple SPH Fluid Solver

- find all neighbors j of particle i
  - typically accelerated with a uniform grid
  - cell size equal to kernel support , e.g. $2h$
- compute pressure p_i
  - e.g., from density ρ_i using a state equation, e.g. $p_i = k\left(\left(\frac{\rho_i}{\rho_0}\right)^7 - 1\right)$
  - ρ_0 is the desired rest density of the fluid
  - k is a user-defined stiffness constant that scales pressure, pressure gradient, and the resulting pressure force
  - SPH with state equation is referred to as SESPH
- compute pressure force, viscosity / gravitational force
- compute other forces, e.g. due to boundaries
- update velocity and position

UNI FREIBURG

Simple SPH Fluid Solver

- **for all** *particle* i **do**
  find neighbors j

  for all *particle* i **do**
  $\rho_i = \sum_j m_j W_{ij}$
  compute p_i from ρ_i

  for all *particle* i **do**
  $\mathbf{F}_i^{pressure} = -\frac{m_i}{\rho_i} \nabla p_i$
  $\mathbf{F}_i^{viscosity} = m_i \nu \nabla^2 \mathbf{v}_i$
  $\mathbf{F}_i^{other} = m_i \mathbf{g}$
  $\mathbf{F}_i(t) = \mathbf{F}_i^{pressure} + \mathbf{F}_i^{viscosity} + \mathbf{F}_i^{other}$

  for all *particle* i **do**
  $\mathbf{v}_i(t + \Delta t) = \mathbf{v}_i(t) + \Delta t \mathbf{F}_i(t)/m_i$
  $\mathbf{x}_i(t + \Delta t) = \mathbf{x}_i(t) + \Delta t \mathbf{v}_i(t + \Delta t)$

UNI FREIBURG

Simple SPH Fluid Solver

- **for all** *particle* i **do**
  find neighbors j

  for all *particle* i **do**
  $\rho_i = \sum_j m_j W_{ij}$
  compute p_i from ρ_i

  for all *particle* i **do**
  $\mathbf{a}_i^{pressure} = -\frac{1}{\rho_i}\nabla p_i$
  $\mathbf{a}_i^{viscosity} = \nu\nabla^2\mathbf{v}_i$
  $\mathbf{a}_i^{other} = \mathbf{g}$
  $\mathbf{a}_i(t) = \mathbf{a}_i^{pressure} + \mathbf{a}_i^{viscosity} + \mathbf{a}_i^{other}$

  for all *particle* i **do**
  $\mathbf{v}_i(t+\Delta t) = \mathbf{v}_i(t) + \Delta t\mathbf{a}_i(t)$
  $\mathbf{x}_i(t+\Delta t) = \mathbf{x}_i(t) + \Delta t\mathbf{v}_i(t+\Delta t)$

UNI FREIBURG

Accelerations with SPH

- $$\mathbf{a}_i^{pressure} = -\sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) \nabla W_{ij}$$
  $$\mathbf{a}_i^{viscosity} = 2\nu \sum_j \frac{m_j}{\rho_j} \mathbf{v}_{ij} \frac{\mathbf{x}_{ij} \cdot \nabla W_{ij}}{\mathbf{x}_{ij} \cdot \mathbf{x}_{ij} + 0.01h^2}$$
  $$\mathbf{a}_i^{other} = \mathbf{g}$$

UNI FREIBURG

Setting

- kernel has to be defined, e.g. cubic with support of $2h$
- particle mass m_i has to be specified
  - e.g., $m_i = h^3 \rho_0$ for a particle spacing of h
  - smaller spacing would result in smaller mass and more neighbors per particle
- numerical integration scheme
  - semi-implicit Euler (a.k.a. symplectic Euler or Euler-Cromer) is commonly used
- time step
  - size is governed by the Courant-Friedrich-Levy (CFL) condition
  - e.g., $\Delta t \leq \lambda \frac{h}{\|\mathbf{v}^{\max}\|}$ with $\lambda = 0.1$ and particle spacing h
  - motivation: for $\lambda \leq 1$, a particle moves less than its diameter per time step

UNI FREIBURG

Simulation in Computer Graphics

Particle-based Fluid Simulation

Matthias Teschner

Computer Science Department
University of Freiburg

UNI FREIBURG

Simple SPH Fluid Solver

- **for all** *particle* i **do**
  find neighbors j
  for all *particle* i **do**
  $\rho_i = \sum_j m_j W_{ij}$
  compute p_i from ρ_i
  for all *particle* i **do**
  $\mathbf{a}_i^{pressure} = -\frac{1}{\rho_i}\nabla p_i$
  $\mathbf{a}_i^{viscosity} = \nu\nabla^2\mathbf{v}_i$
  $\mathbf{a}_i^{other} = \mathbf{g}$
  $\mathbf{a}_i(t) = \mathbf{a}_i^{pressure} + \mathbf{a}_i^{viscosity} + \mathbf{a}_i^{other}$
  for all *particle* i **do**
  $\mathbf{v}_i(t+\Delta t) = \mathbf{v}_i(t) + \Delta t\mathbf{a}_i(t)$
  $\mathbf{x}_i(t+\Delta t) = \mathbf{x}_i(t) + \Delta t\mathbf{v}_i(t+\Delta t)$

Navier-Stokes equation

$$\mathbf{a}_i^{pressure} = -\sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}\right)\nabla W_{ij}$$
$$\mathbf{a}_i^{viscosity} = 2\nu\sum_j \frac{m_j}{\rho_j}\mathbf{v}_{ij}\frac{\mathbf{x}_{ij}\cdot\nabla W_{ij}}{\mathbf{x}_{ij}\cdot\mathbf{x}_{ij}+0.01h^2}$$
$$\mathbf{a}_i^{other} = \mathbf{g}$$

SPH approximations

UNI FREIBURG

Outline

- concept of an SPH fluid simulator
- momentum equation
- SPH basics
- neighborhood search
- boundary handling
- incompressibility
- surface reconstruction

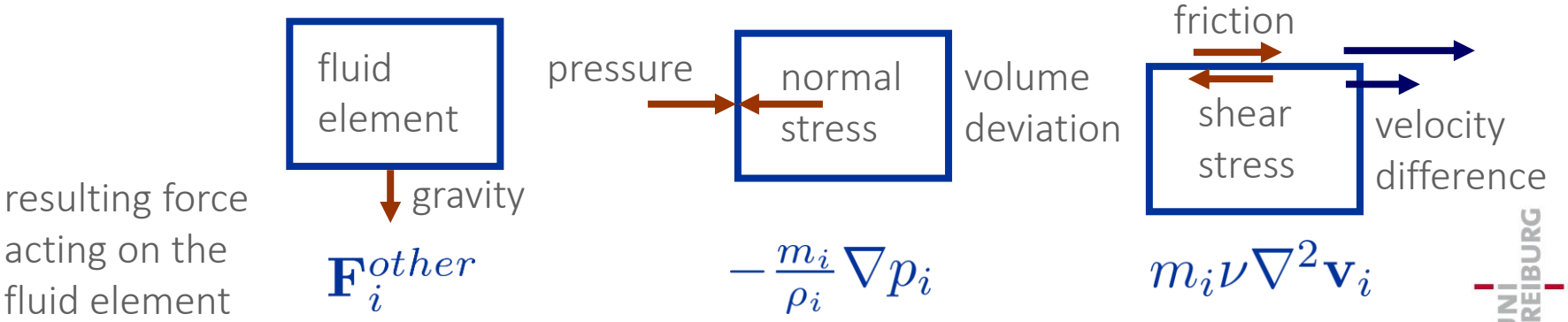
UNI FREIBURG

Force Types

- momentum equation

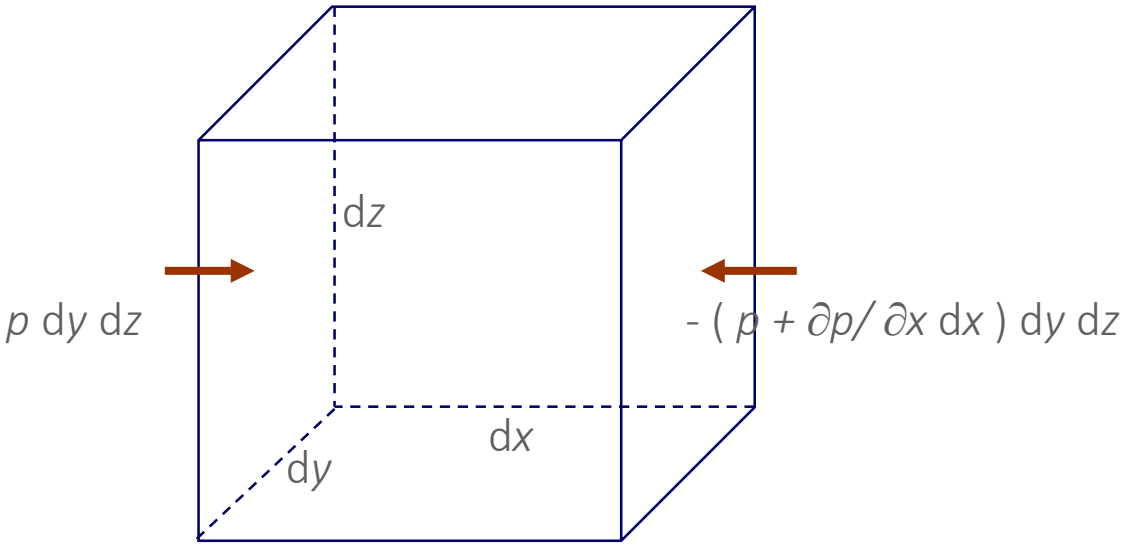
$$\frac{\mathrm{d}\mathbf{v}_i}{\mathrm{d}t} = -\frac{1}{\rho_i}\nabla p_i + \nu\nabla^2\mathbf{v}_i + \frac{\mathbf{F}_i^{other}}{m_i}$$

- body forces, e.g. gravity (external)
- surface forces (internal, i.e. conservative)
  - based on shear and normal stress distribution on the surface due to deformation of the fluid element
  - normal stress related to volume deviation
  - normal and shear stress related to friction due to velocity differences



Pressure Force in x-direction

- pressure force acts orthogonal to the surface of the fluid element



- resulting pressure force

$$\left(p - \left(p + \frac{\partial p}{\partial x}\,\mathrm{d}x\right)\right)\,\mathrm{d}y\,\mathrm{d}z = -\frac{\partial p}{\partial x}\,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z = -\frac{\partial p}{\partial x}\,V$$

Overall Pressure Force

- force at particle i

$$\mathbf{F}_i^{pressure} = -\begin{pmatrix} \frac{\partial p_i}{\partial x_{i,x}} \\ \frac{\partial p_i}{\partial x_{i,y}} \\ \frac{\partial p_i}{\partial x_{i,z}} \end{pmatrix} V_i = -\nabla p_i \; V_i = -\frac{m_i}{\rho_i} \nabla p_i$$

- respective acceleration

$$\mathbf{a}_i^{pressure} = \frac{\mathbf{F}_i^{pressure}}{m_i} = -\frac{1}{\rho_i} \nabla p_i$$

UNI FREIBURG

Cauchy Momentum Equation

- Lagrange form
  $\frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t} = \frac{1}{\rho}\nabla \cdot \sigma + \frac{\mathbf{F}^{other}}{m}$
- σ is the stress tensor (a 3x3 matrix in 3D) describing the pressure distribution at the surface of a fluid element $\sigma = -p\mathbf{I_3} + \tau$
- $\nabla \cdot \sigma$ is the resulting force per volume acting on the fluid element
- τ is the viscous stress tensor $\tau = \nu \begin{pmatrix} \frac{\partial u}{\partial x} + \frac{\partial u}{\partial x} & \frac{\partial v}{\partial x} + \frac{\partial u}{\partial y} & \frac{\partial w}{\partial x} + \frac{\partial u}{\partial z} \\ \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} + \frac{\partial v}{\partial y} & \frac{\partial w}{\partial y} + \frac{\partial v}{\partial z} \\ \frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} & \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} & \frac{\partial w}{\partial z} + \frac{\partial w}{\partial z} \end{pmatrix}$
- $\nabla \cdot \tau = \nu\nabla^2\mathbf{v}$ is the resulting viscosity force per volume
- $\frac{\mathrm{d}\mathbf{v}_i}{\mathrm{d}t} = -\frac{1}{\rho_i}\nabla p_i + \nu\nabla^2\mathbf{v}_i + \frac{\mathbf{F}_i^{other}}{m_i}$

UNI FREIBURG

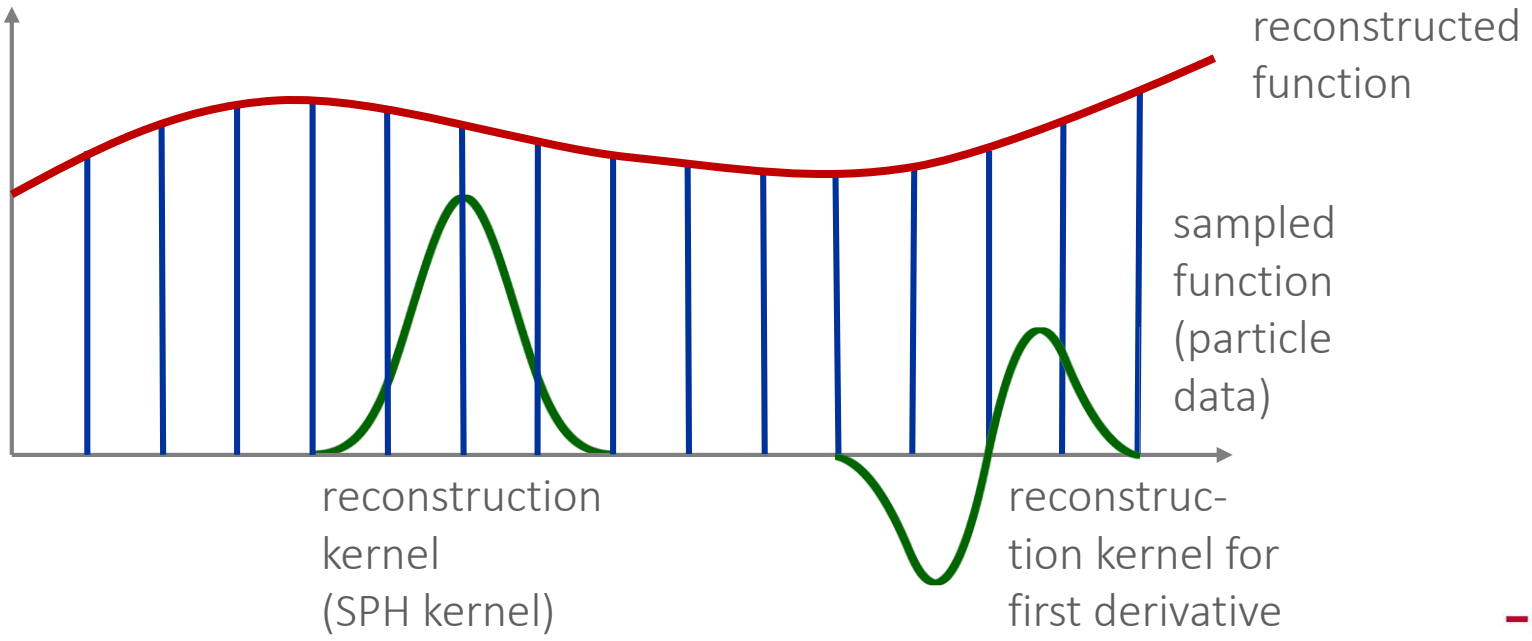
Outline

- concept of an SPH fluid simulator
- momentum equation
- SPH basics
- neighborhood search
- boundary handling
- incompressibility
- surface reconstruction

UNI FREIBURG

Illustration

- approximate a function and its derivatives from discrete samples, e.g. $\rho, \nabla p, \nabla^2 \mathbf{v}$
- convolution of discrete samples with reconstruction filter, e.g. cubic spline



UNI FREIBURG

Derivation

- quantity A at position $\mathbf{x}$ can be written as
  $A(\mathbf{x}) = \int_\Omega A(\mathbf{x}')\delta(\mathbf{x}-\mathbf{x}')\mathrm{d}\mathbf{x}'$
- dirac delta $\delta(\mathbf{x}) = \delta(x)\delta(y)\delta(z)$ and $\delta(x) = \begin{cases} \infty & x = 0 \\ 0 & x \neq 0 \end{cases}$
- $\int_{-\infty}^{+\infty} \delta(x)\mathrm{d}x = 1$
- dirac delta is approximated with a kernel function with limited local support h
  $A(\mathbf{x}) \approx \int_{\Omega_h} A(\mathbf{x}')W(\mathbf{x}-\mathbf{x}', h)\mathrm{d}\mathbf{x}'$

UNI FREIBURG

Kernel Function

- integral should be normalized (unity condition)
  $\int_\Omega W(\mathbf{x}-\mathbf{x}', h)\mathrm{d}\mathbf{x}' = 1$
- support should be compact
  $W(\mathbf{x}-\mathbf{x}', h) = 0 \text{ for } \|\mathbf{x}-\mathbf{x}'\| > h$
- should be symmetric
  $W(\mathbf{x}-\mathbf{x}', h) = W(\mathbf{x}'-\mathbf{x}, h)$
- should be non-negative
  $W(\mathbf{x}-\mathbf{x}', h) \geq 0$
- should converge to the Dirac delta for $h \to 0$
- should be differentiable

UNI FREIBURG

Kernel Function

- close to a Gaussian, but with compact support
  - support typically between h and $3h$
- e.g. cubic spline (1D: $\alpha = \frac{1}{6h}$ 2D: $\alpha = \frac{5}{14\pi h^2}$ 3D: $\alpha = \frac{1}{4\pi h^3}$)

$$W(q) = \alpha \begin{cases} (2-q)^3 - 4(1-q)^3 & 0 \leq q < 1 \\ (2-q)^3 & 1 \leq q < 2 \\ 0 & q \geq 2 \end{cases} \qquad q = \frac{\|\mathbf{x}_i - \mathbf{x}_j\|}{h}$$

- number of particles / samples that are considered in the interpolation depends on
  - dimensionality, kernel support, particle spacing
  - e.g., 3D, cubic spline support $2h$, particle spacing h result practically in 30-40 neighbors
  - number of neighbors should not be too small to appropriately sample the kernel function

UNI FREIBURG

First Kernel Derivative

- $$\nabla W_{ij} = \left(\frac{\partial W_{ij}}{\partial x_{i,x}}, \frac{\partial W_{ij}}{\partial x_{i,y}}, \frac{\partial W_{ij}}{\partial x_{i,z}} \right)^{\mathrm{T}}$$

  $$\nabla W_{ij} = \frac{\partial W(q)}{\partial q} \nabla q$$

- e.g. cubic spline (1D: $\alpha = \frac{1}{6h}$ 2D: $\alpha = \frac{5}{14\pi h^2}$ 3D: $\alpha = \frac{1}{4\pi h^3}$)

  $$\nabla q = \frac{\mathbf{x}_{ij}}{\|\mathbf{x}_{ij}\| h}$$

  $$\frac{\partial W(q)}{\partial q} = \alpha \begin{cases} -3(2-q)^2 + 12(1-q)^2 & 0 \leq q < 1 \\ -3(2-q)^2 & 1 \leq q < 2 \\ 0 & q \geq 2 \end{cases}$$

  $$\nabla W_{ij} = \alpha \frac{\mathbf{x}_{ij}}{\|\mathbf{x}_{ij}\| h} \begin{cases} -3(2-q)^2 + 12(1-q)^2 & 0 \leq q < 1 \\ -3(2-q)^2 & 1 \leq q < 2 \\ 0 & q \geq 2 \end{cases}$$

UNI FREIBURG

Second Kernel Derivative

- $\nabla^2 W_{ij} = \nabla \cdot (\nabla W_{ij}) = \frac{\partial^2 W_{ij}}{\partial x_{i,x}^2} + \frac{\partial^2 W_{ij}}{\partial x_{i,y}^2} + \frac{\partial W_{ij}^2}{\partial x_{i,z}^2}$

  $\nabla^2 W_{ij} = \frac{\partial^2 W(q)}{\partial q^2}(\nabla q)^2 + \frac{\partial W(q)}{\partial q}(\nabla \cdot (\nabla q))$

- e.g. cubic spline (1D: $\alpha = \frac{1}{6h}$ 2D: $\alpha = \frac{5}{14\pi h^2}$ 3D: $\alpha = \frac{1}{4\pi h^3}$)

  $(\nabla q)^2 = \frac{\mathbf{x}_{ij}}{\|\mathbf{x}_{ij}\|h} \cdot \frac{\mathbf{x}_{ij}}{\|\mathbf{x}_{ij}\|h} = \frac{\|\mathbf{x}_{ij}\|^2}{\|\mathbf{x}_{ij}\|^2 h^2} = \frac{1}{h^2}$

  $\nabla \cdot (\nabla q) = \frac{2}{h\|\mathbf{x}\|}$

  $$\frac{\partial W(q)}{\partial q} = \alpha \begin{cases} -3(2-q)^2 + 12(1-q)^2 & 0 \leq q < 1 \\ -3(2-q)^2 & 1 \leq q < 2 \\ 0 & q \geq 2 \end{cases}$$

  $$\frac{\partial^2 W(q)}{\partial q^2} = \alpha \begin{cases} 6(2-q) - 24(1-q) & 0 \leq q < 1 \\ 6(2-q) & 1 \leq q < 2 \\ 0 & q \geq 2 \end{cases}$$

UNI FREIBURG

Design of a Kernel Function 1D

- shape close to a Gaussian, e.g.
- $$\alpha\tilde{W}(\tfrac{\|\mathbf{x}_i-\mathbf{x}_j\|}{h}) = \alpha\tilde{W}(\tfrac{x}{h}) = \alpha\tilde{W}(q) = W(q) = \alpha \begin{cases} (2-q)^3 - 4(1-q)^3 & 0 \le q < 1 \\ (2-q)^3 & 1 \le q < 2 \\ 0 & q \ge 2 \end{cases}$$
- $2\int_0^{2h} \alpha\tilde{W}(x)\mathrm{d}x = 2\int_0^2 \alpha\tilde{W}(q)h\mathrm{d}q = 1$ integration by substitution
- $\alpha = \frac{1}{2\int_0^2 \tilde{W}(q)h\mathrm{d}q}$
- 1D: integration over a line segment $2\int_0^2 \tilde{W}(q)h\mathrm{d}q =$

  $2\int_0^1 \left[(2-q)^3 - 4(1-q)^3\right] h\mathrm{d}q + 2\int_1^2 (2-q)^3 h\mathrm{d}q = 2\frac{11}{4}h + 2\frac{1}{4}h$

  $\alpha = \frac{1}{6h}$

Design of a Kernel Function

- 2D: integration over the area of a circle

$$\int_0^{2\pi}\int_0^{2h}\tilde{W}(x)x\,\mathrm{d}x\mathrm{d}\phi = \int_0^{2\pi}\int_0^{2}\tilde{W}(q)hqh\,\mathrm{d}q\mathrm{d}\phi =$$

$$2\pi\int_0^1\left[q(2-q)^3-4q(1-q)^3\right]h^2\mathrm{d}q+2\pi\int_1^2 q(2-q)^3h^2\mathrm{d}q = 2\pi\tfrac{11}{10}h^2+2\pi\tfrac{3}{10}h^2$$

$$\alpha = \frac{5}{14\pi h^2}$$

- 3D: integration over the volume of a sphere

$$\int_0^{2\pi}\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}}\int_0^{2h}\tilde{W}(x)x^2\sin\theta\,\mathrm{d}x\mathrm{d}\theta\mathrm{d}\phi = \int_0^{2\pi}\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}}\int_0^{2}\tilde{W}(q)(qh)^2h\sin\theta\,\mathrm{d}q\mathrm{d}\theta\mathrm{d}\phi =$$

$$4\pi\int_0^1\left[q^2(2-q)^3-4q(1-q)^3\right]h^3\mathrm{d}q+4\pi\int_1^2 q^2(2-q)^3h^3\mathrm{d}q = 4\pi\tfrac{19}{30}h^3+4\pi\tfrac{11}{30}h^3$$

$$\alpha = \frac{1}{4\pi h^3}$$

UNI FREIBURG

Particle Approximation

- $A(\mathbf{x}) \approx \int_{\Omega_h} A(\mathbf{x}')W(\mathbf{x}-\mathbf{x}',h)\mathrm{d}\mathbf{x}'$

  $= \int_{\Omega_h} \frac{A(\mathbf{x}')}{\rho(\mathbf{x}')}W(\mathbf{x}-\mathbf{x}',h)\rho(\mathbf{x}')\mathrm{d}\mathbf{x}'$

- consider a limited number of samples / particles $\mathbf{x}_j$ representing a mass $m(\mathbf{x}_j) = \rho(\mathbf{x}_j)V(\mathbf{x}_j)$

  $A(\mathbf{x}_i) \approx \sum_j A(\mathbf{x}_j)W(\mathbf{x}_i-\mathbf{x}_j,h)V(\mathbf{x}_j)$

  $A(\mathbf{x}_i) \approx \sum_j \frac{A(\mathbf{x}_j)}{\rho(\mathbf{x}_j)}W(\mathbf{x}_i-\mathbf{x}_j,h)m(\mathbf{x}_j)$

- typical notation

  $A_i = \sum_j \frac{m_j}{\rho_j}A_j W_{ij}$

UNI FREIBURG

Spatial Derivatives

- $\nabla_{\mathbf{x}} A(\mathbf{x}) \approx \int_{\Omega_h} [\nabla_{\mathbf{x}'} A(\mathbf{x}')] \, W(\mathbf{x} - \mathbf{x}', h) \mathrm{d}\mathbf{x}'$
- $\nabla_{\mathbf{x}'} [A(\mathbf{x}') W(\mathbf{x}' - \mathbf{x}, h)] = [\nabla_{\mathbf{x}'} A(\mathbf{x}')] \, W(\mathbf{x}' - \mathbf{x}, h) + A(\mathbf{x}') \nabla_{\mathbf{x}'} W(\mathbf{x}' - \mathbf{x}, h)$

  W is symmetric

  $\nabla_{\mathbf{x}'} [A(\mathbf{x}') W(\mathbf{x}' - \mathbf{x}, h)] = [\nabla_{\mathbf{x}'} A(\mathbf{x}')] \, W(\mathbf{x} - \mathbf{x}', h) + A(\mathbf{x}') \nabla_{\mathbf{x}'} W(\mathbf{x}' - \mathbf{x}, h)$

  $[\nabla_{\mathbf{x}'} A(\mathbf{x}')] \, W(\mathbf{x} - \mathbf{x}', h) = \nabla_{\mathbf{x}'} [A(\mathbf{x}') W(\mathbf{x}' - \mathbf{x}, h)] - A(\mathbf{x}') \nabla_{\mathbf{x}'} W(\mathbf{x}' - \mathbf{x}, h)$
- $\int_{\Omega_h} \nabla_{\mathbf{x}'} [A(\mathbf{x}') W(\mathbf{x}' - \mathbf{x}, h)] \, \mathrm{d}\mathbf{x}' = \int_S A(\mathbf{x}') W(\mathbf{x}' - \mathbf{x}, h) \mathbf{n} \mathrm{d}S$ Gauss theorem, S is the surface of Ω

  $\int_S A(\mathbf{x}') W(\mathbf{x}' - \mathbf{x}, h) \mathbf{n} \mathrm{d}S = 0$ W = 0 on the surface S

  $\nabla_{\mathbf{x}} A(\mathbf{x}) \approx - \int_{\Omega_h} A(\mathbf{x}') \nabla_{\mathbf{x}'} W(\mathbf{x}' - \mathbf{x}, h) \mathrm{d}\mathbf{x}' = \int_{\Omega_h} A(\mathbf{x}') \nabla_{\mathbf{x}} W(\mathbf{x} - \mathbf{x}', h) \mathrm{d}\mathbf{x}'$
- $\nabla_x A(\mathbf{x}_i) \approx \sum_j A(\mathbf{x}_j) \nabla W(\mathbf{x}_i - \mathbf{x}_j, h) V(\mathbf{x}_j)$

  $\nabla_x A(\mathbf{x}_i) \approx \sum_j \frac{m(\mathbf{x}_j)}{\rho(\mathbf{x}_j)} A(\mathbf{x}_j) \nabla W(\mathbf{x}_i - \mathbf{x}_j, h)$

UNI FREIBURG

Spatial Derivatives

- original forms

  $\nabla A_i = \sum_j \frac{m_j}{\rho_j} A_j \nabla W_{ij}$

  $\nabla^2 A_i = \sum_j \frac{m_j}{\rho_j} A_j \nabla^2 W_{ij}$
- however, resulting forces do not preserve momentum and are not necessarily zero for constant values $A_i = A_j$

UNI FREIBURG

Gradient (Anti-symmetric)

- momentum-preserving form

$$\nabla\left(\frac{A_i}{\rho_i}\right) = \frac{\rho_i \nabla A_i - A_i \nabla \rho_i}{\rho_i^2} = \frac{\nabla A_i}{\rho_i} - \frac{A_i \nabla \rho_i}{\rho_i^2}$$

$$\nabla A_i = \rho_i \left(\nabla\left(\frac{A_i}{\rho_i}\right) + \frac{A_i \nabla \rho_i}{\rho_i^2}\right)$$

- SPH approximation

$$\nabla A_i = \rho_i \left(\sum_j \frac{m_j}{\rho_j}\frac{A_j}{\rho_j}\nabla W_{ij} + A_i \sum_j \frac{m_j}{\rho_j}\frac{\rho_j}{\rho_i^2}\nabla W_{ij}\right)$$

$$= \rho_i \sum_j m_j \left(\frac{A_i}{\rho_i^2} + \frac{A_j}{\rho_j^2}\right)\nabla W_{ij}$$

- applied to pressure gradient, linear and angular momentum is preserved for arbitrary samplings

example with two particles i and j

$$\mathbf{a}_i = m_j \left(\frac{A_i}{\rho_i^2} + \frac{A_j}{\rho_j^2}\right)\nabla W_{ij} = -m_i \left(\frac{A_j}{\rho_j^2} + \frac{A_i}{\rho_i^2}\right)\nabla W_{ji} = -\mathbf{a}_j \qquad \nabla W_{ij} = -\nabla W_{ji}$$

UNI FREIBURG

Gradient (Symmetric)

- term that vanishes for constant function values

$$\nabla\left(\rho_i A_i\right) = \rho_i \nabla\left(A_i\right) + A_i \nabla\left(\rho_i\right)$$

$$\nabla A_i = \frac{1}{\rho_i}\left(\nabla\left(\rho_i A_i\right) - A_i \nabla \rho_i\right)$$

- SPH approximation

$$\nabla A_i = \frac{1}{\rho_i}\left(\sum_j \frac{m_j}{\rho_j}\frac{A_j}{\rho_j}\nabla W_{ij} - A_i \sum_j \frac{m_j}{\rho_j}\rho_j \nabla W_{ij}\right)$$

$$= \frac{1}{\rho_i}\sum_j m_j\left(A_j - A_i\right)\nabla W_{ij} = \frac{1}{\rho_i}\sum_j m_j A_{ji} \nabla W_{ij}$$

- applied to velocity divergence, zero divergence for a constant velocity field is obtained for arbitrary samplings

UNI FREIBURG

Laplacian

- second derivative is error prone and sensitive to particle disorder
- too few samples to appropriately approximate the second kernel derivative
- therefore, the Laplacian is typically approximated with a finite difference approximation of the first derivative

$$\nabla^2 A_i = 2 \sum_j \frac{m_j}{\rho_j} A_{ij} \frac{\mathbf{x}_{ij} \cdot \nabla W_{ij}}{\mathbf{x}_{ij} \cdot \mathbf{x}_{ij} + 0.01h^2}$$

$$A_{ij} = A_i - A_j$$

$$\mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j$$

UNI FREIBURG

Spatial Derivatives - Summary

- original approximations

$$\nabla A_i = \sum_j \frac{m_j}{\rho_j} A_j \nabla W_{ij}$$

$$\nabla^2 A_i = \sum_j \frac{m_j}{\rho_j} A_j \nabla^2 W_{ij}$$

- currently preferred approximations
  - improved robustness in case of particle disorder, i.e. $\sum_j \nabla W_{ij} \neq \mathbf{0}$
  - $\nabla p_i = \rho_i \sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) \nabla W_{ij}$ — preserves linear and angular momentum
  - $\nabla^2 \mathbf{v}_i = 2 \sum_j \frac{m_j}{\rho_j} \mathbf{v}_{ij} \frac{\mathbf{x}_{ij} \cdot \nabla W_{ij}}{\mathbf{x}_{ij} \cdot \mathbf{x}_{ij} + 0.01h^2}$ — improved robustness as it avoids the second kernel derivative
  - $\nabla \cdot \mathbf{v}_i = -\frac{1}{\rho_i} \sum_j m_j \mathbf{v}_{ij} \nabla W_{ij}$ — zero for uniform velocity field
  - $\mathbf{v}_{ij} = \mathbf{v}_i - \mathbf{v}_j \qquad \mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j$

UNI FREIBURG

Kernel Properties

- in case of ideal sampling
- $\rho_i = \sum_j m_j W_{ij} = m_i \sum_j W_{ij}$ $\qquad m_i = m_j$
- $m_i \sum_j W_{ij} = \rho_i = \frac{m_i}{V_i} \quad \Rightarrow \quad \sum_j W_{ij} = \frac{1}{V_i}$
- $\nabla W_{ij} = -\nabla W_{ji} \quad \nabla W_{ij} = \alpha \frac{\mathbf{x}_{ij}}{\|\mathbf{x}_{ij}\| h} \cdots$
- $\sum_j \nabla W_{ij} = \mathbf{0}$

UNI FREIBURG

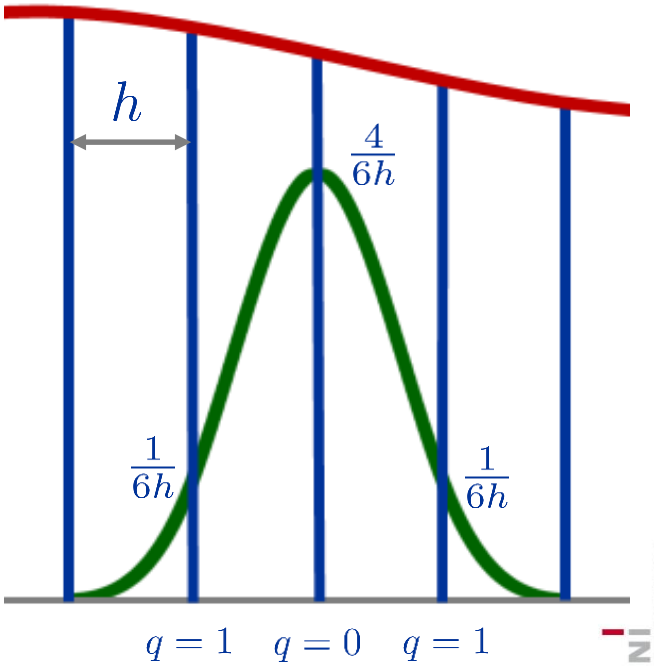
Kernel Illustration

- 1D illustration

- $$W(q) = \frac{1}{6h} \begin{cases} (2-q)^3 - 4(1-q)^3 & 0 \le q < 1 \\ (2-q)^3 & 1 \le q < 2 \\ 0 & q \ge 2 \end{cases} \qquad q = \frac{\|\mathbf{x}_i - \mathbf{x}_j\|}{h}$$

- $$W(0) = \frac{1}{6h}\left((2-0)^3 - 4(1-0)^3\right) = \frac{4}{6h}$$
  $$W(1) = \frac{1}{6h}(2-1)^3 = \frac{1}{6h}$$
  $$W(2) = 0$$

- $$\sum_j W_{ij} = W(0) + 2W(1) + 2W(2) = \frac{1}{h}$$



Kernel Illustration

- 2D illustration

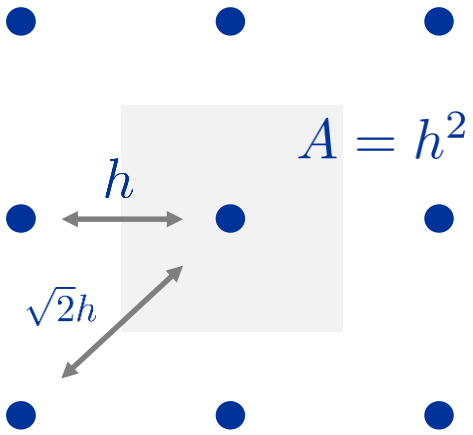
$$W(q) = \frac{5}{14\pi h^2} \begin{cases} (2-q)^3 - 4(1-q)^3 & 0 \le q < 1 \\ (2-q)^3 & 1 \le q < 2 \\ 0 & q \ge 2 \end{cases} \qquad q = \frac{\|\mathbf{x}_i - \mathbf{x}_j\|}{h}$$

$$W(0) = \frac{5}{14\pi h^2}\left((2-0)^3 - 4(1-0)^3\right) = \frac{20}{14\pi h^2}$$

$$W(1) = \frac{5}{14\pi h^2}(2-1)^3 = \frac{5}{14\pi h^2}$$

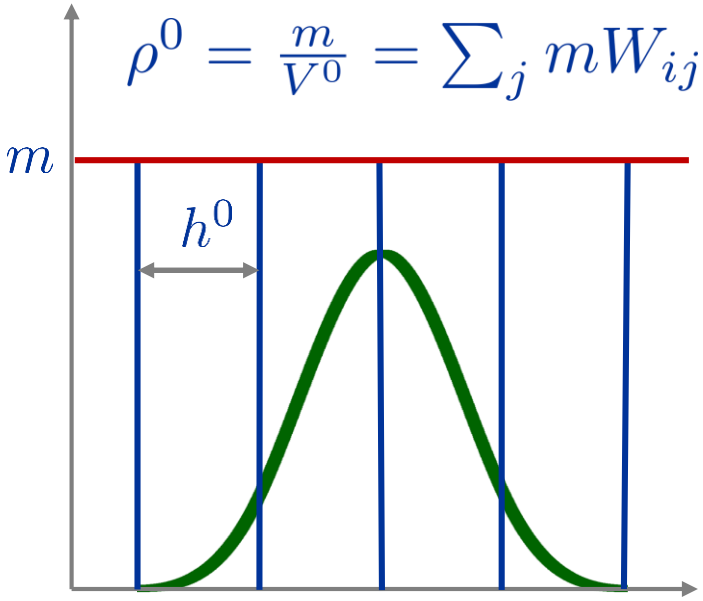
$$W(\sqrt{2}) = \frac{5}{14\pi h^2}(2-\sqrt{2})^3 \approx \frac{1.005}{14\pi h^2}$$

$$\sum_j W_{ij} = W(0) + 4W(1) + 4W(\sqrt{2}) \approx \frac{1.001}{h^2}$$

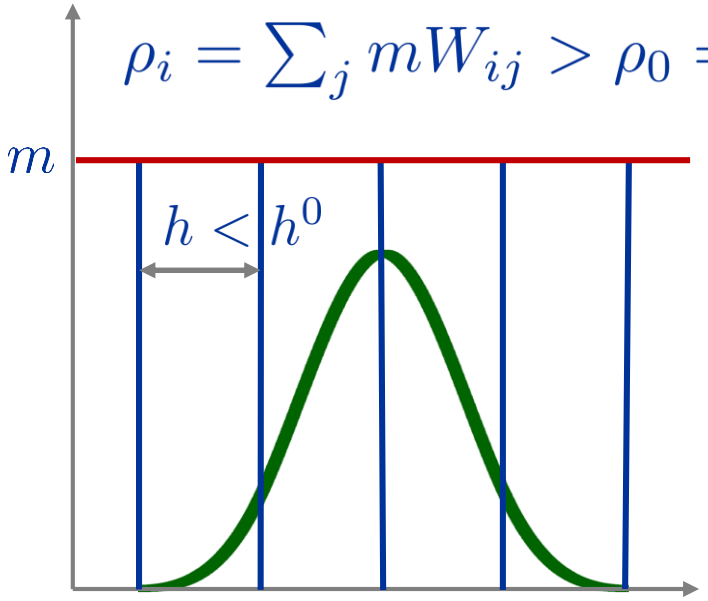


Kernel Illustration

- density computation
  - is not an interpolation of the function m, but detects erroneous sampling



correct sampling



dense sampling
(kernel contributions do not sum up to 1/V)

UNI FREIBURG

Simulation in Computer Graphics

Particle-based Fluid Simulation

Matthias Teschner

Computer Science Department
University of Freiburg

UNI FREIBURG

Simple SPH Fluid Solver

- **for all** *particle* i **do**
  find neighbors j
  for all *particle* i **do**
  $\rho_i = \sum_j m_j W_{ij}$
  compute p_i from ρ_i
  for all *particle* i **do**

$$\mathbf{a}_i^{pressure} = -\frac{1}{\rho_i}\nabla p_i$$
$$\mathbf{a}_i^{viscosity} = \nu\nabla^2\mathbf{v}_i$$
$$\mathbf{a}_i^{other} = \mathbf{g}$$
$$\mathbf{a}_i(t) = \mathbf{a}_i^{pressure} + \mathbf{a}_i^{viscosity} + \mathbf{a}_i^{other}$$

Navier-Stokes equation

  for all *particle* i **do**
  $\mathbf{v}_i(t+\Delta t) = \mathbf{v}_i(t) + \Delta t\mathbf{a}_i(t)$
  $\mathbf{x}_i(t+\Delta t) = \mathbf{x}_i(t) + \Delta t\mathbf{v}_i(t+\Delta t)$

$$\mathbf{a}_i^{pressure} = -\sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}\right)\nabla W_{ij}$$
$$\mathbf{a}_i^{viscosity} = 2\nu\sum_j \frac{m_j}{\rho_j}\mathbf{v}_{ij}\frac{\mathbf{x}_{ij}\cdot\nabla W_{ij}}{\mathbf{x}_{ij}\cdot\mathbf{x}_{ij}+0.01h^2}$$
$$\mathbf{a}_i^{other} = \mathbf{g}$$

SPH approximations

UNI FREIBURG

Outline

- concept of an SPH fluid simulator
- momentum equation
- SPH basics
- neighborhood search
- boundary handling
- incompressibility
- surface reconstruction

UNI FREIBURG

SPH Simulation Step Using a State Equation (SESPH)

- foreach particle do
  - compute density
  - compute pressure
- foreach particle do
  - compute forces
  - update velocities and positions

- density and force computation
  process all neighbors of a particle

UNI FREIBURG

Neighbor Search

- for the computation of SPH sums in 3D, each particle needs to know at least 30-40 neighbors in each simulation step
- current scenarios
  - up to 30 million fluid particles
  - up to 1 billion neighbors
  - up to 10000 simulation steps
  - up to 10^{13} neighbors processed per simulation
- efficient construction and processing of dynamically changing neighbor sets is essential

UNI FREIBURG

Motivation

up to 30 million fluid particles, up to 1 billion neighbors,
11 s computation time for neighbor search on a 16-core PC

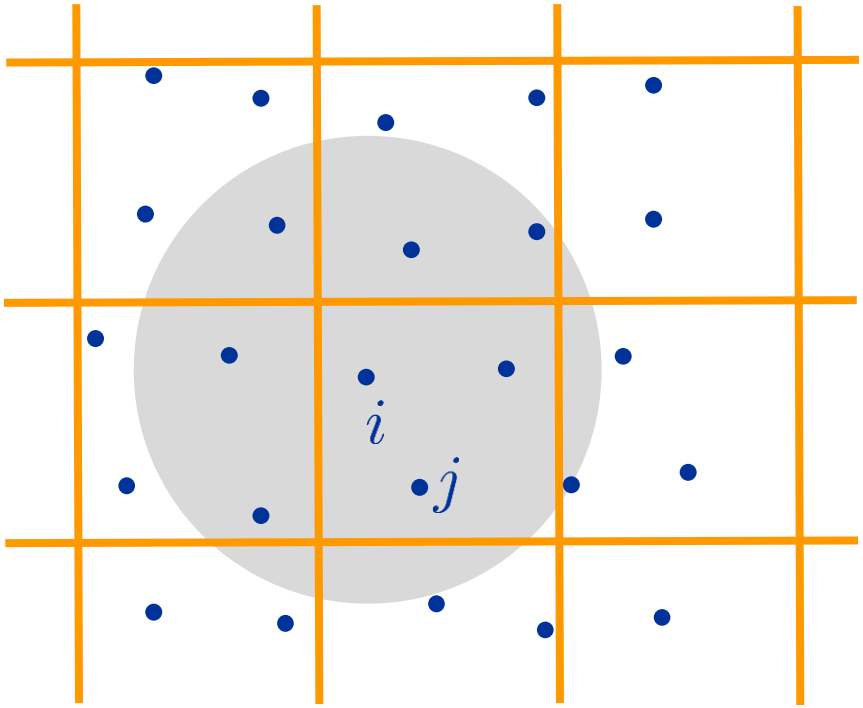
Characteristics

- SPH computes sums
  - dynamically changing sets of neighboring particles
  - temporal coherence
- spatial data structures accelerate the neighbor search
  - fast query
  - fast generation (at least once for each simulation step)
  - sparsely, non-uniformly filled simulation domain
- space subdivision
  - each particle is placed in a convex space cell, e.g. a cube
- similarities to collision detection and intersection tests in raytracing
  - however, cells adjacent to the cell of a particle have to be accessed

UNI FREIBURG

Characteristics

- hierarchical data structures are less efficient
  - construction in O (n log n), access in O (log n)
- uniform grid is generally preferred
  - construction in O (n), access in O (1)



UNI FREIBURG

Characteristics

- Verlet lists
  - motivated by temporal coherence
  - potential neighbors are computed within a distance larger than the actual kernel support
  - actual neighbors are computed from the set of potential neighbors
  - potential neighbors are updated every n-th simulation step
  - memory-intensive (processes more neighbors than a standard grid)
- storing neighbors is generally expensive
  - might be avoided for, e.g., a low number of neighbor queries per simulation step or in case of very efficient computation
- data structures have to process
  - fluid particles of multiple phases, e.g. air
  - rigid particles (static or moving)
  - deformable particles

UNI FREIBURG

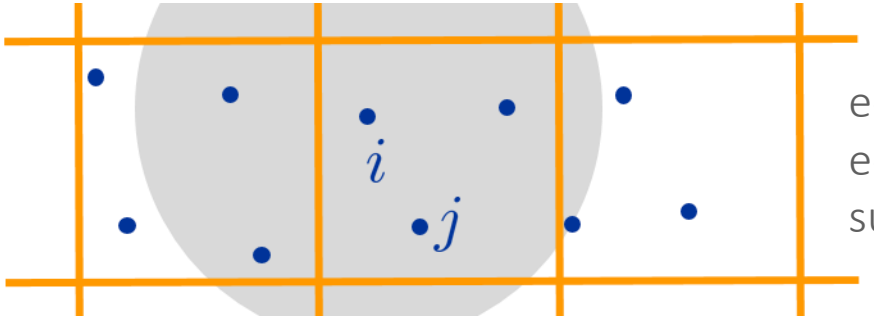
Outline

- concept of an SPH fluid simulator
- momentum equation
- SPH basics
- neighborhood search
  - uniform grid
  - index sort
  - spatial hashing
  - discussion
- boundary handling
- incompressibility
- surface reconstruction

UNI FREIBURG

Basic Grid

- particle is stored in a cell with coordinates (k, l, m)
- 27 cells are queried in the neighborhood search (k±1, l±1, m±1)
- cell size equals the kernel support of a particle
  - larger cells increase the number of tested particles
  - smaller cells increase the number of tested cells
- parallel construction suffers from race conditions
  - insertion of particles from different threads in the same cell



edge length equals kernel support

UNI FREIBURG

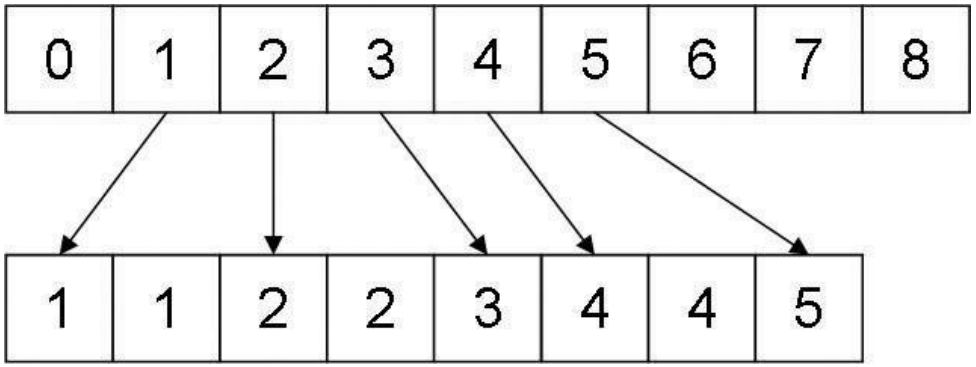
Outline

- concept of an SPH fluid simulator
- momentum equation
- SPH basics
- neighborhood search
  - uniform grid
  - index sort
  - spatial hashing
  - discussion
- incompressibility
- boundary handling
- surface reconstruction

UNI FREIBURG

Construction

- cell index $c = k + l \cdot K + m \cdot K \cdot L$
  is computed for a particle
  - K and L denote the number of cells in x and y direction
- particles are sorted with respect to their cell index
  - e.g., radix sort, O(n)
- each grid cell (k, l, m) stores a reference
  to the first particle in the sorted list

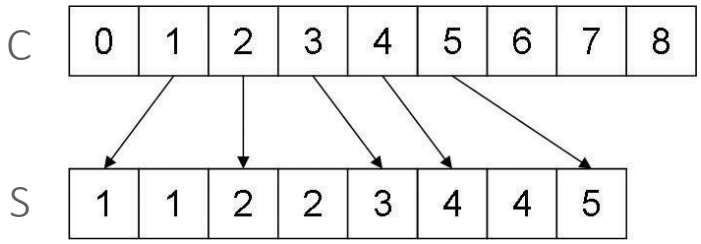


uniform grid

sorted particles with
their cell indices

Construction

- generate C
- store the number of particles in each cell of C
  - loop over all particles and increment the respective value in C
- accumulate the values in C
- generate S
- associate particle i with cell j:
  S [--C [j]] = i
  - stores the particles
    in reversed order into S
  - after insertion C contains the correct offsets



[Lagae]

Construction

- parallelizable
- memory allocations are avoided
- constant memory consumption
- entire spatial grid has to be represented to find neighboring cells

UNI FREIBURG

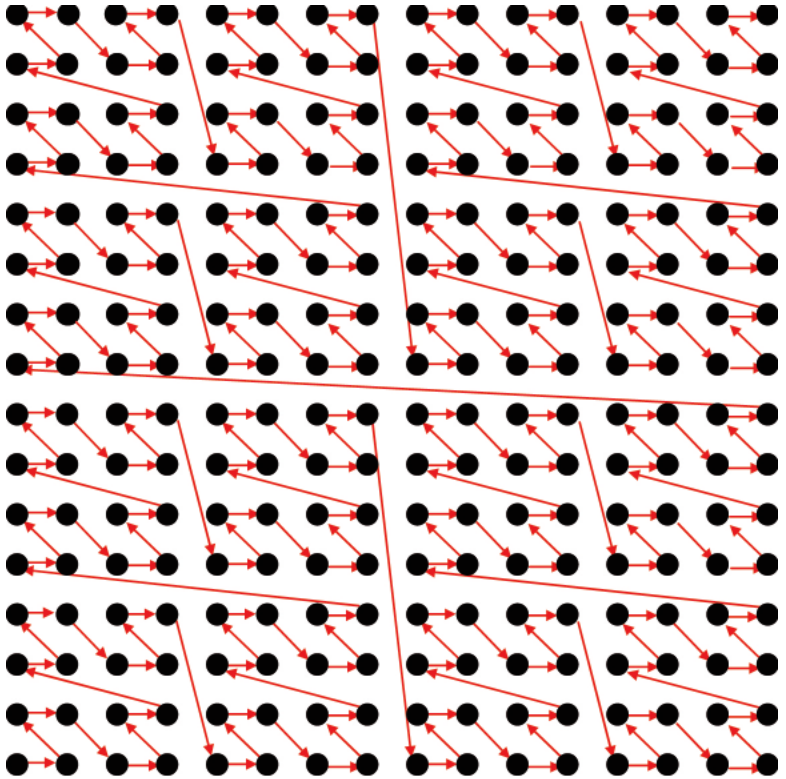
Query

- sorted particle array is queried
  - parallelizable
- particles in the same cell are queried
- references to particles of adjacent cells are obtained from the references stored in the uniform grid
- improved cache-hit rate
  - particles in the same cell are close in memory
  - particles of neighboring cells are not necessarily close in memory

UNI FREIBURG

Z-Index Sort

- particles are sorted with respect to a z-curve index
- improved cache-hit rate
  - particles in adjacent cells are close in memory
- efficient computation of z-curve indices possible



z-curve

UNI FREIBURG

Z-Index Sort - Sorting

- particle attributes and z-curve indices can be processed separately
- handles (particle identifier, z-curve index) are sorted in each time step
  - reduces memory transfer
  - spatial locality is only marginally influenced due to temporal coherence
- attribute sets are sorted every n^{th} simulation step
  - restores spatial locality

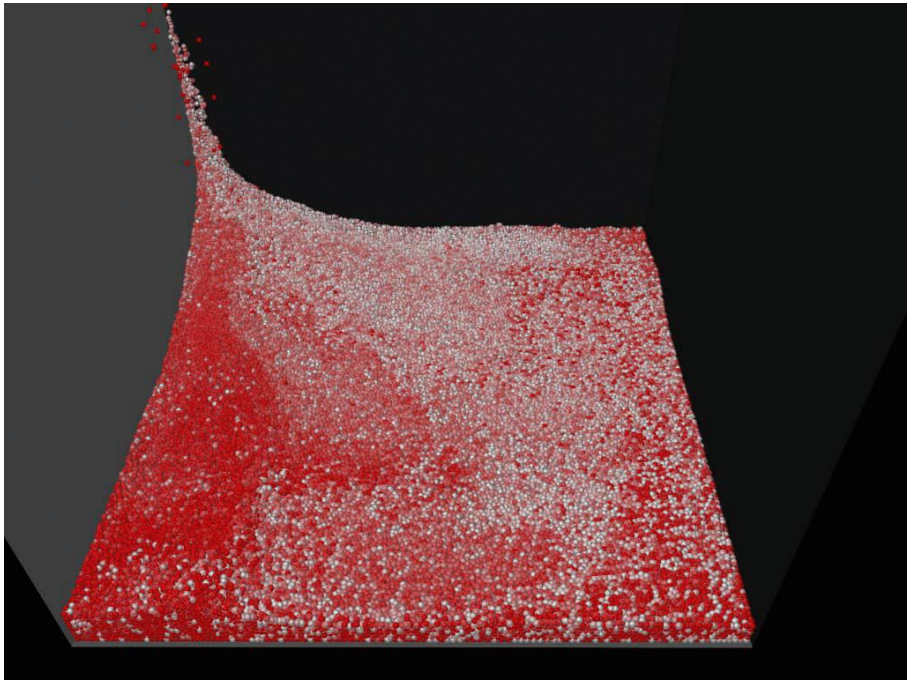
UNI FREIBURG

Z-Index Sort - Sorting

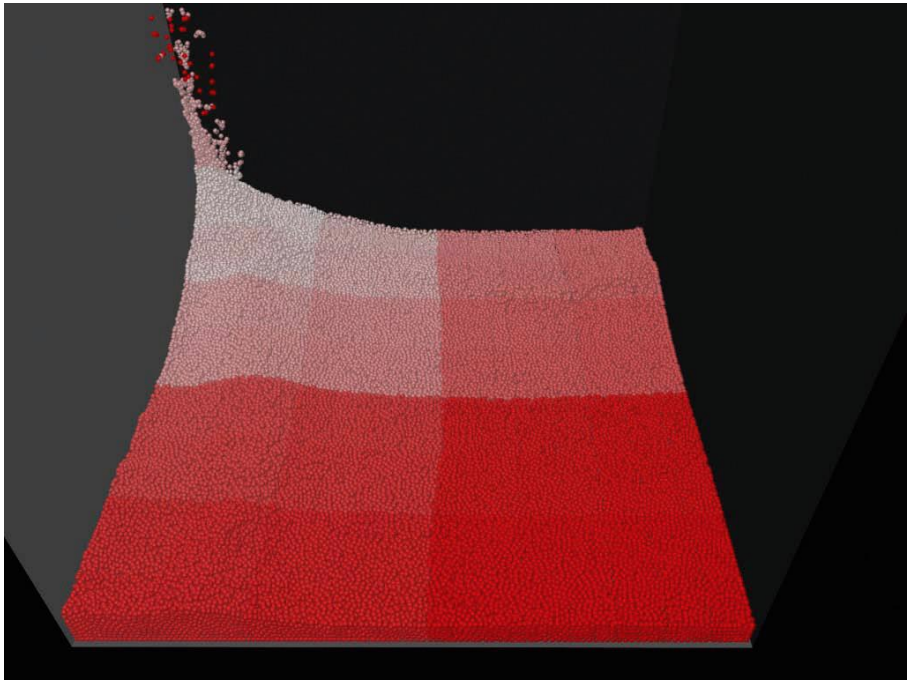
- instead of radix sort, insertion sort can be employed
  - O (n) for almost sorted arrays
  - due to temporal coherence, only 2% of all particles change their cell, i.e. z-curve index, in each time step

UNI FREIBURG

Z-Index Sort - Reordering



particles colored according
to their location in memory



spatial compactness is
enforced using a z-curve

UNI FREIBURG

Outline

- concept of an SPH fluid simulator
- momentum equation
- SPH basics
- neighborhood search
  - uniform grid
  - index sort
  - spatial hashing
  - discussion
- boundary handling
- incompressibility
- surface reconstruction

UNI FREIBURG

Spatial Hashing

- hash function maps a grid cell to a hash cell
  - infinite 3D domain is mapped to a finite 1D list
  - in contrast to index sort, infinite domains can be handled

- implementation
  - compute a cell index c or a cell identifier (x, y, z) for a particle
  - compute a hash function $i = h(c)$ or $i = h(x, y, z)$
  - store the particle in a 1D array (hash table) at index i

UNI FREIBURG

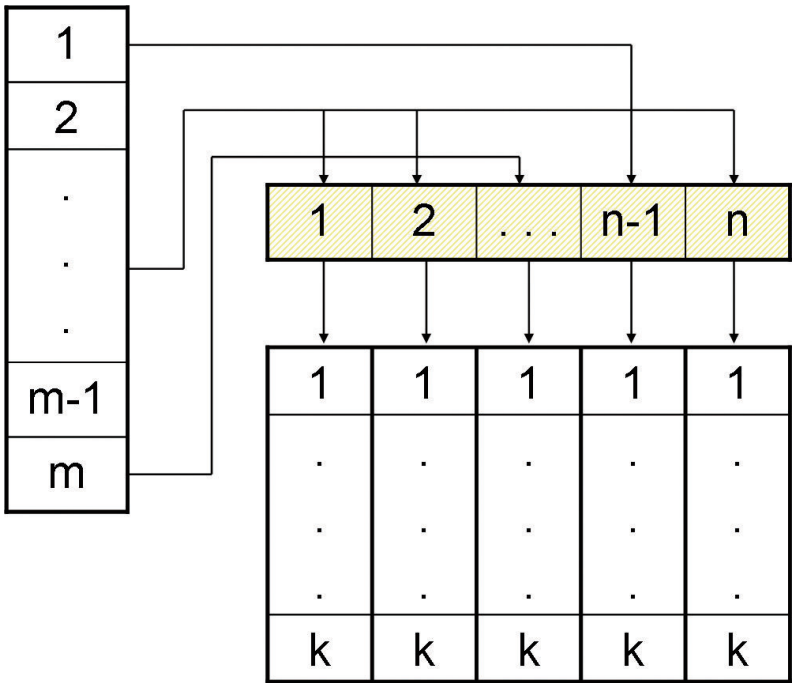
Spatial Hashing

- large hash tables reduce number of hash collisions
  - hash collisions occur, if different spatial cells are mapped to the same hash cell
  - hash collisions slow down the query
- reduced memory allocations
  - memory for a certain number of entries is allocated for each hash cell
- reduced cache-hit rate
  - hash table is sparsely filled
  - filled and empty cells are alternating

UNI FREIBURG

Compact Hashing

- hash cells store handles to a compact list of used cells
  - k entries are pre-allocated for each element in the list of used cells
  - elements in the used-cell list are generated if a particle is placed in a new cell
  - elements are deleted, if a cell gets empty
- memory consumption is reduced from O (m · k) to O (m + n · k) with m » n
- list of used cells is queried in the neighbor search



Compact Hashing - Construction

- not parallelizable
  - particles from different threads might be inserted in the same cell
- larger hash table compared to spatial hashing to reduce hash collisions
- temporal coherence is employed
  - list of used cells is not rebuilt, but updated
  - set of particles with changed cell index is estimated (about 2% of all particles)
  - particle is removed from the old cell and added to the new cell (not parallelizable)

UNI FREIBURG

Compact Hashing - Query

- processing of used cells
  - bad spatial locality
  - used cells close in memory are not close in space
- hash-collision flag
  - if there is no hash collision in a cell, hash indices of adjacent cells have to be computed only once for all particles in this cell
  - large hash table results in 2% cells with hash collisions

UNI FREIBURG

Compact Hashing - Query

- particles are sorted with respect to a z-curve every n^{th} step
- after sorting, the list of used cells has to be rebuilt
- if particles are serially inserted into the list of used cells, the list is consistent with the z-curve
  - improved cache hit rate during the traversal of the list of used cells

UNI FREIBURG

Compact Hashing - Reordering

UNI FREIBURG

Outline

- concept of an SPH fluid simulator
- momentum equation
- SPH basics
- neighborhood search
  - uniform grid
  - index sort
  - spatial hashing
  - discussion
- boundary handling
- incompressibility
- surface reconstruction

UNI FREIBURG

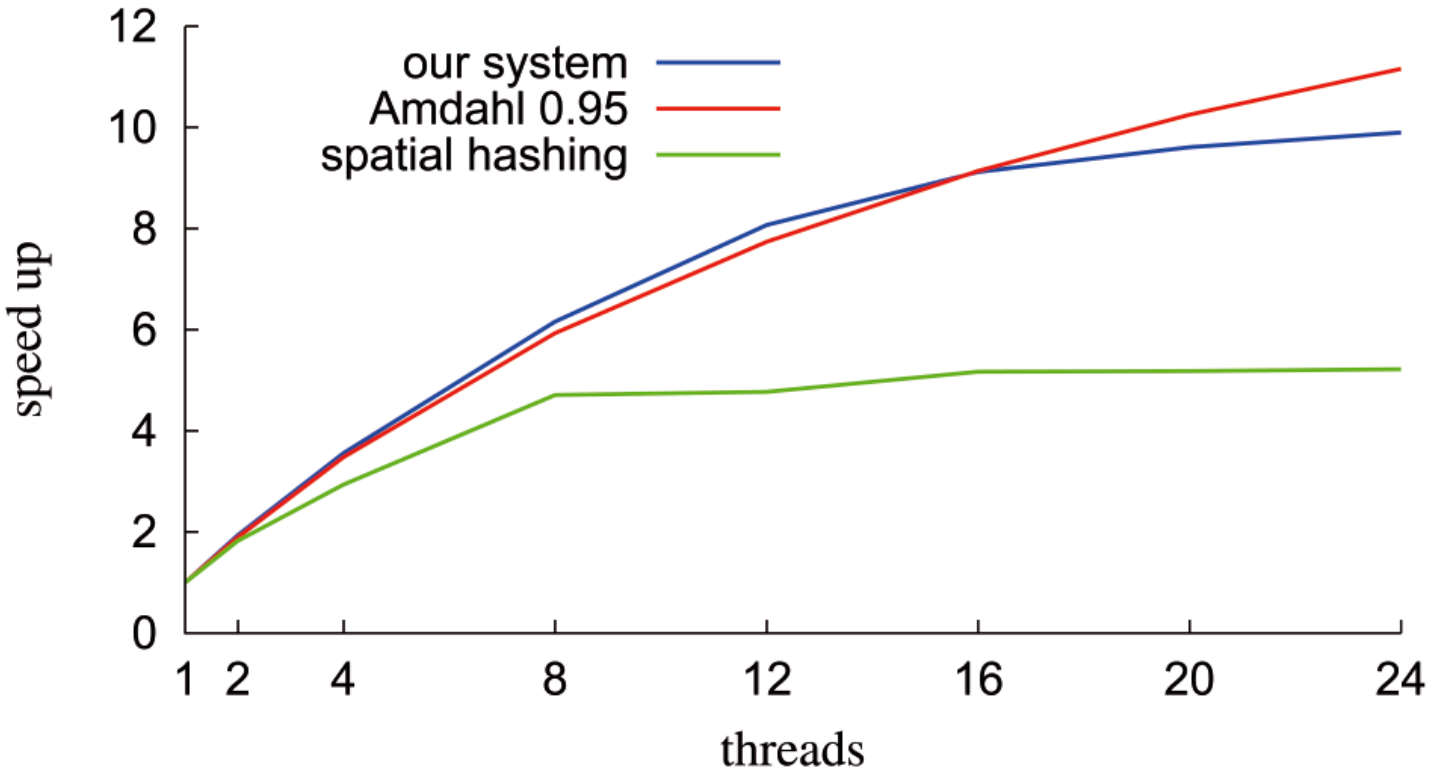
Comparison

method	construction	query	total
basic grid	26	38	64
index sort	36	29	65
z-index sort	16	27	43
spatial hashing	42	86	128
compact hashing	8	32	40

- measurements in ms for 130K particles
- ongoing research
  - currently, compact hashing is used, compact list stores references to a sorted particle list

UNI FREIBURG

Parallel Scaling



UNI FREIBURG

Discussion

- index sort
  - fast query as particles are processed in the order of cell indices
- z-index sort
  - fast construction due to radix sort or insertion sort of an almost sorted list
  - sorting with respect to the z-curve improves cache-hit rate
- spatial hashing
  - slow query due to hash collisions and due to the traversal of the sparsely filled hash table
- compact hashing
  - fast construction (update) due to temporal coherence
  - fast query due to the compact list of used cells, due to the hash-collision flag and due to z-curve

UNI FREIBURG

References

- z-index sort, compact hashing
  - IHMSEN M., AKINCI N., BECKER M., TESCHNER M.: A Parallel SPH Implementation on Multi-core CPUs. *Computer Graphics Forum*, 2011.
- index sort
  - PURCELL T. J., DONNER C., CAMMARANO M., JENSEN H. W., HANRAHAN P.: Photon Mapping on Programmable Graphics Hardware. *ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware,* 2003.
- spatial hashing
  - TESCHNER M., HEIDELBERGER B., MÜLLER M., POMERANETS D., GROSS M.: Optimized Spatial Hashing for Collision Detection of Deformable Objects. *Vision, Modeling, Visualization*, 2003.

UNI FREIBURG

Simulation in Computer Graphics

Particle-based Fluid Simulation

Matthias Teschner

Computer Science Department
University of Freiburg

UNI FREIBURG

Simple SPH Fluid Solver

- **for all** *particle* i **do**
  find neighbors j
  for all *particle* i **do**
  $\rho_i = \sum_j m_j W_{ij}$
  compute p_i from ρ_i
  for all *particle* i **do**

$$\mathbf{a}_i^{pressure} = -\frac{1}{\rho_i}\nabla p_i$$
$$\mathbf{a}_i^{viscosity} = \nu\nabla^2\mathbf{v}_i$$
$$\mathbf{a}_i^{other} = \mathbf{g}$$
$$\mathbf{a}_i(t) = \mathbf{a}_i^{pressure} + \mathbf{a}_i^{viscosity} + \mathbf{a}_i^{other}$$

Navier-Stokes equation

  for all *particle* i **do**
  $\mathbf{v}_i(t+\Delta t) = \mathbf{v}_i(t) + \Delta t\mathbf{a}_i(t)$
  $\mathbf{x}_i(t+\Delta t) = \mathbf{x}_i(t) + \Delta t\mathbf{v}_i(t+\Delta t)$

$$\mathbf{a}_i^{pressure} = -\sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}\right)\nabla W_{ij}$$
$$\mathbf{a}_i^{viscosity} = 2\nu\sum_j \frac{m_j}{\rho_j}\mathbf{v}_{ij}\frac{\mathbf{x}_{ij}\cdot\nabla W_{ij}}{\mathbf{x}_{ij}\cdot\mathbf{x}_{ij}+0.01h^2}$$
$$\mathbf{a}_i^{other} = \mathbf{g}$$

SPH approximations

UNI FREIBURG

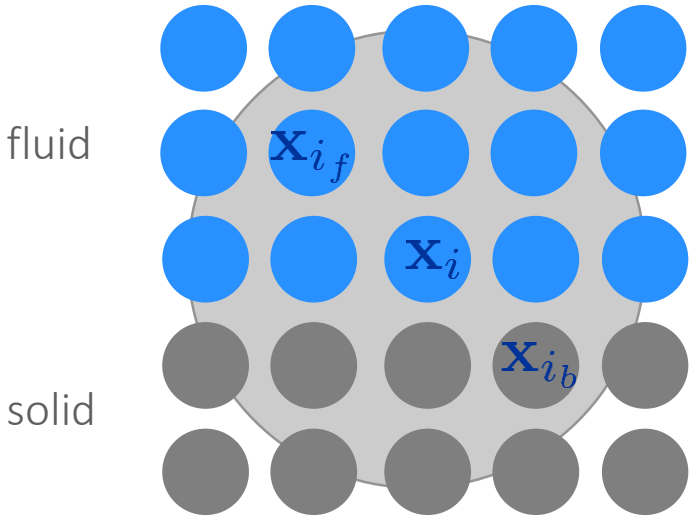
Outline

- concept of an SPH fluid simulator
- momentum equation
- SPH basics
- neighborhood search
- boundary handling
- incompressibility
- surface reconstruction

UNI FREIBURG

Concept

- rigids are uniformly sampled with particles



$$\rho_i = \sum_{i_f} m_{i_f} W_{ii_f} + \sum_{i_b} m_{i_b} W_{ii_b}$$

$$m_i = m_{i_f} = m_{i_b}$$

$$\rho_i \approx m_i \sum_{i_f} W_{ii_f} + m_i \sum_{i_b} W_{ii_b}$$

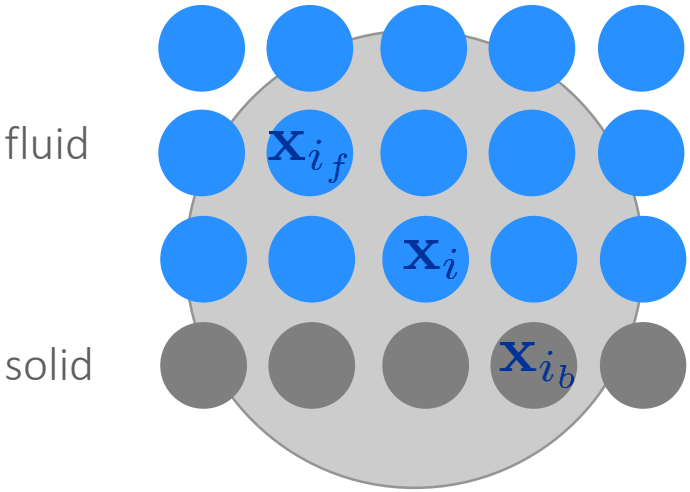
$$p_i = \dots$$

$$\mathbf{a}_i^p = -\sum_{i_f} m_{i_f} \left(\frac{p_i}{\rho_i^2} + \frac{p_{i_f}}{\rho_{i_f}^2} \right) \nabla W_{ii_f} - \sum_{i_b} m_{i_b} \left(\frac{p_i}{\rho_i^2} + \frac{p_{i_b}}{\rho_{i_b}^2} \right) \nabla W_{ii_b}$$

$$\mathbf{a}_i^p \approx -m_i \sum_{i_f} \left(\frac{p_i}{\rho_0^2} + \frac{p_{i_f}}{\rho_0^2} \right) \nabla W_{ii_f} - m_i \sum_{i_b} \left(\frac{p_i}{\rho_0^2} + \frac{p_i}{\rho_0^2} \right) \nabla W_{ii_b}$$

Missing Contributions

- rigids are uniformly sampled with particles



$$\rho_i \approx m_i \sum_{i_f} W_{ii_f} + m_i \sum_{i_b} W_{ii_b} + x$$

$$\rho_i \approx m_i \sum_{i_f} W_{ii_f} + m_i \gamma_1 \sum_{i_b} W_{ii_b}$$

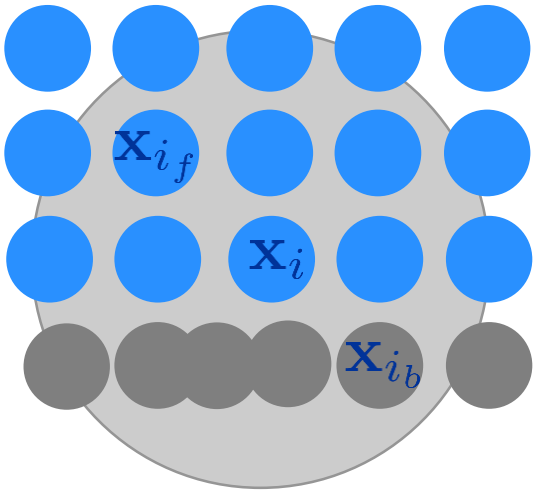
$$\sum_{i_f} W_{ii_f} + \gamma_1 \sum_{i_b} W_{ii_b} = \frac{1}{V_i}$$

$$\mathbf{a}_i^p \approx -m_i \sum_{i_f} \left(\frac{p_i}{\rho_0^2} + \frac{p_{i_f}}{\rho_0^2} \right) \nabla W_{ii_f} - m_i \gamma_2 \sum_{i_b} \left(\frac{p_i}{\rho_0^2} + \frac{p_i}{\rho_0^2} \right) \nabla W_{ii_b}$$

$$\sum_{i_f} \nabla W_{ii_f} + \gamma_2 \sum_{i_b} \nabla W_{ii_b} = \mathbf{0}$$

Non-uniform Sampling

- rigids are non-uniformly sampled with particles



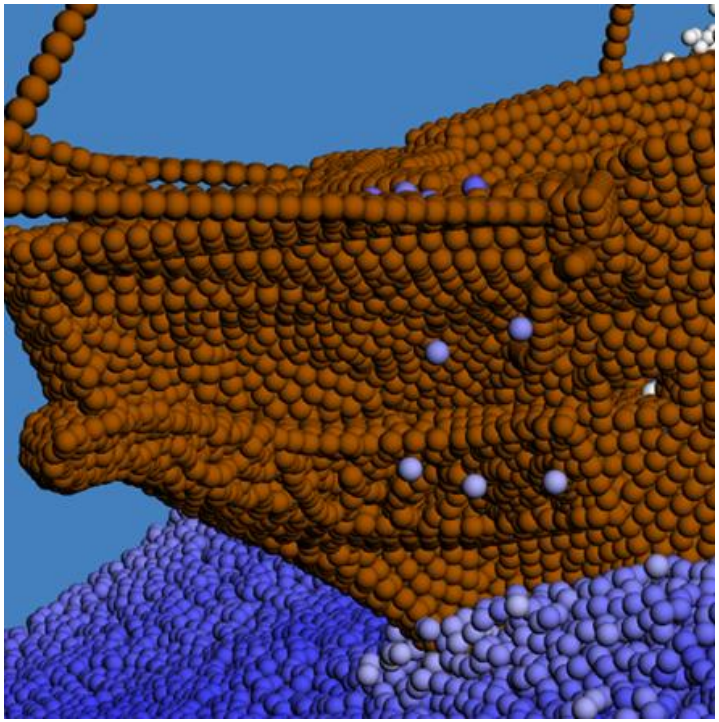
$$\rho_i \approx m_i \sum_{i_f} W_{ii_f} + \gamma_1 \sum_{i_b} m_{i_b} W_{ii_b}$$

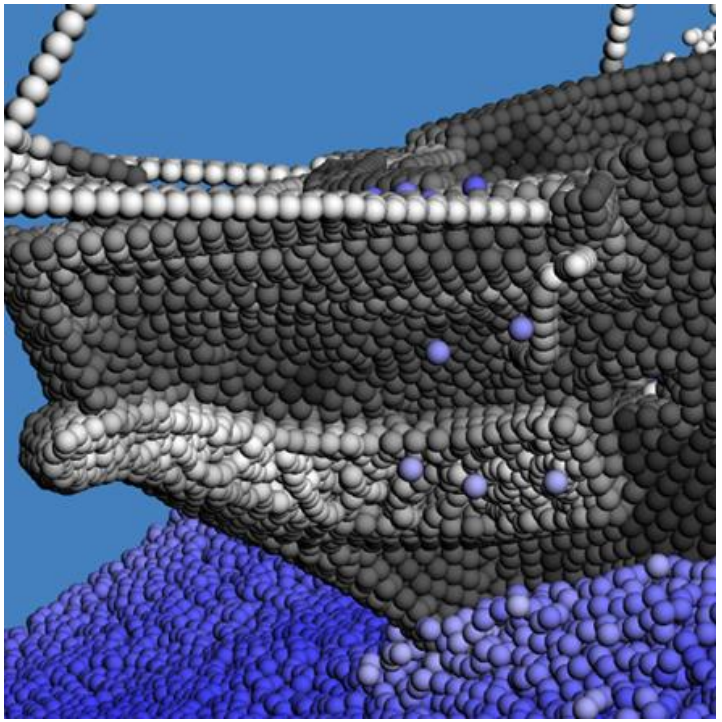
$$\frac{1}{V_{i_b}} \approx \gamma_3 \sum_{i_{b_b}} W_{i_b i_{b_b}}$$

$$m_{i_b} = \frac{\rho_0}{\gamma_3 \sum_{i_{b_b}} W_{i_b i_{b_b}}}$$

$$\mathbf{a}_i^p \approx -m_i \sum_{i_f} \left(\frac{p_i}{\rho_0^2} + \frac{p_{i_f}}{\rho_0^2} \right) \nabla W_{ii_f} - \gamma_2 \sum_{i_b} m_{i_b} \left(\frac{p_i}{\rho_0^2} + \frac{p_i}{\rho_0^2} \right) \nabla W_{ii_b}$$

Non-uniform Sampling





color-coded volume
of boundary particles

UNI FREIBURG

Rigid-Fluid Coupling

www.youtube.com/cgfreiburg

UNI FREIBURG

Rigid-Fluid Coupling

www.youtube.com/cgfreiburg

References

- rigid-fluid coupling
  - IHMSEN M., AKINCI N., GISSLER M., TESCHNER M.: Boundary Handling and Adaptive Time-stepping for PCISPH. *VRIPHYS*, 2010.
  - AKINCI N., IHMSEN M., AKINCI G., SOLENTHALER B., TESCHNER M.: Versatile Rigid-Fluid Coupling for Incompressible SPH. *ACM TOG (SIGGRAPH 2012)*, 2012.
- elastic-fluid coupling
  - AKINCI N., CORNELIS J., AKINCI G., TESCHNER M.: Coupling Elastic Solids with Smoothed Particle Hydrodynamics Fluids. *Journal of Computer Animation and Virtual Worlds (CASA 2013),* 2013.

UNI FREIBURG

Outline

- concept of an SPH fluid simulator
- momentum equation
- SPH basics
- neighborhood search
- boundary handling
- incompressibility
- surface reconstruction

UNI FREIBURG

Incompressibility

- is essential for a realistic fluid behavior
  - less than 0.1% in typical scenarios
- inappropriate compression leads, e.g., to oscillations at the free surface
- compression is time-step dependent
  - volume changes should be imperceptible in adaptive time-stepping schemes
- is computationally expensive
  - simple computations require small time steps
  - large time steps require complex computations

UNI FREIBURG

State Equations (EOS, SESPH)

- pressure forces resolve compression induced by non-pressure forces (penalty approach)
  - density fluctuations in the fluid result in density gradients
  - density gradients result in pressure gradients
  - pressure gradients result in pressure force from high to low pressure
- fast computation, but small time steps
- pressure is computed from density, e.g.
  - $p_i = k(\rho_i - \rho_0)$ $p_i = k\rho_i$ $p_i = k\left(\frac{\rho_i}{\rho_0} - 1\right)$ k is user-defined
    - in graphics referred to as compressible SPH
  - $p_i = k_1\left(\left(\frac{\rho_i}{\rho_0}\right)^{k_2} - 1\right)$ k_1, k_2 are user-defined
    - in graphics referred to as weakly compressible SPH
  - compressibility is governed by the stiffness constant(s) and limits the time step

UNI FREIBURG

Non-iterative EOS solver (SESPH)

- **for all** *particle* i **do**
  find neighbors j

  for all *particle* i **do**
  $\rho_i = \sum_j m_j W_{ij}$
  compute p_i from ρ_i with a state equation

  for all *particle* i **do**
  $\mathbf{F}_i^{pressure} = -\frac{m_i}{\rho_i} \nabla p_i$
  $\mathbf{F}_i^{viscosity} = m_i \nu \nabla^2 \mathbf{v}_i$
  $\mathbf{F}_i^{other} = m_i \mathbf{g}$
  $\mathbf{F}_i(t) = \mathbf{F}_i^{pressure} + \mathbf{F}_i^{viscosity} + \mathbf{F}_i^{other}$

  for all *particle* i **do**
  $\mathbf{v}_i(t + \Delta t) = \mathbf{v}_i(t) + \Delta t \mathbf{F}_i(t)/m_i$
  $\mathbf{x}_i(t + \Delta t) = \mathbf{x}_i(t) + \Delta t \mathbf{v}_i(t + \Delta t)$

UNI FREIBURG

SESPH with Splitting

- compute pressure after advecting the particles with non-pressure forces
- concept
  - compute all non-pressure forces $\mathbf{F}_i^{nonp}(t)$
  - compute intermediate velocity $\mathbf{v}_i^* = \mathbf{v}_i(t) + \Delta t \frac{\mathbf{F}_i^{nonp}}{m_i}$
  - compute intermediate position $\mathbf{x}_i^* = \mathbf{x}_i(t) + \Delta t \mathbf{v}_i^*$
  - compute intermediate density $\rho_i^*(\mathbf{x}_i^*)$
  - compute pressure p_i from intermediate density ρ_i^* using an EOS
  - compute final velocity $\mathbf{v}_i(t + \Delta t) = \mathbf{v}_i^* - \Delta t \frac{1}{\rho_i^*} \nabla p_i$
- motivation
  - consider competing forces
  - take (positive or negative) effects of non-pressure forces into account when computing the pressure forces

UNI FREIBURG

SESPH with Splitting

- **for all** *particle* i **do**
  find neighbors j

  for all *particle* i **do**
  $\mathbf{F}_i^{viscosiy} = m_i \nu \nabla^2 \mathbf{v}_i$
  $\mathbf{F}_i^{other} = m_i \mathbf{g}$
  $\mathbf{v}_i^* = \mathbf{v}_i(t) + \Delta t \frac{\mathbf{F}_i^{viscosity} + \mathbf{F}_i^{other}}{m_i}$

  for all *particle* i **do**
  $\rho_i^* = \sum_j m_j W_{ij} + \Delta t \sum_j m_j (\mathbf{v}_i^* - \mathbf{v}_j^*) \nabla W_{ij}$
  compute p_i using ρ_i^*

  for all *particle* i **do**
  $\mathbf{F}_i^{pressure} = -\frac{m_i}{\rho_i^*} \nabla p_i$

  for all *particle* i **do**
  $\mathbf{v}_i(t + \Delta t) = \mathbf{v}_i^* + \Delta t \mathbf{F}_i^{pressure} / m_i$
  $\mathbf{x}_i(t + \Delta t) = \mathbf{x}_i(t) + \Delta t \mathbf{v}_i(t + \Delta t)$

- follows from the continuity equation
- avoids neighbor search
- see next slide

UNI FREIBURG

Differential Density Update

- continuity equation

$$\frac{\mathrm{D}\rho_i}{\mathrm{D}t} = -\rho_0 \nabla \cdot \mathbf{v}_i$$

velocity divergence corresponds to an in- / outflow at a fluid element which corresponds to a density change

- time discretization

$$\frac{\rho_i^* - \rho_i(t)}{\Delta t} = -\rho_0 \nabla \cdot \mathbf{v}_i^*$$

- space discretization

$$\frac{\rho_i^* - \sum_j m_i W_{ij}}{\Delta t} = -\rho_0 \left(-\frac{1}{\rho_0} \sum_j m_j (\mathbf{v}_i^* - \mathbf{v}_j^*) \nabla W_{ij} \right)$$

- predicted density due to the divergence of $\mathbf{v}_i^*$

$$\rho_i^* = \sum_j m_i W_{ij} + \Delta t \sum_j m_j (\mathbf{v}_i^* - \mathbf{v}_j^*) \nabla W_{ij}$$

UNI FREIBURG

Iterative SESPH with Splitting

- pressure forces are iteratively accumulated and refined
- concept
  - compute non-pressure forces, intermediate velocity and position
  - iteratively
    - compute intermediate density from intermediate position
    - compute pressure from intermediate density
    - compute pressure forces
    - update intermediate velocity and position
- motivation
  - parameterized by a desired density error, not by a stiffness constant
  - provides a fluid state with a guaranteed density error

UNI FREIBURG

Iterative SESPH with Splitting

- **for all** *particle* i **do**
  find neighbors j
  for all *particle* i **do**
  $\mathbf{F}_i^{viscosity} = m_i \nu \nabla^2 \mathbf{v}_i \qquad \mathbf{F}_i^{other} = m_i \mathbf{g}$
  $\mathbf{v}_i^* = \mathbf{v}_i(t) + \Delta t \frac{\mathbf{F}_i^{viscosity} + \mathbf{F}_i^{other}}{m_i} \qquad \mathbf{x}_i^* = \mathbf{x}_i(t) + \Delta t \mathbf{v}_i^*$
  repeat
  for all *particle* i **do**
  compute ρ_i^* using $\mathbf{x}_i^*$
  compute p_i using ρ_i^*, e.g. $p_i = k(\rho_i^* - \rho_0)$
  compute ρ_{err}, e.g. average or maximum
  for all *particle* i **do**
  $\mathbf{F}_i^{pressure} = -\frac{m_i}{\rho_i^*} \nabla p_i \qquad \mathbf{v}_i^* = \mathbf{v}_i^* + \Delta t \frac{\mathbf{F}_i^{pressure}}{m_i} \qquad \mathbf{x}_i^* = \mathbf{x}_i^* + \Delta t^2 \frac{\mathbf{F}_i^{pressure}}{m_i}$
  until $\rho_{err} < \eta$ user-defined density error
  for all *particle* i **do**
  $\mathbf{v}_i(t + \Delta t) = \mathbf{v}_i^* \qquad \mathbf{x}_i(t + \Delta t) = \mathbf{x}_i^*$

UNI FREIBURG

Iterative SESPH - Variants

- different quantities are accumulated
  - pressure forces (local Poisson SPH)
  - pressure (predictive-corrective SPH, PCISPH)
    - advantageous, if pressure is required for other computations
  - distances (position-based fluids, PBF)
    - $\Delta \mathbf{x}_i = -\frac{1}{\rho_0} \sum_j (\frac{p_i}{\beta_i} + \frac{p_j}{\beta_j}) \nabla W_{ij}$ β is a pre-computed constant
- different EOS and stiffness constants are used
  - $k = \frac{\rho_i^* r_i^2}{2\rho_0 \Delta t^2}$ in local Poisson SPH
  - $k = \frac{2\rho_0^2}{m_i^2 \cdot \Delta t^2 \sum_j \nabla W_{ij}^0 \cdot \sum_j \nabla W_{ij}^0 + \sum_j (\nabla W_{ij}^0 \cdot \nabla W_{ij}^0)}$ in PCISPH W^0 is precomputed
  - $k = 1$ in PBF $(p_i = \frac{\rho_i}{\rho_0} - 1)$

UNI FREIBURG

PCISPH - Motivation

- density at the next timestep should be rest density

$$\rho(t+\Delta t) = \rho_0 = \underbrace{\sum_j m_i W_{ij} + \Delta t \sum_j m_j(\mathbf{v}_i^* - \mathbf{v}_j^*)\nabla W_{ij}}_{\rho_i^*} + \Delta t \sum_j m_j(\mathbf{v}_i^p - \mathbf{v}_j^p)\nabla W_{ij}$$

- $\mathbf{v}_i^p = \Delta t \frac{\mathbf{F}_i^p}{m_i}$

$$\mathbf{F}_i^p = -m_i \sum_j (\frac{p_i}{\rho_i^2} - \frac{p_j}{\rho_j^2})\nabla W_{ij} \approx -m_i^2 \frac{2p^i}{\rho_0^2} \sum_j \nabla W_{ij}$$

$$\rho_0 - \rho_i^* = \Delta t \cdot m_i \cdot \sum_j \left(-m_i^2 \frac{2p_i}{\rho_0^2} \sum_j \nabla W_{ij} + m_i^2 \frac{2p_j}{\rho_0^2} \sum_k \nabla W_{jk}\right) \nabla W_{ij}$$

- for particle j, do not consider all contributions, but only the contribution from i

$$\mathbf{F}_{j|i}^p = -m_i (\frac{p_i}{\rho_i^2} - \frac{p_j}{\rho_j^2})\nabla W_{ij} \approx -m_i^2 \frac{2p^i}{\rho_0^2} \sum_j \nabla W_{ij}$$

UNI FREIBURG

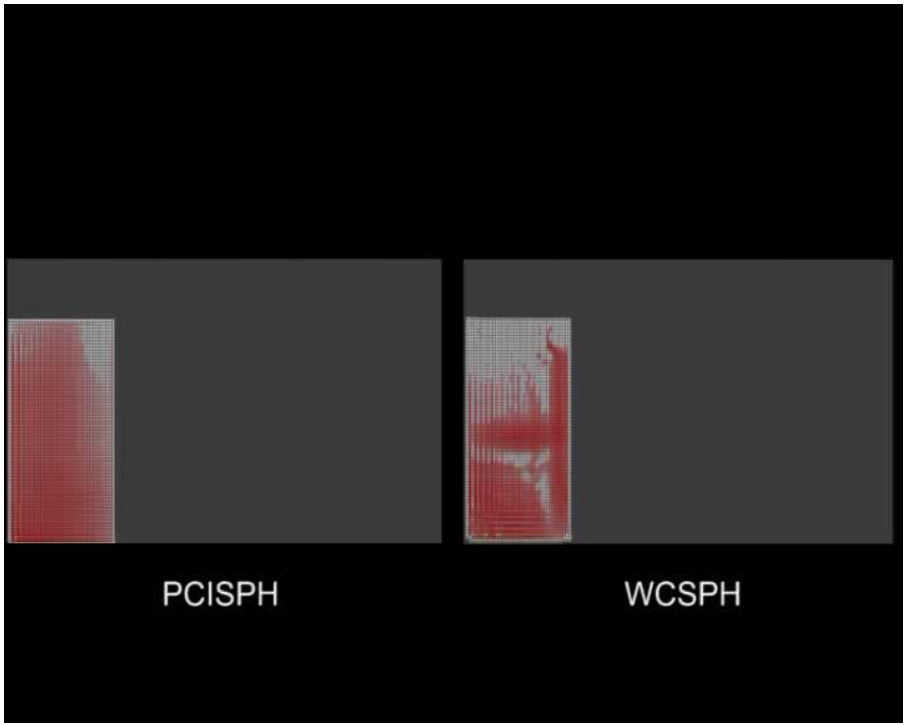
Iterative SESPH - Performance

- typically three to five iterations for density errors between 0.1% and 1%
- typical speed-up over non-iterative SESPH: 50
  - more computations per time step compared to SESPH
  - significantly larger time step than in SESPH
- EOS and stiffness constant influence the number of required iterations to get a desired density error
  - rarely analyzed
- non-linear relation between time step and iterations
  - largest possible time step does not necessarily lead to an optimal overall performance

UNI FREIBURG

Pressure Computation

- iterative SESPH (PCISPH)
  - [Solenthaler 2009]
  - iterative pressure computation
  - large time step
- non-iterative SESPH (WCSPH)
  - [Becker and Teschner 2007]
  - efficient to compute
  - small time step
- computation time for the PCISPH scenario is 20 times shorter than WCSPH



Projection Schemes

- compute pressure with a pressure Poisson equation
  $\Delta t \nabla^2 p_i = \rho_0 \nabla \cdot \mathbf{v}_i^* = \frac{1}{\Delta t}(\rho_0 - \rho_i^*)$
- $\mathbf{v}_i^*$ is the predicted velocity considering all non-pressure forces
- ρ_i^* is the corresponding predicted density, e.g. $\rho_i^* = \rho_i - \Delta t \cdot \rho_i \cdot \nabla \cdot \mathbf{v}_i^*$
- density invariance is preferred
- divergence-free schemes suffer from drift

UNI FREIBURG

Projection Schemes - Derivation

- $$\frac{d\mathbf{v}_i}{dt} = -\frac{1}{\rho_i}\nabla p_i + \nu\nabla^2\mathbf{v}_i + \frac{\mathbf{F}_i^{other}}{m_i}$$

$$\mathbf{v}_i(t+\Delta t) = \mathbf{v}_i(t) + \Delta t\left(-\frac{1}{\rho_i}\nabla p_i + \nu\nabla^2\mathbf{v}_i + \frac{\mathbf{F}_i^{other}}{m_i}\right)$$

$$\mathbf{v}_i^* = \mathbf{v}_i(t) + \Delta t\left(\nu\nabla^2\mathbf{v}_i + \frac{\mathbf{F}_i^{other}}{m_i}\right)$$

$$\mathbf{v}_i(t+\Delta t) = \mathbf{v}_i^* - \Delta t\frac{1}{\rho_i}\nabla p_i$$

$$\frac{\mathbf{v}_i(t+\Delta t)-\mathbf{v}_i^*}{\Delta t} = -\frac{1}{\rho_i}\nabla p_i$$

$$\nabla\cdot\frac{\mathbf{v}_i(t+\Delta t)-\mathbf{v}_i^*}{\Delta t} = -\nabla\cdot\frac{1}{\rho_i}\nabla p_i$$

UNI FREIBURG

Projection Schemes - Derivation

- $\nabla \cdot \frac{\mathbf{v}_i(t+\Delta t)-\mathbf{v}_i^*}{\Delta t} = -\nabla \cdot \frac{1}{\rho_i}\nabla p_i$

$$\nabla \cdot \mathbf{v}_i(t+\Delta t) - \nabla \cdot \mathbf{v}_i^* = -\nabla \cdot \Delta t \frac{1}{\rho_i}\nabla p_i$$

$$\nabla \cdot \mathbf{v}_i(t+\Delta t) = 0$$

divergence of the velocity at the next time step should be zero.

$$\nabla \cdot \mathbf{v}_i^* = \Delta t \frac{1}{\rho_i}\nabla^2 p_i$$

UNI FREIBURG

Projection Schemes

- linear system with unknown pressure values
- iterative solvers
  - Conjugate Gradient
  - relaxed Jacobi
- fast computation per iteration
  - 30-40 non-zero entries in each equation
  - very few information per particle
  - matrix-free implementations
- huge time steps
- convergence tends to be an issue
  - up to 100 iterations, dependent on the formulation

UNI FREIBURG

Implicit Incompressible SPH Derivation

- discretizing the continuity equation
  $\frac{\mathrm{D}\rho_i(t+\Delta t)}{\mathrm{D}t} = -\rho_i(t+\Delta t)\nabla \cdot \mathbf{v}_i(t+\Delta t)$ to
  $\frac{\rho_i(t+\Delta t)-\rho_i(t)}{\Delta t} = \sum_j m_j \mathbf{v}_{ij}(t+\Delta t)\nabla W_{ij}(t)$
  forward difference SPH
- unknown velocity $\mathbf{v}_i(t+\Delta t)$ can be rewritten using known non-pressure accel. $\mathbf{a}_i^{nonp}(t)$ and unknown pressure accel. $\mathbf{a}_i^p(t)$

$$\mathbf{v}_i(t+\Delta t) = \underbrace{\mathbf{v}_i(t) + \Delta t \mathbf{a}_i^{nonp}(t)}_{\mathbf{v}_i^*} + \Delta t \mathbf{a}_i^p(t)$$

predicted velocity only
using non-pressure forces

UNI FREIBURG

Implicit Incompressible SPH Derivation

- with $\rho_i(t+\Delta t) = \rho_0$ rest density is the desired density at t+Δt

  and $\rho_i^* = \rho_i(t) + \Delta t \sum_j m_j \mathbf{v}_{ij}^* \nabla W_{ij}(t)$

  predicted density, if only non-pressure forces are applied

  the discretized continuity equation can be written as

  $\frac{\rho_0 - \rho_i^*}{\Delta t} = \Delta t \sum_j m_j \left(\mathbf{a}_i^p(t) - \mathbf{a}_j^p(t)\right) \nabla W_{ij}(t)$

  with unknown pressure accel. $\mathbf{a}_i^p(t)$
- unknown pressure accel. $\mathbf{a}_i^p(t)$ can be rewritten using unknown pressures $p_i(t)$

  $\mathbf{a}_i^p(t) = -\sum_j m_j \left(\frac{p_i(t)}{\rho_i^2(t)} - \frac{p_j(t)}{\rho_j^2(t)}\right) \nabla W_{ij}(t)$

  resulting in a linear system with unknown pressures

UNI FREIBURG

Implicit Incompressible SPH
Linear System

- one equation per particle

  $\sum_j a_{ij} p_j = s_i = \frac{\rho_0 - \rho_i^*}{\Delta t}$

- iterative solver, e.g.,

$$p_i^{l+1} = (1-\omega) p_i^l + \omega \frac{s_i - \sum_{j \neq i} a_{ij} p_j^l}{a_{ii}}$$

  - relaxed Jacobi
  - matrix-free implementation
  - user-defined $\omega (= 0.5)$
  - p_i^l is the computed pressure in iteration l
  - p_i^0 is initialized, e.g., $p_i^0 = 0$ or $p_i^0 = p_i(t - \Delta t)$
  - system is not necessarily symmetric
    (a prerequisite for Conjugate Gradient)

UNI FREIBURG

Implicit Incompressible SPH Interpretation

- PPE $\Delta t^2 \nabla^2 p_i = \rho_0 - \rho_i^*$

  density change due to pressure accelerations

  predicted density error

- discretized PPE
  - $\mathbf{Ap} = \mathbf{s}$
  - $\underbrace{\Delta t^2 \sum_j m_j \left(\mathbf{a}_i^p - \mathbf{a}_j^p\right) \nabla W_{ij}}_{(\mathbf{Ap})_i} = \underbrace{\rho_0 - \rho_i^*}_{s_i} \qquad \mathbf{a}_i^p = -\sum_j m_j \left(\frac{p_i}{\rho_i^2} - \frac{p_j}{\rho_j^2}\right) \nabla W_{ij}$

  - $\Delta t \sum_j m_j \left(\Delta t \mathbf{a}_i^p - \Delta t \mathbf{a}_j^p\right) \nabla W_{ij} = \rho_0 - \rho_i^*$
    $\Delta t \sum_j m_j \left(\mathbf{v}_i^p - \mathbf{v}_j^p\right) \nabla W_{ij} = \rho_0 - \rho_i^*$
    $\Delta t \cdot \rho_i \cdot \nabla \cdot \mathbf{v}_i^p = \rho_0 - \rho_i^*$

    pressure accel. causes a velocity change $\mathbf{v}^p$ whose divergence causes a density change

UNI FREIBURG

Implicit Incompressible SPH Boundary Handling

- PPE

$$\Delta t^2 \nabla^2 p_f = \rho_0 - \rho_f^* = \rho_0 - \rho_f + \Delta t \rho_0 \nabla \cdot \mathbf{v}_f^*$$

- discretized PPE including boundary handling $\mathbf{Ap} = \mathbf{s}$

$$\mathbf{a}_f^p = -\sum_{f_f} m_{f_f} \left(\frac{p_f}{\rho_f^2} + \frac{p_{f_f}}{\rho_{f_f}^2} \right) \nabla W_{f f_f} - \gamma \sum_{f_b} m_{f_b} 2 \frac{p_f}{\rho_f^2} \nabla W_{f f_b}$$

$$\underbrace{\Delta t^2 \sum_{f_f} m_{f_f} \left(\mathbf{a}_f^p - \mathbf{a}_{f_f}^p \right) \nabla W_{f f_f} + \Delta t^2 \sum_{f_b} m_{f_b} \mathbf{a}_f^p \nabla W_{f f_b}}_{(\mathbf{Ap})_f} =$$

$$\underbrace{\rho_0 - \rho_f - \Delta t \sum_{f_f} m_{f_f} \left(\mathbf{v}_f^* - \mathbf{v}_{f_f}^* \right) \nabla W_{f f_f} - \Delta t \sum_{f_b} m_{f_b} \left(\mathbf{v}_f^* - \mathbf{v}_{f_b}(t + \Delta t) \right) \nabla W_{f f_b}}_{\mathbf{s}_f}$$

Implicit Incompressible SPH Implementation with Boundary Handling

- initialization
  - density $\rho_f = \sum_{f_f} m_{f_f} W_{ff_f} + \sum_{f_b} m_{f_b} W_{ff_b}$
  - predicted velocity $\mathbf{v}_f^* = \mathbf{v}_f + \Delta t \mathbf{a}_f^{nonp}$
  - source term $s_f = \rho_0 - \rho_f - \Delta t \sum_{f_f} m_{f_f} \left(\mathbf{v}_f^* - \mathbf{v}_{f_f}^* \right) \nabla W_{ff_f} - \Delta t \sum_{f_b} m_{f_b} \dots$
  - pressure $p_f^0 = 0$
  - diagonal element of matrix $\mathbf{A}$

$$
\begin{aligned}
a_{ff} =& \Delta t^2 \sum_{f_f} m_{f_f} \left(- \sum_{f_f} \frac{m_{f_f}}{\rho_0^2} \nabla W_{ff_f} - 2\gamma \sum_{f_b} \frac{m_{f_b}}{\rho_0^2} \nabla W_{ff_b} \right) \nabla W_{ff_f} \\
&+ \Delta t^2 \sum_{f_f} m_{f_f} \left(\frac{m_f}{\rho_0^2} \nabla W_{f_f f} \right) \nabla W_{ff_f} \\
&+ \Delta t^2 \sum_{f_b} m_{f_b} \left(- \sum_{f_f} \frac{m_{f_f}}{\rho_0^2} \nabla W_{ff_f} - 2\gamma \sum_{f_b} \frac{m_{f_b}}{\rho_0^2} \nabla W_{ff_b} \right) \nabla W_{ff_b}
\end{aligned}
$$

Implicit Incompressible SPH Implementation with Boundary Handling

- iteration l
  - first particle loop
    - predicted pressure acceleration

      $(\mathbf{a}_f^p)^l = -\sum_{f_f} m_{f_f} \left(\frac{p_f^l}{\rho_f^2} + \frac{p_{f_f}^l}{\rho_{f_f}^2} \right) \nabla W_{f f_f} - \gamma \sum_{f_b} m_{f_b} 2 \frac{p_f^l}{\rho_f^2} \nabla W_{f f_b}$
  - second particle loop
    - predicted density change due to pressure acceleration

      $(\mathbf{Ap})_f^l = \Delta t^2 \sum_{f_f} m_{f_f} \left((\mathbf{a}_f^p)^l - (\mathbf{a}_{f_f}^p)^l \right) \nabla W_{f f_f} + \Delta t^2 \sum_{f_b} m_{f_b} (\mathbf{a}_f^p)^l \nabla W_{f f_b}$
    - pressure update

      $p_f^{l+1} = \max \left(p_f^l + \omega \frac{s_f - (\mathbf{Ap})_f^l}{a_{f f}}, 0 \right)$
    - predicted density deviation per particle

      $(\rho_f^{\text{error}})^l = (\mathbf{Ap})_f^l - s_f$

UNI FREIBURG

Comparison with Iterative SESPH

- breaking dam
  - 100k particles
  - 0.01% average density error
  - particle radius 0.025m

	PCISPH			IISPH			PCISPH / IISPH		
		total comp. time [s]			total comp. time [s]		ratio		
Δt [s]	avg. iter.	pressure	overall	avg. iter.	pressure	overall	iterations	pressure	overall
0.0005	4.3	540	1195	2.2	148	978	2.0	3.6	1.2
0.00067	7.2	647	**1145**	2.9	149	753	2.5	4.3	1.5
0.001	14.9	856	1187	4.9	164	576	3.0	5.2	2.1
0.0025	66.5	1495	1540	18.4	242	410	3.6	**6.2**	3.8
0.004	-	-	-	33.5	273	**379**	-	-	-
0.005	-	-	-	45.8	297	383	-	-	-

- largest possible time step does not necessarily result in the best performance

References

- state equation SPH (SESPH)
  - BECKER M., TESCHNER M.: Weakly Compressible SPH for Free Surface Flows. *ACM SIGGRAPH/Eurographics SCA*, 2007.
- iterative SESPH
  - SOLENTHALER B., PAJAROLA R.: Predictive-corrective Incompressible SPH. *ACM TOG (SIGGRAPH 2009),* 2009.
  - HE X., LIU N., Li S., WANG H., WANG G.: Local Poisson SPH for Viscous Incompressible Fluids, *Computer Graphics Forum*, 2012.
  - MACKLIN M., MUELLER M., Position-based Fluids, *ACM TOG (SIGGRAPH 2013)*, 2013.
- incompressible SPH
  - IHMSEN M., CORNELIS J., SOLENTHALER B., HORVATH C., TESCHNER M.: Implicit Incompressible SPH. *IEEE Transactions on Visualization and Computer Graphics TVCG*, 2013.

UNI FREIBURG

Outline

- concept of an SPH fluid simulator
- momentum equation
- SPH basics
- neighborhood search
- boundary handling
- incompressibility
- surface reconstruction

UNI FREIBURG