Simulation in Computer Graphics

Particle Fluids

Matthias Teschner
Particle Fluids in Animation

Cooperation with Pixar Animation Studios

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

Copyright NHB Studios, Berlin, Hamburg, Dusseldorf
Particle Fluids in Engineering

PreonLab
FIFTY2 Technology
FORD F-150
Water wading
Particle Fluids in Engineering

PreonLab
FIFTY2 Technology

AVL

Lubrication
Validation
Outline

- Concept of an SPH fluid simulator
- Momentum equation
- SPH basics
- Neighborhood search
- Boundary handling
- Incompressibility
Concept
Concept
Fluid Representation
Fluid Representation

- Fluid body is subdivided into small moving parcels, i.e. particles, with fluid properties
**Particles / Fluid Parcels**

- Represent small fluid portions
- Are represented by a sample position $x_i$
- Move with their velocity $v_i$
- 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 $\rho_0$
- Define number $n$ of particles $V_i = \frac{V}{n}$
- Particles of uniform size
- Compute particle mass as $m_i = \rho_0 \cdot V_i$
- Sample $x_i$ 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.
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 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
Simulation Step - Example

- Gravity and viscosity would change the parcel volume
  \[ \dot{x}(t) = 0, \quad \dot{v}(t) = 0 \]
  \[ \text{Gravity} \quad \text{Viscosity} \]
  \[ \nu \nabla^2 v(t) = 0 \]

- Pressure acceleration avoids the volume/density change
  \[ -\frac{1}{\rho} \nabla p = -g \]
  \[ \text{Pressure acceleration} \]
Simulation Step - Example

- Current state
  \[ \begin{align*}
  x(t) &= 0 \\
  v(t) &= 0
  \end{align*} \]

- Overall acceleration
  \[ a(t) = g + \nu \nabla^2 v(t) - \frac{1}{\rho} \nabla p \]
  \[ = g + 0 - g = 0 \]

- Subsequent state
  \[ \begin{align*}
  x(t + \Delta t) &= x(t) + \Delta t \cdot v(t) = 0 \\
  v(t + \Delta t) &= v(t) + \Delta t \cdot a(t) = 0
  \end{align*} \]
**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 SPH
  - Gingold and Monaghan, Lucy
Simulation Step - Implementation

- Determine adjacent particles / neighbors \( x_j(t) \) of particle \( x_i(t) \) (\( x_i(t) \) is neighbor of \( x_i(t) \))
- Compute accelerations \( a_i(t) = \sum_j \ldots \) as sums of neighbors
- Advect the particles, e.g. Euler-Cromer
- Determine neighbors of particle \( x_i(t + \Delta t) \)
- ...
Governing Equations

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

\[
\frac{d\mathbf{x}_i}{dt} = \mathbf{v}_i
\]

- Time rate of change of the velocity \( \mathbf{v}_i \) of an advected sample is governed by the Lagrange form of the Navier-Stokes equation

\[
\frac{d\mathbf{v}_i}{dt} = -\frac{1}{\rho_i} \nabla p_i + \nu \nabla^2 \mathbf{v}_i + \frac{\mathbf{F}_{\text{other}}}{m_i}
\]
Accelerations

\[ -\frac{1}{\rho_i} \nabla p_i : \text{acceleration due to pressure differences} \]

- Preserves the fluid volume / density
- Acts in normal direction at the surface of the fluid element
- Small and preferably constant density deviations are important for high-quality simulation
Accelerations

- $\nu \nabla^2 \mathbf{v}_i$: acceleration due to friction forces between particles with different velocities
  - Friction forces act in tangential (and normal) direction at fluid elements
  - Kinematic viscosity $\nu \approx 10^{-6} \text{m}^2 \cdot \text{s}^{-1}$: larger friction is less realistic, but can improve the stability
  - Dynamic viscosity $\eta = \mu = \nu \cdot \rho_0$
- $\frac{\mathbf{F}_{\text{other}}}{m_i}$: e.g., gravity
Accelerations

\[-\frac{1}{\rho} \nabla p = -\frac{1}{\rho} \left( \begin{array}{c} \frac{\partial p}{\partial x_x} \\ \frac{\partial p}{\partial x_y} \\ \frac{\partial p}{\partial x_z} \end{array} \right) = -\frac{1}{\rho} \nabla \cdot \left( \begin{array}{ccc} p & 0 & 0 \\ 0 & p & 0 \\ 0 & 0 & p \end{array} \right)\]

\[\nu \nabla^2 \mathbf{v} = \nu \nabla \cdot (\nabla \mathbf{v}) = \nu \nabla \cdot \left( \begin{array}{ccc} \frac{\partial v_x}{\partial x_x} & \frac{\partial v_x}{\partial x_y} & \frac{\partial v_x}{\partial x_z} \\ \frac{\partial v_y}{\partial x_x} & \frac{\partial v_y}{\partial x_y} & \frac{\partial v_y}{\partial x_z} \\ \frac{\partial v_z}{\partial x_x} & \frac{\partial v_z}{\partial x_y} & \frac{\partial v_z}{\partial x_z} \end{array} \right)\]

\[= \nu \left( \begin{array}{ccc} \frac{\partial^2 v_x}{\partial x_x^2} + \frac{\partial^2 v_x}{\partial x_y^2} + \frac{\partial^2 v_x}{\partial x_z^2} \\ \frac{\partial^2 v_y}{\partial x_x^2} + \frac{\partial^2 v_y}{\partial x_y^2} + \frac{\partial^2 v_y}{\partial x_z^2} \\ \frac{\partial^2 v_z}{\partial x_x^2} + \frac{\partial^2 v_z}{\partial x_y^2} + \frac{\partial^2 v_z}{\partial x_z^2} \end{array} \right)\]
Forces

- 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

\[
\frac{dv}{dt} = g + \nu \nabla^2 v - \frac{1}{\rho} \nabla p
\]

**Lagrangian:** Acceleration of a moving parcel.

\[
\frac{\partial v}{\partial t} = g + \nu \nabla^2 v - \frac{1}{\rho} \nabla p - (v \cdot \nabla)v
\]

**Eulerian:** Acceleration at a static cell.

\[
\frac{Dv}{Dt} = g + \nu \nabla^2 v - \frac{1}{\rho} \nabla p
\]

\[
\frac{Dv}{Dt} = \frac{\partial v}{\partial t} + (v \cdot \nabla)v \quad \text{or}
\]

\[
\frac{Dv}{Dt} = \frac{dv}{dt} \quad \frac{dx}{dt} = v
\]
Smoothed Particle Hydrodynamics

– Proposed by Gingold / Monagahan and Lucy (1977)
– SPH interpolates quantities at arbitrary positions and approximates the spatial derivatives with a finite number of samples, i.e. adjacent particles
SPH for Fluids

- SPH in a Lagrangian fluid simulation
  - Fluid is represented with particles
  - Particle positions and velocities are governed by
    \[
    \frac{dx_i}{dt} = v_i \quad \text{and} \quad \frac{dv_i}{dt} = -\frac{1}{\rho_i} \nabla p_i + \nu \nabla^2 v_i + \frac{F_{\text{other}}}{m_i}
    \]
  - \(\rho_i\), \(-1/\rho_i \nabla p_i\), \(\nu \nabla^2 v_i\) and \(F_{\text{other}}/m_i\) are computed with SPH
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 $\mathbf{x}_i$
  - $W_{ij} = W\left(\frac{\|\mathbf{x}_i - \mathbf{x}_j\|}{h}\right) = W(q)$
  - $h$ is typically the particle size
  - $W(q) > 0$ for, e.g. $0 \leq q < 2$
Kernel Function

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

$$W(q) = \alpha \begin{cases} 
(2 - q)^3 - 4(1 - q)^3 & 0 \leq q < 1 \\
(2 - q)^3 & 1 \leq q < 2 \\
0 & q \geq 2
\end{cases} \quad q = \frac{\|x_i - x_j\|}{h}$$

- Number of considered neighbors depends on
  - Dimensionality, kernel support, particle spacing
  - E.g., 3D, cubic spline, support $2h$, particle spacing $h$
    results in 30-40 neighboring particles
  - Number of neighbors influences performance / accuracy
Kernel Function in 1D

\[ W(x_j - x_i) = \begin{cases} 
\frac{1}{6h} \left( (2 - \frac{||x_i - x_j||}{h})^3 - 4(1 - \frac{||x_i - x_j||}{h})^3 \right) & 0 \leq \frac{||x_i - x_j||}{h} < 1 \\
(2 - \frac{||x_i - x_j||}{h})^3 & 1 \leq \frac{||x_i - x_j||}{h} < 2 \\
0 & \frac{||x_i - x_j||}{h} \geq 2
\end{cases} \]

\[ W(x_j - x_i) = \frac{2}{3h} e^{-\frac{||x_i - x_j||^2}{2(0.59h)^2}} \]
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

\[ \nabla A_i = \rho_i \sum_j m_j \left( \frac{A_i}{\rho_i^2} + \frac{A_j}{\rho_j^2} \right) \nabla W_{ij} \]
\[ \nabla^2 A_i = 2 \sum_j \frac{m_j}{\rho_j} A_{ij} \frac{x_{ij} \cdot \nabla W_{ij}}{x_{ij} \cdot x_{ij} + 0.01h^2} \]
\[ \nabla \cdot A_i = -\frac{1}{\rho_i} \sum_j m_j A_{ij} \nabla W_{ij} \]

\[ A_{ij} = A_i - A_j \quad A_{ij} = A_i - A_j \quad x_{ij} = x_i - x_j \]
Kernel Derivative in 1D

\[ W(x_j - x_i) = \frac{1}{6h} \left\{ \begin{array}{ll}
(2 - \frac{||x_i-x_j||}{h})^3 & 0 \leq \frac{||x_i-x_j||}{h} < 1 \\
(2 - \frac{||x_i-x_j||}{h})^3 & 1 \leq \frac{||x_i-x_j||}{h} < 2 \\
0 & \frac{||x_i-x_j||}{h} \geq 2
\end{array} \right. \]

\[ \nabla W(x_j - x_i) = \frac{-(x_i-x_j)}{6h^2||x_i-x_j||} \left\{ \begin{array}{ll}
-3(2 - \frac{||x_i-x_j||}{h})^2 + 12(1 - \frac{||x_i-x_j||}{h})^2 & 0 \leq \frac{||x_i-x_j||}{h} < 1 \\
-3(2 - \frac{||x_i-x_j||}{h})^2 & 1 \leq \frac{||x_i-x_j||}{h} < 2 \\
0 & \frac{||x_i-x_j||}{h} \geq 2
\end{array} \right. \]
Density

- 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

- Differential update
  - Using the continuity equation
  - Time rate of change of the density is related to the divergence of the velocity field $\frac{d\rho_i}{dt} = -\rho_i \nabla \cdot \mathbf{v}_i$
    $\frac{d\rho_i}{dt} = \sum_j m_j \mathbf{v}_{ij} \nabla W_{ij}$
  - Drift
Pressure

- Quantifies fluid compression
  - E.g., state equation \( p_i = \max \left( k \left( \frac{\rho_i}{\rho_0} - 1 \right), 0 \right) \)
  - Rest density of the fluid \( \rho_0 \)
  - User-defined stiffness \( k \)
- Pressure acceleration
  - \( \mathbf{a}_i^p = - \frac{1}{\rho_i} \nabla p_i = - \sum_j m_j \left( \frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) \nabla W_{ij} \)
  - Accelerates particles from high to low pressure, i.e. from high to low compression to minimize density deviation \( \frac{\rho_i}{\rho_0} - 1 \)
Simple SPH Fluid Solver

- Find neighbors of all particles
- Compute density
- Compute pressure
- Compute non-pressure accelerations, e.g. viscosity, gravity
- Compute pressure acceleration
- Update velocity and position

Contact handling, i.e. boundary handling is often realized as pressure acceleration.
SPH Discretizations

– Density computation \( \rho_i = \sum_j m_j W_{ij} \)
– Pressure acceleration \( -\frac{1}{\rho_i} \nabla p_i = -\sum_j m_j \left( \frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) \nabla W_{ij} \)
– Viscosity acceleration \( \nu \nabla^2 \mathbf{v}_i = 2\nu \sum_j m_j \frac{\mathbf{v}_{ij} \cdot \mathbf{x}_{ij}}{\mathbf{x}_{ij} \cdot \mathbf{x}_{ij} + 0.01h^2} \nabla W_{ij} \)
**Simple SPH Fluid Solver**

- **for all particle** \( i \) **do**
  - find neighbors \( j \)

**for all particle** \( i \) **do**

\[
\begin{align*}
\rho_i &= \sum_j m_j W_{ij} & \text{Compute density} \\
p_i &= k \left( \frac{\rho_i}{\rho_0} - 1 \right) & \text{Compute pressure}
\end{align*}
\]

**for all particle** \( i \) **do**

\[
\begin{align*}
\mathbf{a}_{i}^{\text{nonp}} &= \nu \nabla^2 \mathbf{v}_i + \mathbf{g} & \text{Compute non-pressure accelerations} \\
\mathbf{a}_{i}^{p} &= -\frac{1}{\rho_i} \nabla p_i & \text{Compute pressure acceleration}
\end{align*}
\]

**for all particle** \( i \) **do**

\[
\begin{align*}
\mathbf{v}_i(t + \Delta t) &= \mathbf{v}_i(t) + \Delta t \mathbf{a}_i(t) \\
\mathbf{x}_i(t + \Delta t) &= \mathbf{x}_i(t) + \Delta t \mathbf{v}_i(t + \Delta t)
\end{align*}
\]
Boundary Handling

– Boundaries can be represented with static fluid samples
– Computations incorporate boundary samples, e.g.

\[
\rho_i = \sum_f m_f W_{if} + \sum_b m_b W_{ib}
\]

\[
-\frac{1}{\rho_i} \nabla p_i = -\sum_f m_f \left( \frac{p_i}{\rho_i^2} + \frac{p_f}{\rho_f^2} \right) \nabla W_{if} - \sum_b m_b \left( \frac{p_i}{\rho_i^2} + \frac{p_b}{\rho_b^2} \right) \nabla W_{ib}
\]

– Fluid sample at boundary
  – Density and pressure increases
  – Pressure acceleration resolves contact
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 $h$
  – Spacing governs particle mass
  – Ratio of support vs. spacing governs the number of neighbors
– Numerical integration scheme
  – Semi-implicit Euler (symplectic Euler or Euler-Cromer) is commonly used
Setting

- Time step
  - Size is governed by the Courant-Friedrich-Levy (CFL) condition
  - E.g., $\Delta t \leq \lambda \frac{h}{\|v_{\text{max}}\|}$ with $\lambda = 0.1$ and particle spacing $h$
  - Motivation: For $\lambda \leq 1$, a particle moves less than its size / diameter per time step
Outline

– Concept of an SPH fluid simulator
– Momentum equation
– SPH basics
– Neighborhood search
– Boundary handling
– Incompressibility
**Force Types**

- Momentum equation
  \[
  \frac{d\mathbf{v}_i}{dt} = -\frac{1}{\rho_i} \nabla p_i + \nu \nabla^2 \mathbf{v}_i + \frac{\mathbf{F}_{\text{other}}}{m_i}
  \]

- Body forces

- Surface forces
  - 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

\[
\left( p - \left( p + \frac{\partial p}{\partial x} \, dx \right) \right) \, dy \, dz = - \frac{\partial p}{\partial x} \, dx \, dy \, dz = - \frac{\partial p}{\partial x} \, V
\]
Overall Pressure Force

– Pressure force at particle $i$

$$\mathbf{F}_{i}^{p} = - \begin{pmatrix} \frac{\partial p_i}{\partial x_{i,x}} \\
\frac{\partial p_i}{\partial x_{i,y}} \\
\frac{\partial p_i}{\partial x_{i,z}} \end{pmatrix} \quad \mathbf{V}_i = -\nabla p_i \quad \mathbf{V}_i = -\frac{m_i}{\rho_i} \nabla p_i$$

– Pressure acceleration

$$\mathbf{a}_{i}^{p} = \frac{\mathbf{F}_{i}^{p}}{m_i} = -\frac{1}{\rho_i} \nabla p_i$$
Cauchy Momentum Equation

- Lagrange form \( \frac{dv_i}{dt} = \frac{1}{\rho} \nabla \cdot \mathbf{\sigma} + \frac{F_{\text{other}}}{m} \)
- \( \mathbf{\sigma} \) is the stress tensor (a 3x3 matrix in 3D) describing the pressure distribution at the surface of a fluid element \( \mathbf{\sigma} = -\rho \mathbf{I}_3 + \mathbf{\tau} \)
- \( \nabla \cdot \mathbf{\sigma} \) is the resulting force per volume
- \( \mathbf{\tau} \) is the viscous stress tensor
- \( \nabla \cdot \mathbf{\tau} = \nu \nabla^2 \mathbf{v} \) is the resulting viscosity force per volume
- \( \frac{dv_i}{dt} = -\frac{1}{\rho_i} \nabla p_i + \nu \nabla^2 \mathbf{v}_i + \frac{F_{\text{other}}}{m_i} \)
Outline

- Concept of an SPH fluid simulator
- Momentum equation
- SPH basics
- Neighborhood search
- Boundary handling
- Incompressibility
Illustration

– Approximation of a function and its derivatives from discrete samples, e.g. $\rho, \nabla p, \nabla^2 v$

– Convolution of discrete samples with reconstruction filter

$\rho, p, \ldots$

Reconstructed function

Function samples (particle data)

Reconstruction kernel (reversed SPH kernel)

Reconstruction kernel for the first derivative (reversed derivative of the SPH kernel)
Derivation

- Quantity $A$ at position $x$ can be written as
  $$A(x) = \int_\Omega A(x')\delta(x - x')dx'$$
  Convolution of $A$ and $\delta(x' - x)$

- Dirac delta $\delta(x) = \delta(x)\delta(y)\delta(z)$ and $\delta(x) = \begin{cases} \infty & x = 0 \\ 0 & x \neq 0 \end{cases}$

- $\int_{-\infty}^{+\infty} \delta(x)dx = 1$

- Dirac delta is approximated with a kernel function with limited local support, e.g. $2h$
  $$A(x) \approx \int_{\Omega_h} A(x')W(x - x', 2h)dx'$$
  Convolution of $A$ and $W(x' - x)$

  Particle size $h$
Kernel Function

- Integral should be normalized (unity condition) \( \int_{\Omega} W(x' - x, 2h)dx' = 1 \)
- Support should be compact \( W(x' - x, 2h) = 0 \) for \( ||x - x'|| > 2h \)
- Should be symmetric \( W(x' - x, 2h) = W(x' - x, 2h) \)
- Should be non-negative \( W(x' - x, 2h) \geq 0 \)
- Should converge to the Dirac delta for \( 2h \rightarrow 0 \)
- Should be differentiable
Particle Approximation

- \[ A(x) \approx \int_{\Omega_h} A(x') W(x - x', 2h) \, dx' \]
  \[ = \int_{\Omega_h} \frac{A(x')}{\rho(x')} W(x - x', 2h) \rho(x') \, dx' \]

- Consider a limited number of samples / particles \( x_j \) representing a mass \( m(x_j) = \rho(x_j) V(x_j) \)
  \[ A(x_i) \approx \sum_j A(x_j) W(x_i - x_j, 2h) V(x_j) \]
  \[ A(x_i) \approx \sum_j \frac{A(x_j)}{\rho(x_j)} W(x_i - x_j, 2h) m(x_j) \]

- Typical notation
  \[ A_i = \sum_j \frac{m_j}{\rho_j} A_j W_{ij} \]
Kernel Function

- Close to a Gaussian
  - Compact support between $2h$ and $5h$
- E.g., cubic spline

\[ W(q) = \alpha \begin{cases} 
(2 - q)^3 - 4(1 - q)^3 & 0 \leq q < 1 \\
(2 - q)^3 & 1 \leq q < 2 \\
0 & q \geq 2 
\end{cases} \quad q = \frac{||x_i - x_j||}{h} \]

with \( \alpha = \frac{1}{6h} \) (1D), \( \alpha = \frac{5}{14\pi h^2} \) (2D), \( \alpha = \frac{1}{4\pi h^3} \) (3D)

- Number of considered samples depends on
  - Dimensionality, kernel support, particle spacing
  - Number of neighbors should not be too small
Kernel Function

\[ W(x_j - x_i) = \alpha \begin{cases} 
(2 - \frac{\|x_i - x_j\|}{h})^3 & 0 \leq \frac{\|x_i - x_j\|}{h} < 1 \\
(2 - \frac{\|x_i - x_j\|}{h})^3 & 1 \leq \frac{\|x_i - x_j\|}{h} < 2 \\
0 & \|x_i - x_j\| \geq 2 
\end{cases} \]

\[ W(x_j - x_i) = W(x_i - x_j) \quad \text{depends on the distance between samples} \]
Kernel Function - Implementation

- Reversed kernel function as used in SPH sums

\[ W(x_i - x_j) = \alpha \begin{cases} 
(2 - \frac{\|x_i - x_j\|}{h})^3 & 0 \leq \frac{\|x_i - x_j\|}{h} < 1 \\
(2 - \frac{\|x_i - x_j\|}{h})^3 & 1 \leq \frac{\|x_i - x_j\|}{h} < 2 \\
0 & \frac{\|x_i - x_j\|}{h} \geq 2 
\end{cases} \]

- Implementation

\[
\begin{align*}
d &:= \text{distance}(x_i, x_j)/h; \\
t1 &:= \max(1-d, 0); \\
t2 &:= \max(2-d, 0); \\
w &:= \alpha \ast (t2 \ast t2 \ast t2 - 4 \ast t1 \ast t1 \ast t1); 
\end{align*}
\]
First Kernel Derivative

- $\nabla W_{ij} = \left( \frac{\partial W_{ij}}{\partial x_{j,x}}, \frac{\partial W_{ij}}{\partial x_{j,y}}, \frac{\partial W_{ij}}{\partial x_{j,z}} \right)^T$ \quad $\nabla W_{ij} = \frac{\partial W(q)}{\partial q} \nabla q$

- E.g., cubic spline

$$q = \frac{\|x_i - x_j\|}{h} \quad \nabla q = \frac{-(x_i - x_j)}{\|x_i - x_j\| h}$$

Derivative of $q$ with respect to $x_j$

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

$$\nabla W_{ij} = \alpha \frac{-(x_i - x_j)}{\|x_i - x_j\| h} \left\{ \begin{array}{ll}
-3(2 - q)^2 + 12(1 - q)^2 & 0 \leq q < 1 \\
-3(2 - q)^2 & 1 \leq q < 2 \\
0 & q \geq 2
\end{array} \right.$$
Kernel Derivative

\[ \nabla W(x_j - x_i) = \alpha \frac{(x_i - x_j)}{\|x_i - x_j\|} \begin{cases} 
-3\left(2 - \frac{\|x_i - x_j\|}{h}\right)^2 + 12\left(1 - \frac{\|x_i - x_j\|}{h}\right)^2 & 0 \leq \frac{\|x_i - x_j\|}{h} < 1 \\
-3\left(2 - \frac{\|x_i - x_j\|}{h}\right)^2 & 1 \leq \frac{\|x_i - x_j\|}{h} < 2 \\
0 & \|x_i - x_j\| \geq 2
\end{cases} \]

\[ \nabla W(x_j - x_i) = -\nabla W(x_i - x_j) \]

\[ x_j - x_i \]
SPH computes a convolution of $A$ and $\nabla W$ to approximate $\nabla A$. Therefore, the reversed kernel derivative $\nabla W(x_i - x_j)$ is used:

$$\nabla A(x_i) = \sum_j A(x_j) \nabla W(x_i - x_j) \frac{m(x_i)}{\rho(x_j)}$$

SPH notation:

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

$$\nabla W(x_i - x_j) = \alpha \frac{x_i - x_j}{\|x_i - x_j\|} \cdots$$

$$= -\nabla W(x_j - x_i) = \nabla W_{ij}$$

$\nabla W$ is anti-symmetric.
Kernel Derivative - Implementation

- Reversed kernel derivative as used in SPH sums

\[ \nabla W(x_i - x_j) = \alpha \frac{x_i - x_j}{\|x_i - x_j\|} \begin{cases} 
-3(2 - \frac{\|x_i - x_j\|}{h})^2 + 12(1 - \frac{\|x_i - x_j\|}{h})^2 & 0 \leq \frac{\|x_i - x_j\|}{h} < 1 \\
-3(2 - \frac{\|x_i - x_j\|}{h})^2 & 1 < \frac{\|x_i - x_j\|}{h} < 2 \\
0 & \frac{\|x_i - x_j\|}{h} \geq 2 
\end{cases} \]

- Implementation

\[
d := \text{distance}(x_i, x_j)/h;
\]
\[
t1 := \max(1-d, 0);
\]
\[
t2 := \max(2-d, 0);
\]
\[
w1 := \alpha \ast (x_i - x_j)/(d \ast h) \ast (-3 \ast t2 \ast t2 + 12 \ast t1 \ast t1);
\]
Second Kernel Derivative

\[ \nabla^2 W_{ij} = \nabla \cdot (\nabla W_{ij}) = \frac{\partial^2 W_{ij}}{\partial x^2_{j,x}} + \frac{\partial^2 W_{ij}}{\partial x^2_{j,y}} + \frac{\partial W_{ij}^2}{\partial x^2_{j,z}} \]

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

- E.g., cubic spline

\[ (\nabla q)^2 = \frac{-x_{ij}}{\|x_{ij}\| h} \cdot \frac{-x_{ij}}{\|x_{ij}\| h} = \frac{\|x_{ij}\|^2}{\|x_{ij}\|^2 h^2} = \frac{1}{h^2} \]

\[ \nabla \cdot (\nabla q) = \frac{d-1}{h \|x_{ij}\|} \quad d \text{ is the dimensionality} \]

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

\[ \frac{\partial^2 W(q)}{\partial q^2} = \alpha \begin{cases} 
6(2 - q) - 24(1 - q) & 0 \leq q < 1 \\
6(2 - q) & 1 \leq q < 2 \\
0 & q \geq 2 
\end{cases} \]
Design of a Kernel Function 1D

- Shape close to a Gaussian, e.g.

\[
\alpha \tilde{W}(\|x_i - x_j\|) = \alpha \tilde{W}\left(\frac{x}{h}\right) = \alpha \tilde{W}(q) = W(q) = \alpha \left\{
\begin{array}{cl}
(2 - q)^3 - 4(1 - q)^3 & 0 \leq q < 1 \\
(2 - q)^3 & 1 \leq q < 2 \\
0 & q \geq 2
\end{array}
\right.
\]

\[
2 \int_0^{2h} \alpha \tilde{W}(x) \, dx = 2 \int_0^{2} \alpha \tilde{W}(q) \, hdq = 1 \quad \text{Integration by substitution}
\]

\[
\alpha = \frac{1}{2 \int_0^{2} \tilde{W}(q) \, hdq}
\]

- 1D: integration over a line segment

\[
2 \int_0^{1} [(2 - q)^3 - 4(1 - q)^3] \, hdq + 2 \int_1^{2} (2 - q)^3 \, hdq = 2 \frac{11}{4} h + 2 \frac{1}{4} h
\]

\[
\alpha = \frac{1}{6h}
\]
Design of a Kernel Function

– 2D: Integration over the area of a circle
\[ \int_0^{2\pi} \int_0^h \tilde{W}(x)x \, dx \, d\phi = \int_0^{2\pi} \int_0^2 \tilde{W}(q)h\, dq \, d\phi = 2\pi \int_0^1 \left[q(2 - q)^3 - 4q(1 - q)^3\right] h^2 \, dq + 2\pi \int_1^2 q(2 - q)^3 h^2 \, dq = 2\pi \frac{11}{10} h^2 + 2\pi \frac{3}{10} h^2 \]
\[ \alpha = \frac{5}{14\pi h^2} \]

– 3D: Integration over the volume of a sphere
\[ \int_0^{2\pi} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \int_0^h \tilde{W}(x)x^2 \sin\theta \, dx \, d\theta \, d\phi = \int_0^{2\pi} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \int_0^2 \tilde{W}(q)(qh)^2 h \sin\theta \, dq \, d\theta \, d\phi = 4\pi \int_0^1 \left[q^2(2 - q)^3 - 4q(1 - q)^3\right] h^3 \, dq + 4\pi \int_1^2 q^2(2 - q)^3 h^3 \, dq = 4\pi \frac{19}{30} h^3 + 4\pi \frac{11}{30} h^3 \]
\[ \alpha = \frac{1}{4\pi h^3} \]
Spatial Derivatives

\[ \nabla_x A(x) \approx \int_{\Omega_h} [\nabla_{x'} A(x')] W(x - x', 2h) dx' \]

\[ \nabla_{x'} [A(x') W(x' - x, 2h)] = [\nabla_{x'} A(x')] W(x' - x, 2h) + A(x') \nabla_{x'} W(x' - x, 2h) \]

W is symmetric

\[ \nabla_{x'} [A(x') W(x' - x, 2h)] = [\nabla_{x'} A(x')] W(x - x', 2h) + A(x') \nabla_{x'} W(x' - x, 2h) \]

\[ [\nabla_{x'} A(x')] W(x - x', 2h) = \nabla_{x'} [A(x') W(x' - x, 2h)] - A(x') \nabla_{x'} W(x' - x, 2h) \]

\[ \int_{\Omega_h} \nabla_{x'} [A(x') W(x' - x, 2h)] dx' = \int_S A(x') W(x' - x, 2h) dS \]

\[ \int_S A(x') W(x' - x, 2h) dS = 0 \quad W = 0 \text{ on the surface } S \]

\[ \nabla_x A(x) \approx -\int_{\Omega_h} A(x') \nabla_{x'} W(x' - x, 2h) dx' = \int_{\Omega_h} A(x') \nabla_{x'} W(x - x', 2h) dx' \]

\[ \nabla_x A(x_i) \approx \sum_j A(x_j) \nabla W(x_i - x_j, 2h) V(x_j) \]

\[ \nabla_x A(x_i) \approx \sum_j \frac{m(x_i, x_j)}{\rho(x_j)} A(x_j) \nabla W(x_i - x_j, 2h) \]

Gauss theorem

S is the surface of \( \Omega \)

W = 0 on the surface S

\[ \nabla_x A(x) \approx -\int_{\Omega_h} A(x') \nabla_{x'} W(x' - x, 2h) dx' = \int_{\Omega_h} A(x') \nabla_{x'} W(x - x', 2h) dx' \]

\[ \nabla_x A(x_i) \approx \sum_j A(x_j) \nabla W(x_i - x_j, 2h) V(x_j) \]

\[ \nabla_x A(x_i) \approx \sum_j \frac{m(x_i, x_j)}{\rho(x_j)} A(x_j) \nabla W(x_i - x_j, 2h) \]
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 \( A_i = A_j \) in case of erroneous sampling
Gradient - Anti-symmetric Form

- Momentum-preserving form
\[ \nabla \left( \frac{A_i}{\rho_i} \right) = \frac{\rho_i \nabla A_i - A_i \nabla \rho_i}{\rho_i^2} = \nabla A_i - \frac{A_i \nabla \rho_i}{\rho_i^2} \]
\[ \nabla A_i = \rho_i \left( \nabla \left( \frac{A_i}{\rho_i} \right) + \frac{A_i \nabla \rho_i}{\rho_i^2} \right) \]

- SPH approximation
\[ \nabla A_i = \rho_i \left( \sum_j \frac{m_j}{\rho_j} \frac{A_j}{\rho_j} \nabla W_{ij} + A_i \sum_j \frac{m_j}{\rho_j} \frac{\rho_j}{\rho_i^2} \nabla W_{ij} \right) \]
\[ = \rho_i \sum_j m_j \left( \frac{A_j}{\rho_j^2} + \frac{A_i}{\rho_i^2} \right) \nabla W_{ij} \]

- Applied to pressure gradient, linear and angular momentum is preserved for arbitrary samplings
\[ \mathbf{a}_i = m_j \left( \frac{A_i}{\rho_i^2} + \frac{A_j}{\rho_j^2} \right) \nabla W_{ij} = -m_i \left( \frac{A_i}{\rho_j^2} + \frac{A_j}{\rho_i^2} \right) \nabla W_{ji} = -\mathbf{a}_j \quad \nabla W_{ij} = -\nabla W_{ji} \]
Gradient – Symmetric Form

– Term that vanishes for constant function values
\[ \nabla (\rho_i A_i) = \rho_i \nabla (A_i) + A_i \nabla (\rho_i) \]
\[ \nabla A_i = \frac{1}{\rho_i} (\nabla (\rho_i A_i) - A_i \nabla \rho_i) \]

– SPH approximation
\[ \nabla A_i = \frac{1}{\rho_i} \left( \sum_j \frac{m_j}{\rho_j} \frac{A_j}{\rho_j} \nabla W_{ij} - A_i \sum_j \frac{m_j}{\rho_j} \rho_j \nabla W_{ij} \right) \]
\[ = \frac{1}{\rho_i} \sum_j m_j (A_j - A_i) \nabla W_{ij} = \frac{1}{\rho_i} \sum_j m_j A_{ji} \nabla W_{ij} \]

– Applied to velocity divergence, zero divergence for a constant velocity field is obtained for arbitrary samplings
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

\[
\nabla^2 A_i = 2 \sum_j \frac{m_j}{\rho_j} A_{ij} \frac{x_{ij} \cdot \nabla W_{ij}}{x_{ij} \cdot x_{ij} + 0.01h^2}
\]

\[
A_{ij} = A_i - A_j \quad x_{ij} = x_i - x_j
\]
Spatial Derivatives - Summary

- Original approximations

\[ \nabla A_i = \sum_j \frac{m_j}{\rho_j} A_j \nabla W_{ij} \quad \nabla^2 A_i = \sum_j \frac{m_j}{\rho_j} A_j \nabla^2 W_{ij} \]

- Currently preferred approximations

  - Robustness in case of particle disorder, i.e. \( \sum_j \nabla W_{ij} \neq 0 \)

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

Preserves linear and angular momentum

\[ \nu \nabla^2 \mathbf{v}_i = 2 \sum_j \frac{m_j}{\rho_j} \frac{\mathbf{v}_{ij} \cdot \mathbf{x}_{ij}}{\mathbf{x}_{ij} \cdot \mathbf{x}_{ij} + 0.01 h^2} \nabla W_{ij} \]

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

\[ \mathbf{v}_{ij} = \mathbf{v}_i - \mathbf{v}_j \quad \mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j \]
Some Kernel Properties

- In case of ideal sampling
  \[ \rho_i = \sum_j m_j W_{ij} = m_i \sum_j W_{ij} \quad m_i = m_j \]
  \[ m_i \sum_j W_{ij} = \rho_i = \frac{m_i}{V_i} \quad \Rightarrow \quad \sum_j W_{ij} = \frac{1}{V_i} = \frac{\rho_i}{m_i} \]
  \[ \nabla W_{ij} = -\nabla W_{ji} \quad \nabla W_{ij} = \alpha_i \left( \frac{x_{ij}}{\|x_{ij}\|} \right) \ldots \]
  \[ \sum_j \nabla W_{ij} = 0 \]
  \[ \sum_j (x_i - x_j) \otimes \nabla W_{ij} = -\frac{1}{V_i} \cdot \mathbf{I} \]
- Can be used for test purposes
Kernel Illustration

- 1D illustration

\[ W(q) = \begin{cases} 
(2 - q)^3 - 4(1 - q)^3 & 0 \leq q < 1 \\
(2 - q)^3 & 1 \leq q < 2 \\
0 & q \geq 2
\end{cases} \quad q = \frac{\|x_i - x_j\|}{h} \]

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

\[ W(1) = \frac{1}{6h} (2 - 1)^3 = \frac{1}{6h} \]

\[ W(2) = 0 \]

\[ \sum_j W_{ij} = W(0) + 2W(1) + 2W(2) = \frac{1}{h} \]
Kernel Illustration

- 2D illustration

\[ W(q) = \frac{5}{14\pi h^2} \begin{cases} 
(2 - q)^3 - 4(1 - q)^3 & 0 \leq q < 1 \\
(2 - q)^3 & 1 \leq q < 2 \\
0 & q \geq 2 
\end{cases} \]

\[ q = \frac{||x_i - x_j||}{h} \]

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

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

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

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

- Density computation is not an interpolation of the function $m$, but detects erroneous sampling.

$\rho^0 = \frac{m}{V_0} = \sum_j m W_{ij}$

$\rho_i = \sum_j m W_{ij} > \rho_0 = \frac{m}{V_0}$

Correct sampling

Dense sampling
(Kernel contributions do not sum up to $1/V$)
Outline

– Concept of an SPH fluid simulator
– Momentum equation
– SPH basics
– Neighborhood search
– Boundary handling
– Incompressibility
SPH Simulation Step With 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
Neighbor Search

- For the computation of SPH sums in 3D, each particle needs to know at least 30-40 neighbors in each step
- Example setting
  - 30 million fluid particles
  - Up to 1 billion neighbors
  - 10000 simulation steps
  - Up to $10^{13}$ neighbors processed per simulation
- Efficient construction and processing of dynamically changing neighbor sets is essential
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
Characteristics

- 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
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 $O(1)$
Characteristics

– Neighbor storage is generally expensive
  – Might be avoided for, e.g., a low number of neighbor queries per 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
Outline

– Concept of an SPH fluid simulator
– Momentum equation
– SPH basics
– Neighborhood search
  – Uniform grid
  – Index sort
  – Spatial hashing
  – Discussion
– ...

University of Freiburg – Computer Science Department – 82
Concept

- Particle is stored in a cell with coordinates \((k, l, m)\)
- In \(d\)-D, potential neighbors in \(3^d\) cells are queried to estimate actual neighbors
- 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

Edge length equals kernel support
Concept - Variant

- Verlet lists
  - Neighbor candidates are computed within a distance larger than the kernel support every $n^{th}$ step
  - Actual neighbors are computed from neighbor candidates in each step
  - Neighbor candidates are valid for $n$ steps
  - Motivated by temporal coherence: Particle does not move farther than its size in one step.
Concept - Variant

- Verlet lists
  - Proposed in 1967
  - Still popular in Lagrangian simulations
  - Acceleration data structure
    - Is only updated every $n^{th}$ step
    - Is memory-intensive, requires storage of a comparatively large number of neighbor candidates
Outline

– Concept of an SPH fluid simulator
– Momentum equation
– SPH basics
– Neighborhood search
  – Uniform grid
  – Index sort
  – Spatial hashing
  – Discussion
– ...

University of Freiburg – Computer Science Department – 86
Construction

- Compute cell index \( c = k + l \cdot K + m \cdot K \cdot L \) for all particles
  - \( K \) and \( L \) denote the number of cells in \( x \) and \( y \) direction
- Particles are sorted with respect to their cell index
- Each grid cell \((k, l, m)\) with index \( c \) stores a reference to the first particle in the sorted list

![Diagram](image)
**Construction**

Compute cell indices for particles and increment counter in C

<table>
<thead>
<tr>
<th>C</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>counter</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

Accumulate counters in C

<table>
<thead>
<tr>
<th>C</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>counter</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>

Associate particle $i$ with cell $j$: $L[j].counter.particle = i$

<table>
<thead>
<tr>
<th>C</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>counter</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>

...  

<table>
<thead>
<tr>
<th>C</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>counter</td>
<td>0</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td>8</td>
</tr>
</tbody>
</table>

**Table:**

<table>
<thead>
<tr>
<th>L</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>particle</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>L</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>particle</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>L</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>particle</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>L</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>particle</td>
<td>6</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>7</td>
<td>2</td>
<td>4</td>
<td>0</td>
</tr>
</tbody>
</table>
## Construction

<table>
<thead>
<tr>
<th>C</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>counter</td>
<td>0</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td>8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>L</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>particle</td>
<td>6</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>7</td>
<td>2</td>
<td>4</td>
<td>0</td>
</tr>
</tbody>
</table>

- Particles are sorted with respect to grid cell
- Counter points to first particle in a cell
- Difference of two subsequent counters indicates the particle number of a grid cell
Construction

- Two iterations over particles
- One iteration over grid cells
- Entire simulation domain has to be represented
- Parallelizable
- Memory allocations are avoided
- Constant memory consumption
Query

– For a particle
  – Indices to grid cell and to adjacent cells are computed
    (Once for all particles in the same grid cell)
  – All particles in grid cell and adjacent cells are tested
– Parallelizable
– Improved cache-hit rate
  – Particles in the same cell are close in memory
  – Particles of neighboring cells are not necessarily close in memory
Space-filling Curves

- Alternative computation for grid cell indices
- E.g., 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
**Sorting**

- Particle attributes and z-curve indices can be processed separately
- Handles (particle identifier, z-curve index) are sorted in each time step
  - Reduced memory transfer
  - Spatial locality is only marginally influenced due to temporal coherence
- Attribute sets are sorted every $n^{th}$ step
  - Restores spatial locality
Sorting

- Radix sort or insertion sort can be employed
  - $O(n)$ for almost sorted arrays
  - Due to temporal coherence, a small percentage of all particles change their cell, i.e. z-curve index, in each step
Z-Index Sort - Reordering

Particle color indicates memory location

Spatial compactness using a z-curve
Outline

- Concept of an SPH fluid simulator
- Momentum equation
- SPH basics
- Neighborhood search
  - Uniform grid
  - Index sort
  - Spatial hashing
  - Discussion
- ...
Spatial Hashing

- Hash function maps a grid cell to a hash cell
  - Infinite 3D domain is mapped to a finite 1D list
  - 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 $i$
Spatial Hashing

\[ i = h(c) \]  \quad \text{Hash function}

\[ 3 = h(0) \]
\[ 1 = h(1) \]
\[ 4 = h(2) \]
\[ 7 = h(3) \]

<table>
<thead>
<tr>
<th>particle</th>
<th>1</th>
<th>3</th>
<th>7</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5</td>
<td>2</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Spatial Hashing

- Large hash tables reduce number of hash collisions
  - Different spatial cells with the same hash value cause hash collisions which slow down the query
- Reduced memory allocations
  - Memory for $m$ entries is allocated for each hash cell
- Reduced cache-hit rate
  - Hash table is sparsely filled
  - Alternating filled and empty cells
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
- List of used cells is queried in the neighbor search
Compact Hashing - Construction

- Larger hash table compared to spatial hashing to reduce hash collisions
- Temporal coherence can be employed
  - List of used cells is not rebuilt, but updated
  - Particles with changed cell index are estimated
  - Particle is removed from the old cell and added to the new cell
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
Compact Hashing - Query

- Particles are sorted with respect to a z-curve every $n^{th}$ step
- After sorting, the list of used cells is 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
Compact Hashing - Reordering

Z-curve

preserving the spatial locality
improves the performance
Outline

- Concept of an SPH fluid simulator
- Momentum equation
- SPH basics
- Neighborhood search
  - Uniform grid
  - Index sort
  - Spatial hashing
  - Discussion
- ...
Comparison

- Measurements in ms for 130K particles
- Ongoing research
  - Focus on sorting, parallelization and vectorization
  - Octrees, k-D trees, BVHs can be realized with sorting

<table>
<thead>
<tr>
<th>Method</th>
<th>Construction</th>
<th>Query</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic grid</td>
<td>26</td>
<td>38</td>
<td>64</td>
</tr>
<tr>
<td>Index sort</td>
<td>36</td>
<td>29</td>
<td>65</td>
</tr>
<tr>
<td>Z-index sort</td>
<td>16</td>
<td>27</td>
<td>43</td>
</tr>
<tr>
<td>Spatial hashing</td>
<td>42</td>
<td>86</td>
<td>128</td>
</tr>
<tr>
<td>Compact hashing</td>
<td>8</td>
<td>32</td>
<td>40</td>
</tr>
</tbody>
</table>
Parallel Scaling

- Compact hashing
- Amdahl 0.95
- Spatial hashing

The diagram illustrates the speed up of parallel scaling for different hashing techniques as the number of threads increases. The speed up is measured on the y-axis, and the number of threads is shown on the x-axis. The graph shows a comparison between compact hashing, Amdahl 0.95, and spatial hashing, indicating the performance improvement with increased parallel processing.
Discussion

- Index sort
  - Fast construction based on sorting
  - Fast query
  - Particles are processed in the order of cell indices

- Z-index sort
  - Sorting with respect to a space filling curve improves cache-hit rate
Discussion

- Spatial hashing
  - Less efficient query due to hash collisions and due to the traversal of the sparsely filled hash table

- Compact hashing
  - Fast construction (or update due to temporal coherence)
  - Fast query due to the compact list of used cells, due to the hash-collision flag and due to the z-curve
Outline

- Concept of an SPH fluid simulator
- Momentum equation
- SPH basics
- Neighborhood search
- Boundary handling
- Incompressibility
Concept

- Boundaries are sampled with particles that contribute to density, pressure and pressure acceleration of the fluid.

\[ \rho_i < \rho_0 \]
\[ p_i = 0 \]
\[ p_{ib} = 0 \]
\[ F_{pi} = 0 \]

\[ \rho_i > \rho_0 \]
\[ p_i > 0 \]
\[ p_{ib} > 0 \]
\[ F_{pi} \neq 0 \]

- Boundary handling: How to compute \( \rho_i, p_i, p_{ib}, F_{pi} \)?
Several Layers with Uniform Boundary Samples

- Boundary particles are handled as static fluid samples

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

\[ m_i = m_{i_f} = m_{i_b} \]

\[ \rho_i = m_i \sum_{i_f} W_{ii_f} + m_i \sum_{i_b} W_{ii_b} \]

\[ p_i = k \left( \frac{\rho_i}{\rho_0} - 1 \right) \]

- Pressure acceleration

\[ a_i^p = -m_i \sum_{i_f} \left( \frac{p_i}{\rho_i^2} + \frac{p_{i_f}}{\rho_{i_f}^2} \right) \nabla W_{ii_f} - m_i \sum_{i_b} \left( \frac{p_i}{\rho_i^2} + \frac{p_{i_b}}{\rho_{i_b}^2} \right) \nabla W_{ii_b} \]

Boundary neighbors contribute to the density

All samples have the same size, i.e. same mass and rest density

University of Freiburg – Computer Science Department – 112
Pressure at Boundary Samples

- Pressure acceleration at boundaries requires pressure at boundary samples
- Various solutions, e.g. mirroring, extrapolation, PPE
- Mirroring
  - Formulation with unknown boundary pressure $p_{ib}$
  - $a_i^p = -m_i \sum_{i,j} \left( \frac{p_i}{\rho_i^2} + \frac{p_{ij}}{\rho_{ij}^2} \right) \nabla W_{ii,j} - m_i \sum_{i,b} \left( \frac{p_i}{\rho_i^2} + \frac{p_{ib}}{\rho_{ib}^2} \right) \nabla W_{ii,b}$
  - Mirroring of pressure and density from fluid to boundary $p_{ib} = p_i$
  - $a_i^p = -m_i \sum_{i,j} \left( \frac{p_i}{\rho_i^2} + \frac{p_{ij}}{\rho_{ij}^2} \right) \nabla W_{ii,j} - m_i \sum_{i,b} \left( \frac{p_i}{\rho_i^2} + \frac{p_{ib}}{\rho_{ib}^2} \right) \nabla W_{ii,b}$
Boundary Contribution to Pressure Acceleration

\[- \mathbf{a}_i^p = - \ldots - m_i \sum_{i_b} \left( \frac{p_i}{\rho_i^2} + \frac{p_i}{\rho_i^2} \right) \nabla W_{ii_b} = - \ldots - p_i \frac{2m_i}{\rho_i^2} \sum_{i_b} \nabla W_{ii_b} \]

\[- \sum_{i_b} \nabla W_{ii_b} \]

\[- \sum_{i_b} \nabla W_{ii_b} \]

\[-p_i \frac{2m_i}{\rho_i^2} \sum_{i_b} \nabla W_{ii_b} = 0 \]

\[-p_i \frac{2m_i}{\rho_i^2} \sum_{i_b} \nabla W_{ii_b} \neq 0 \]

\[\rho_i < \rho_0 \]

\[\rho_i > \rho_0 \]

\[p_i = 0 \]

\[p_i > 0 \]
One Layer of Uniform Boundary Samples

- Contributions of missing samples have to be added

\[
\rho_i = m_i \sum_{i_f} W_{ii_f} + m_i \sum_{i_b} W_{ii_b} + x
\]

\[
\rho_i = m_i \sum_{i_f} W_{ii_f} + \gamma_1 m_i \sum_{i_b} W_{ii_b}
\]

\[
\sum_{i_f} W_{ii_f} + \gamma_1 \sum_{i_b} W_{ii_b} = \frac{1}{V_i} \Rightarrow \gamma_1 = \frac{\frac{1}{V_i} - \sum_{i_f} W_{ii_f}}{\sum_{i_b} W_{ii_b}}
\]

Offset typically implemented as scaling coefficient

- Pressure acceleration

\[
a^p_i = -m_i \sum_{i_f} \left( \frac{p_i}{\rho_i^2} + \frac{p_{i_f}}{\rho_{i_f}^2} \right) \nabla W_{ii_f} - p_i \frac{2\gamma_2 m_i}{\rho_i^2} \sum_{i_b} \nabla W_{ii_b}
\]

\[
\sum_{i_f} \nabla W_{ii_f} + \gamma_2 \sum_{i_b} \nabla W_{ii_b} = 0 \Rightarrow \gamma_2 = -\frac{\sum_{i_f} \nabla W_{ii_f} \cdot \sum_{i_b} \nabla W_{ii_b}}{\sum_{i_b} \nabla W_{ii_b} \cdot \sum_{i_b} \nabla W_{ii_b}}
\]

Kernel gradient property
Correction of Missing Contributions

\[ \rho_i = m_0(W_{00} + W_{01} + W_{02}) \]
\[ a_i^p = -p_i \frac{2m_i}{\rho_i^2} (\nabla W_{01} + \nabla W_{02}) \]

\[ \rho_i = \gamma_1 m_0 (W_{00} + W_{01}) \]
\[ a_i^p = -p_i \frac{2\gamma_2 m_i}{\rho_i^2} \nabla W_{01} \]

- The motivation of \( \gamma_1 \) and \( \gamma_2 \) is to compensate contributions of missing samples to \( \rho, p, a^p \).
One Layer of Non-Uniform Boundary Samples

– Non-uniform contributions from boundary samples

\[ \rho_i = m_i \sum_{i_f} W_{iif} + \sum_{i_b} m_{ib} W_{iiib} \]

Fluid

Solid

Missing samples

– Pressure acceleration

\[ a_i^p = -m_i \sum_{i_f} \left( \frac{p_i}{\rho_i^2} + \frac{p_{if}}{\rho_i^2 \rho_{if}^2} \right) \nabla W_{iif} - p_i \frac{2 \gamma_2 m_i}{\rho_i^2} \sum_{i_b} \nabla W_{iiib} \]

Non-uniform sizes, i.e. masses of boundary samples

Contribution, i.e. mass of a boundary sample is approximated from its boundary neighbors
One Layer of Non-Uniform Boundary Samples

For perfect sampling

\[ V_{ib}^0 = h^3 = \frac{1}{\sum_{ib} W_{ibibb}} \]

For arbitrary sampling

\[ m_{ib} = \rho_0 \frac{\gamma_1}{\sum_{ib} \gamma_{ibibb}} \]

In 3D, \( \gamma_1 = 0.7 \)
Typical Boundary Representation

Boundary samples

Color-coded volume of boundary samples
Rigid-Fluid Coupling

Dam break

20M fluid particles
Rigid-Fluid Coupling
Summary

- Boundary is sampled with static fluid particles
- One layer of non-uniform samples
  - Arbitrary triangulated meshes can be used as boundary
  - Missing contributions to fluid density and pressure acceleration have to be corrected
  - Non-uniform boundary samples can be handled
  - Pressure is mirrored from fluid to boundary