Simulation in Computer Graphics

Particle-based Fluid Simulation

Matthias Teschner

Computer Science Department
University of 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
Application (Commercials)

Copyright NHB Studios, Berlin, Hamburg, Dusseldorf
Application (with FIFTY2 Technology)

PreonLab: Drive Through

PreonLab, FIFTY2 Technology GmbH, www.youtube.com -> fifty2
Application (with FIFTY2 Technology)

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

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

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

![Fluid body](image1.png)
![Set of fluid parcels](image2.png)

$x, v, m, V, \rho, p$
**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 $\rho_0$
- 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 $x_i$ 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
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

\[
\mathbf{x}(t) \quad \mathbf{v}(t)
\]

\[
\mathbf{v}(t + \Delta t) = \mathbf{v}(t) + \Delta t \cdot \mathbf{a}(t)
\]

\[
\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \Delta t \cdot \mathbf{v}(t + \Delta t)
\]
Accelerations

- gravity $g$
- 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
  
  \[ x(t) = 0 \]
  \[ \mathbf{v}(t) = 0 \]
  
  Gravity

- pressure acceleration avoids the volume / density change

  \[ -\frac{1}{\rho} \nabla p = -g \]

  Pressure acceleration
**Simulation Step - Example**

- **current state**
  
  \[
  x(t) = 0 \\
  v(t) = 0
  \]

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

- **subsequent state**
  
  \[
  x(t + \Delta t) = x(t) + \Delta t \cdot v(t) = 0
  \]
  
  \[
  v(t + \Delta t) = v(t) + \Delta t \cdot a(t) = 0
  \]
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
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 forces $F_i(t) = \sum_j \ldots$ as sums of neighbors
- advect the particles, e.g. Euler-Cromer
- determine neighbors of particle $x_i(t + \Delta t)$
- ...

![Diagram of particle simulation process]
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 \( \mathbf{v}_i \) is governed by the Lagrange form of the Navier-Stokes equation
  \[
  \frac{d\mathbf{v}_i}{dt} = -\frac{1}{\rho_i} \nabla p_i + \nu \nabla^2 \mathbf{v}_i + \frac{\mathbf{F}_{\text{other}}}{m_i}
  \]
  - 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
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

- $\nu \nabla^2 v_i$ : 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 $\nu \approx 10^{-6} \text{m}^2 \cdot \text{s}^{-1}$: larger friction is less realistic, but can improve the stability, dynamic viscosity $\eta = \mu = \nu \cdot \rho_0$

- $\frac{F_{\text{other}}}{m_i}$ : e.g., gravity, boundary handling
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 $\mathbf{v}_i$

\[ v_i(x_i, y_i, z_i, t) = (u_i, v_i, w_i) \]
\[ x_i(t) = (x_i, y_i, z_i) \]
\[ v_i(x_i + \Delta t \cdot u_i, y_i + \Delta t \cdot v_i, z_i + \Delta t \cdot w_i, t + \Delta t) \]
\[ x_i(t + \Delta t) = (x_i + \Delta t \cdot u_i, y_i + \Delta t \cdot v_i, z_i + \Delta t \cdot w_i) \]
Moving Parcels vs. Static Cells

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

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 \( \frac{dx_i}{dt} = v_i \) and \( \frac{dv_i}{dt} = -\frac{1}{\rho_i} \nabla p_i + \nu \nabla^2 v_i + \frac{F_i}{m_i} \)
  - \( \rho_i \), \( -\frac{1}{\rho_i} \nabla p_i \) and \( \nu \nabla^2 v_i \) are computed with SPH
- SPH is typically used in Lagrangian, mesh-free approaches, but not limited to
**SPH Interpolation**

- Quantity $A_i$ at an arbitrary position $x_i$ is approximately computed with a set of known quantities $A_j$ at sample positions $x_j$
  
  $A_i = \sum_j V_j A_j W_{ij} = \sum_j \frac{m_j}{\rho_j} A_j W_{ij}$

  - $x_i$ is not necessarily a sample position
  - If $x_i$ is a sample position, it contributes to the sum

- $W_{ij}$ is a kernel function that weights the contributions of sample positions $x_j$ according to their distance to $x_i$

  $W_{ij} = W \left( \frac{\|x_i - x_j\|}{h} \right) = W(q)$

  - $d$ is the dimensionality of the simulation domain
  - $h$ is the so-called smoothing length
SPH Interpolation – 2D

Particle of interest

\( W \)

\( h \)

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: $\alpha = \frac{1}{4\pi h^3}$)
  \[
  W(q) = \alpha \begin{cases} 
    (2 - q)^3 - 4(1 - q)^3 & 0 \leq q < 1 \\
    (2 - q)^3 & 1 \leq q < 2 \\
    0 & q \geq 2
  \end{cases}
  \]
  \[
  q = \frac{\|x_i - x_j\|}{h}
  \]
- 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
Kernel Function in 1D

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

\[ W(x) = \frac{2}{3h} e^{-\frac{x^2}{2(0.59h)^2}} \]
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**
  \[ \nabla A_i = \rho_i \sum_j m_j \left( \frac{A_i}{\rho_i^2} + \frac{A_j}{\rho_j^2} \right) \nabla W_{ij} \]
  \[ \nabla^2 A_i = 2 \sum_j \frac{m_j}{\rho_j} A_{ij} \frac{x_{ij} \cdot \nabla W_{ij}}{x_{ij} \cdot x_{ij} + 0.01 h^2} \]
  \[ \nabla \cdot \mathbf{A}_i = -\frac{1}{\rho_i} \sum_j m_j \mathbf{A}_{ij} \nabla W_{ij} \]

\[ A_{ij} = A_i - A_j \quad \mathbf{A}_{ij} = \mathbf{A}_i - \mathbf{A}_j \quad \mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j \]
Kernel Derivative in 1D

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

\[ \nabla W(x) = -\frac{x}{6h^2|x|} \begin{cases} 
-3(2 - \frac{|x|}{h})^2 + 12(1 - \frac{|x|}{h})^2 & 0 \leq |x| < h \\
-3(2 - \frac{|x|}{h})^2 & h \leq |x| < 2h \\
0 & |x| \geq 2h 
\end{cases} \]
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
    \[ \frac{d\rho_i}{dt} = -\rho_i \nabla \cdot \mathbf{v}_i \]
  \[ \frac{d\rho_i}{dt} = \sum_j m_j \mathbf{v}_{ij} \nabla W_{ij} \]
  - no issues for incomplete neighborhoods
  - drift, i.e., less accurate for large time steps
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 $\rho_i$ using a state equation, e.g. $p_i = k \left( \left( \frac{\rho_i}{\rho_0} \right)^7 - 1 \right)$
  - $\rho_0$ 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
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 \(\rho_i\)

for all particle \(i\) do

\(\mathbf{F}_{i_{\text{pressure}}} = -\frac{m_i}{\rho_i} \nabla p_i\)

\(\mathbf{F}_{i_{\text{viscosity}}} = m_i \nu \nabla^2 \mathbf{v}_i\)

\(\mathbf{F}_{i_{\text{other}}} = m_i \mathbf{g}\)

\(\mathbf{F}_i(t) = \mathbf{F}_{i_{\text{pressure}}} + \mathbf{F}_{i_{\text{viscosity}}} + \mathbf{F}_{i_{\text{other}}}\)

for all particle \(i\) do

\(\mathbf{v}_i(t + \Delta t) = \mathbf{v}_i(t) + \Delta t \mathbf{F}_i(t)/m_i\)

\(\mathbf{x}_i(t + \Delta t) = \mathbf{x}_i(t) + \Delta t \mathbf{v}_i(t + \Delta t)\)
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 \( \rho_i \)

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

- for all particle \( i \) do 
  \( \mathbf{v}_i(t + \Delta t) = \mathbf{v}_i(t) + \Delta t \mathbf{a}_i(t) \)  
  \( \mathbf{x}_i(t + \Delta t) = \mathbf{x}_i(t) + \Delta t \mathbf{v}_i(t + \Delta t) \)
Accelerations with SPH

\[ a_{i}^{\text{pressure}} = - \sum_{j} m_{j} \left( \frac{p_{i}}{\rho_{i}^{2}} + \frac{p_{j}}{\rho_{j}^{2}} \right) \nabla W_{ij} \]

\[ a_{i}^{\text{viscosity}} = 2 \nu \sum_{j} \frac{m_{j}}{\rho_{j}} v_{ij} \frac{x_{ij} \cdot \nabla W_{ij}}{x_{ij} \cdot x_{ij} + 0.01 h^2} \]

\[ a_{i}^{\text{other}} = g \]
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., $\Delta t \leq \lambda \frac{h}{\|v_{\text{max}}\|}$ with $\lambda = 0.1$ and particle spacing $h$
  - motivation: for $\lambda \leq 1$, a particle moves less than its diameter per time step
Simulation in Computer Graphics

Particle-based Fluid Simulation

Matthias Teschner

Computer Science Department
University of 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 $\rho_i$
for all particle $i$ do
\[ \mathbf{a}_i^{\text{pressure}} = -\frac{1}{\rho_i} \nabla p_i \]
\[ \mathbf{a}_i^{\text{viscosity}} = \nu \nabla^2 \mathbf{v}_i \]
\[ \mathbf{a}_i^{\text{other}} = \mathbf{g} \]
\[ \mathbf{a}_i(t) = \mathbf{a}_i^{\text{pressure}} + \mathbf{a}_i^{\text{viscosity}} + \mathbf{a}_i^{\text{other}} \]

SPH approximations

Navier-Stokes equation

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

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

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

- **momentum equation**
  \[
  \frac{dv_i}{dt} = -\frac{1}{\rho_i} \nabla p_i + \nu \nabla^2 v_i + \frac{F_{\text{other}}}{m_i}
  \]

- **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

---

**Diagram**

- Fluid element
- Pressure
- Normal stress
- Volume deviation
- Shear stress
- Friction
- Velocity difference
- Resulting force acting on the fluid element

- \[ F_{\text{other}} \]
- \[ -\frac{m_i}{\rho_i} \nabla p_i \]
- \[ m_i \nu \nabla^2 v_i \]
Pressure Force in x-direction

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

\[ \rho \, dy \, dz \quad - \left( \rho + \frac{\partial \rho}{\partial x} \, dx \right) \, dy \, dz \]

- resulting pressure force

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

- force at particle $i$

$$F_{i,\text{pressure}} = -\left( \begin{array}{c} \frac{\partial p_i}{\partial x_{i,x}} \\ \frac{\partial p_i}{\partial x_{i,y}} \\ \frac{\partial p_i}{\partial x_{i,z}} \end{array} \right) \quad V_i = -\nabla p_i \quad V_i = -\frac{m_i}{\rho_i} \nabla p_i$$

- respective acceleration

$$a_{i,\text{pressure}} = \frac{F_{i,\text{pressure}}}{m_i} = -\frac{1}{\rho_i} \nabla p_i$$
Cauchy Momentum Equation

- Lagrange form
  \[ \frac{d\mathbf{v}}{dt} = \frac{1}{\rho} \nabla \cdot \sigma + \frac{\mathbf{F}_{\text{other}}}{m} \]

- \( \sigma \) 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

- \( \tau \) is the viscous stress tensor

\[ \tau = \nu \left( \begin{array}{ccc} \frac{\partial u}{\partial x} + \frac{\partial u}{\partial x} & \frac{\partial v}{\partial x} + \frac{\partial u}{\partial y} & \frac{\partial w}{\partial x} + \frac{\partial u}{\partial z} \\ \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} + \frac{\partial v}{\partial y} & \frac{\partial w}{\partial y} + \frac{\partial v}{\partial z} \\ \frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} & \frac{\partial w}{\partial z} + \frac{\partial w}{\partial y} & \frac{\partial w}{\partial z} + \frac{\partial w}{\partial z} \end{array} \right) \]

- \( \nabla \cdot \tau = \nu \nabla^2 \mathbf{v} \) is the resulting viscosity force per volume

- \( \frac{d\mathbf{v}_i}{dt} = -\frac{1}{\rho_i} \nabla p_i + \nu \nabla^2 \mathbf{v}_i + \frac{\mathbf{F}_{i\text{other}}}{m_i} \)
Outline

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

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

- integral should be normalized (unity condition)
  \[ \int_{\Omega} W(x - x', h) \, dx' = 1 \]
- support should be compact
  \[ W(x - x', h) = 0 \text{ for } \|x - x'\| > h \]
- should be symmetric
  \[ W(x - x', h) = W(x' - x, h) \]
- should be non-negative
  \[ W(x - x', h) \geq 0 \]
- should converge to the Dirac delta for \( h \to 0 \)
- should be differentiable
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: $\alpha = \frac{1}{4\pi h^3}$)

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

- 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
First Kernel Derivative

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

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

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

\[ \nabla q = \frac{x_{ij}}{\|x_{ij}\|} \]

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

\[ \nabla W_{ij} = \alpha \frac{x_{ij}}{\|x_{ij}\|} \begin{cases} 
-3(2-q)^2 + 12(1-q)^2 & 0 \leq q < 1 \\
-3(2-q)^2 & 1 \leq q < 2 \\
0 & q \geq 2 
\end{cases} \]
Second Kernel Derivative

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

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

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

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

\[ \nabla \cdot (\nabla q) = \frac{2}{h \|x\|} \]

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

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

- shape close to a Gaussian, e.g.

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

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

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

- 1D: integration over a line segment \[ 2 \int_0^2 \tilde{W} (q) h \, dq = \]

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

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

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

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

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

- Consider a limited number of samples / particles \( x_j \) representing a mass \( m(x_j) = \rho(x_j)V(x_j) \)

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

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

- Typical notation

  \[ A_i = \sum_j \frac{m_j}{\rho_j} A_j W_{ij} \]
Spatial Derivatives

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

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

  \text{W is symmetric}

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

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

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

  \text{Gauss theorem}

  \text{S is the surface of } \Omega

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

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

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

- \nabla_x A(x_i) \approx \sum_j \frac{m(x_j)}{\rho(x_j)} A(x_j) \nabla W(x_i - x_j, h)
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 \( A_i = A_j \)
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**

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

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

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

example with two particles i and j

\[
a_i = m_j \left( \frac{A_i}{\rho_i^2} + \frac{A_j}{\rho_j^2} \right) \nabla W_{ij} = -m_i \left( \frac{A_j}{\rho_j^2} + \frac{A_i}{\rho_i^2} \right) \nabla W_{ji} = -a_j \quad \nabla W_{ij} = -\nabla W_{ji}
\]
Gradient (Symmetric)

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

- 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

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

\[A_{ij} = A_i - A_j\]

\[x_{ij} = x_i - x_j\]
Spatial Derivatives - Summary

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

- currently preferred approximations
  - improved robustness in case of particle disorder, i.e. \( \sum_j \nabla W_{ij} \neq 0 \)

- \[ \nabla p_i = \rho_i \sum_j m_j \left( \frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) \nabla W_{ij} \]
  preserves linear and angular momentum

- \[ \nabla^2 v_i = 2 \sum_j \frac{m_j}{\rho_j} v_{ij} \frac{x_{ij} \cdot \nabla W_{ij}}{x_{ij} \cdot x_{ij} + 0.01h^2} \]
  improved robustness as it avoids the second kernel derivative

- \[ \nabla \cdot v_i = -\frac{1}{\rho_i} \sum_j m_j v_{ij} \nabla W_{ij} \]
  zero for uniform velocity field

- \[ v_{ij} = v_i - v_j \]
  \[ x_{ij} = x_i - x_j \]
Kernel Properties

- in case of ideal sampling
  \[ \rho_i = \sum_j m_j W_{ij} = m_i \sum_j W_{ij} \quad m_i = m_j \]
  \[ m_i \sum_j W_{ij} = \rho_i = \frac{m_i}{V_i} \quad \Rightarrow \quad \sum_j W_{ij} = \frac{1}{V_i} \]
  \[ \nabla W_{ij} = -\nabla W_{ji} \quad \nabla W_{ij} = \alpha \frac{x_{ij}}{\|x_{ij}\| h} \cdots \]
  \[ \sum_j \nabla W_{ij} = 0 \]
Kernel Illustration

- **1D illustration**

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

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

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

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

**correct sampling**

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

**dense sampling**

(kernel contributions do not sum up to 1/$V$)
Simulation in Computer Graphics

Particle-based Fluid Simulation

Matthias Teschner

Computer Science Department
University of 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 $\rho_i$
  for all particle i do
    $a_i^{\text{pressure}} = -\frac{1}{\rho_i} \nabla p_i$
    $a_i^{\text{viscosity}} = \nu \nabla^2 \mathbf{v}_i$
    $a_i^{\text{other}} = \mathbf{g}$
    $a_i(t) = a_i^{\text{pressure}} + a_i^{\text{viscosity}} + a_i^{\text{other}}$

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

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

$\mathbf{a}_i^{\text{viscosity}} = 2\nu \sum_j \frac{m_j}{\rho_j} \mathbf{v}_{ij} \frac{\mathbf{x}_{ij} \cdot \nabla W_{ij}}{\mathbf{x}_{ij} \cdot \mathbf{x}_{ij} + 0.01h^2}$

SPH approximations

Navier-Stokes equation
Outline

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

- 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
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
Basic Grid

- particle is stored in a cell with coordinates \((k, l, m)\)
- 27 cells are queried in the neighborhood search \((k \pm 1, l \pm 1, m \pm 1)\)
- 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
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
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 and sorted particles]
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:
  - $S[\text{--}C[j]]=i$
    - stores the particles in reversed order into S
    - after insertion C contains the correct offsets
Construction

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

particles colored according to their location in memory

spatial compactness is enforced using a z-curve
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
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 $i$
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
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 $O(m \cdot k)$ to $O(m + n \cdot k)$ with $m \gg n$
- 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)
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
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
Compact Hashing - Reordering
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
Comparison

<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>

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

- Our system
- Amdahl 0.95
- Spatial hashing

Speed up vs. threads graph.
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
References

- z-index sort, compact hashing

- index sort

- spatial hashing
Simulation in Computer Graphics

Particle-based Fluid Simulation

Matthias Teschner

Computer Science Department
University of 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 \(\rho_i\)
  - for all particle \(i\) do
    - \(a_i^{\text{pressure}} = -\sum_j m_j \left( \frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) \nabla W_{ij}\)
    - \(a_i^{\text{viscosity}} = 2\nu \sum_j \frac{m_j}{\rho_j} v_{ij} \frac{x_{ij} \cdot \nabla W_{ij}}{x_{ij} \cdot x_{ij} + 0.01h^2}\)
    - \(a_i^{\text{other}} = g\)
    - \(a_i(t) = a_i^{\text{pressure}} + a_i^{\text{viscosity}} + a_i^{\text{other}}\)
  - for all particle \(i\) do
    - \(v_i(t + \Delta t) = v_i(t) + \Delta t a_i(t)\)
    - \(x_i(t + \Delta t) = x_i(t) + \Delta t v_i(t + \Delta t)\)

SPH approximations
Outline

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

- rrigids are uniformly sampled with particles

\[
\rho_i = \sum_{i_f} m_{i_f} W_{ii_f} + \sum_{i_b} m_{i_b} W_{ii_b}
\]
\[
m_i = m_{i_f} = m_{i_b}
\]
\[
\rho_i \approx m_i \sum_{i_f} W_{ii_f} + m_i \sum_{i_b} W_{ii_b}
\]
\[
p_i = \ldots
\]

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

\[
a_p^i \approx -m_i \sum_{i_f} \left( \frac{p_i}{\rho_0^2} + \frac{p_{i_f}}{\rho_0^2} \right) \nabla W_{ii_f} - m_i \sum_{i_b} \left( \frac{p_i}{\rho_0^2} + \frac{p_{i_b}}{\rho_0^2} \right) \nabla W_{ii_b}
\]
**Missing Contributions**

- rigid are uniformly sampled with particles

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

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

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

\[ a_i^p \approx -m_i \sum_{i_f} \left( \frac{p_i}{\rho_0^2} + \frac{p_{i_f}}{\rho_0^2} \right) \nabla W_{ii_f} - m_i \gamma_2 \sum_{i_b} \left( \frac{p_i}{\rho_0^2} + \frac{p_i}{\rho_0^2} \right) \nabla W_{ii_b} \]

\[ \sum_{i_f} \nabla W_{ii_f} + \gamma_2 \sum_{i_b} \nabla W_{ii_b} = 0 \]
Non-uniform Sampling

- rigid bodies are non-uniformly sampled with particles

\[ \rho_i \approx m_i \sum_{i_f} W_{ii_f} + \gamma_1 \sum_{i_b} m_{ib} W_{ii_b} \]

\[ \frac{1}{V_{ib}} \approx \gamma_3 \sum_{i_{bb}} W_{ib_{ib}} \]

\[ m_{ib} = \frac{\rho_0}{\gamma_3 \sum_{i_{bb}} W_{ib_{ib}}} \]

\[ \mathbf{a}_i \approx -m_i \sum_{i_f} \left( \frac{p_i}{\rho_0^2} + \frac{p_{if}}{\rho_0^2} \right) \nabla W_{ii_f} - \gamma_2 \sum_{i_b} m_{ib} \left( \frac{p_i}{\rho_0^2} + \frac{p_i}{\rho_0^2} \right) \nabla W_{ii_b} \]
Non-uniform Sampling

color-coded volume of boundary particles
Rigid-Fluid Coupling

www.youtube.com/cgfreiburg
Rigid-Fluid Coupling

www.youtube.com/cgfreiburg
References

- **rigid-fluid coupling**

- **elastic-fluid coupling**
Outline

- concept of an SPH fluid simulator
- momentum equation
- SPH basics
- neighborhood search
- boundary handling
- incompressibility
- surface reconstruction
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
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.
  - \( p_i = k(\rho_i - \rho_0) \quad p_i = k\rho_i \quad p_i = k\left(\frac{\rho_i}{\rho_0} - 1\right) \) \( k \) is user-defined
  - In graphics referred to as compressible SPH
  - \( p_i = k_1\left(\frac{\rho_i}{\rho_0}\right)^{k_2} - 1 \) \( k_1, k_2 \) are user-defined
  - In graphics referred to as weakly compressible SPH
- Compressibility is governed by the stiffness constant(s) and limits the time step

University of Freiburg – Computer Science Department – Computer Graphics - 115
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 $\rho_i$ with a state equation

- for all particle $i$ do
  $F_i^{\text{pressure}} = -\frac{m_i}{\rho_i} \nabla p_i$
  $F_i^{\text{viscosity}} = m_i \nu \nabla^2 v_i$
  $F_i^{\text{other}} = m_i g$
  $F_i(t) = F_i^{\text{pressure}} + F_i^{\text{viscosity}} + F_i^{\text{other}}$

- for all particle $i$ do
  $v_i(t + \Delta t) = v_i(t) + \Delta t F_i(t) / m_i$
  $x_i(t + \Delta t) = x_i(t) + \Delta t v_i(t + \Delta t)$
**SESPH with Splitting**

- compute pressure after advecting the particles with non-pressure forces

**concept**
- compute all non-pressure forces $F_{i \text{np}}(t)$
- compute intermediate velocity $v_i^* = v_i(t) + \Delta t \frac{F_{i \text{np}}}{m_i}$
- compute intermediate position $x_i^* = x_i(t) + \Delta t v_i^*$
- compute intermediate density $\rho_i^*(x_i^*)$
- compute pressure $p_i$ from intermediate density $\rho_i^*$ using an EOS
- compute final velocity $v_i(t + \Delta t) = v_i^* - \Delta t \frac{1}{\rho_i^*} \nabla p_i$

**motivation**
- consider competing forces
- take (positive or negative) effects of non-pressure forces into account when computing the pressure forces
**SESPH with Splitting**

- **for all particle i do**
  - find neighbors $j$
- **for all particle i do**
  - $F_{i}^{\text{viscosity}} = m_i \nu \nabla^2 v_i$
  - $F_{i}^{\text{other}} = m_i g$
  - $v_i^* = v_i(t) + \Delta t \frac{F_{i}^{\text{viscosity}} + F_{i}^{\text{other}}}{m_i}$
- **for all particle i do**
  - $\rho_i^* = \sum_j m_j W_{ij} + \Delta t \sum_j m_j (v_i^* - v_j^*) \nabla W_{ij}$
  - compute $p_i$ using $\rho_i^*$
- **for all particle i do**
  - $F_{i}^{\text{pressure}} = -\frac{m_i}{\rho_i^*} \nabla p_i$
- **for all particle i do**
  - $v_i(t + \Delta t) = v_i^* + \Delta t F_{i}^{\text{pressure}} / m_i$
  - $x_i(t + \Delta t) = x_i(t) + \Delta t v_i(t + \Delta t)$

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

- **continuity equation**
  \[
  \frac{D\rho_i}{Dt} = -\rho_0 \nabla \cdot \mathbf{v}_i
  \]
  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 \( \mathbf{v}_i^* \)**
  \[
  \rho_i^* = \sum_j m_i W_{ij} + \Delta t \sum_j m_j (\mathbf{v}_i^* - \mathbf{v}_j^*) \nabla W_{ij}
  \]
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
Iterative SESPH with Splitting

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

- **for all particle** $i$ do
  - $F_{i \text{viscosity}}^i = m_i \nu \nabla^2 v_i$
  - $F_{i \text{other}}^i = m_i g$
  - $v_i^* = v_i(t) + \Delta t \frac{F_{i \text{viscosity}}^i + F_{i \text{other}}^i}{m_i}$
  - $x_i^* = x_i(t) + \Delta tv_i^*$

repeat

- **for all particle** $i$ do
  - compute $\rho_i^*$ using $x_i^*$
  - compute $p_i$ using $\rho_i^*$, e.g. $p_i = k(\rho_i^* - \rho_0)$

compute $\rho_{\text{err}}$, e.g. average or maximum

- **for all particle** $i$ do
  - $F_{i \text{pressure}}^i = -\frac{m_i}{\rho_i^*} \nabla p_i$
  - $v_i^* = v_i^* + \Delta t \frac{F_{i \text{pressure}}^i}{m_i}$
  - $x_i^* = x_i^* + \Delta t^2 \frac{F_{i \text{pressure}}^i}{m_i}$

until $\rho_{\text{err}} < \eta$  
user-defined density error

- **for all particle** $i$ do
  - $v_i(t + \Delta t) = v_i^*$
  - $x_i(t + \Delta t) = x_i^*$
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)
    - $$\Delta x_i = -\frac{1}{\rho_0} \sum_j \left( \frac{p_i}{\beta_i} + \frac{p_j}{\beta_j} \right) \nabla W_{ij} \quad \beta$$ is a pre-computed constant
- different EOS and stiffness constants are used
  - $$k = \frac{\rho_i^* r_i^2}{2\rho_0 \Delta t^2} \text{ in local Poisson SPH}$$
  - $$k = \frac{m_i^2 \Delta t^2 \sum_j \nabla W_{ij}^0 \cdot \sum_j \nabla W_{ij}^0 + \sum_j (\nabla W_{ij}^0 \cdot \nabla W_{ij}^0)}{2\rho_0^2} \text{ in PCISPH} \quad W^0 \text{ is precomputed}$$
  - $$k = 1 \text{ in PBF } (p_i = \frac{\rho_i}{\rho_0} - 1)$$
PCISPH - Motivation

- density at the next timestep should be rest density

\[ \rho(t + \Delta t) = \rho_0 = \sum_j m_i W_{ij} + \Delta t \sum_j m_j (v_i^* - v_j^*) \nabla W_{ij} \]

\[ + \Delta t \sum_j m_j (v_i^p - v_j^p) \nabla W_{ij} \]

- \( v_i^p = \Delta t \frac{F_i^p}{m_i} \)

\[ F_i^p = -m_i \sum_j \left( \frac{p_i}{\rho_i^2} - \frac{p_j}{\rho_j^2} \right) \nabla W_{ij} \approx -m_i^2 \frac{2p_i}{\rho_0^2} \sum_j \nabla W_{ij} \]

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

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

\[ F_{j|i}^p = -m_i \left( \frac{p_i}{\rho_i^2} - \frac{p_j}{\rho_j^2} \right) \nabla W_{ij} \approx -m_i^2 \frac{2p_i}{\rho_0^2} \sum_j \nabla W_{ij} \]
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
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
- \( \rho_i^* \) 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
Projection Schemes - Derivation

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

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

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

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

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

\[ \nabla \cdot \frac{\mathbf{v}_i(t + \Delta t) - \mathbf{v}_i^*}{\Delta t} = -\nabla \cdot \frac{1}{\rho_i} \nabla p_i \]
**Projection Schemes - Derivation**

\[ \nabla \cdot \frac{v_i(t+\Delta t) - v_i^*}{\Delta t} = -\nabla \cdot \frac{1}{\rho_i} \nabla p_i \]

\[ \nabla \cdot v_i(t + \Delta t) - \nabla \cdot v_i^* = -\nabla \cdot \Delta t \frac{1}{\rho_i} \nabla p_i \]

\[ \nabla \cdot v_i(t + \Delta t) = 0 \quad \text{divergence of the velocity at the next time step should be zero.} \]

\[ \nabla \cdot v_i^* = \Delta t \frac{1}{\rho_i} \nabla^2 p_i \]
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
**Implicit Incompressible SPH Derivation**

- **discretizing the continuity equation**
  \[
  \frac{D\rho_i(t+\Delta t)}{Dt} = -\rho_i(t + \Delta t) \nabla \cdot \mathbf{v}_i(t + \Delta t) \quad \text{to}
  \]
  \[
  \frac{\rho_i(t+\Delta t) - \rho_i(t)}{\Delta t} = \sum_j m_j \mathbf{v}_{ij}(t + \Delta t) \nabla W_{ij}(t)
  \]
  forward difference  
  SPH

- **unknown velocity** \( \mathbf{v}_i(t + \Delta t) \) can be rewritten using known non-pressure accel. \( \mathbf{a}_{i}^{\text{nonp}}(t) \) and unknown pressure accel. \( \mathbf{a}_{i}^{p}(t) \)

\[
\mathbf{v}_i(t + \Delta t) = \mathbf{v}_i(t) + \Delta t \mathbf{a}_{i}^{\text{nonp}}(t) + \Delta t \mathbf{a}_{i}^{p}(t)
\]

predicted velocity only using non-pressure forces
Implicit Incompressible SPH

Derivation

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

and \( \rho_i^* = \rho_i(t) + \Delta t \sum_j m_j v_{ij}^* \nabla W_{ij}(t) \)

predicted density, if only non-pressure forces are applied

the discretized continuity equation can be written as

\[
\frac{\rho_0 - \rho_i^*}{\Delta t} = \Delta t \sum_j m_j (a_i^p(t) - a_j^p(t)) \nabla W_{ij}(t)
\]

with unknown pressure accel. \( a_i^p(t) \)

unknown pressure accel. \( a_i^p(t) \) can be rewritten using unknown pressures \( p_i(t) \)

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

resulting in a linear system with unknown pressures
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.,
  \[ p_{i}^{l+1} = (1 - \omega) p_{i}^{l} + \omega \left( s_i - \sum_{j \neq i} a_{ij} p_j^l \right) \]

  - 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)
Implicit Incompressible SPH Interpretation

- **PPE**
  \[ \Delta t^2 \nabla^2 p_i = \rho_0 - \rho_i^* \]
  - density change
  - predicted density error
  - due to pressure accelerations

- **discretized PPE**
  \[ \mathbf{Ap} = \mathbf{s} \]
  \[ \Delta t^2 \sum_j m_j (a_i^p - a_j^p) \nabla W_{ij} = \rho_0 - \rho_i^* \]
  \[ a_i^p = - \sum_j m_j \left( \frac{p_i}{\rho_i^2} - \frac{p_j}{\rho_j^2} \right) \nabla W_{ij} \]
  \[ (\mathbf{Ap})_i \]

- \[ \Delta t \sum_j m_j (\Delta t a_i^p - \Delta t a_j^p) \nabla W_{ij} = \rho_0 - \rho_i^* \]
  - pressure accel. causes a velocity change \( \mathbf{v}^p \)
  - whose divergence causes a density change
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 \( \mathbf{A}_p = \mathbf{s} \)

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

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

\[ (\mathbf{A}_p)_f \]

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

\[ \mathbf{s}_f \]
Implicit Incompressible SPH Implementation with Boundary Handling

- **initialization**
  - **density** \( \rho_f = \sum_{f_f} m_{f_f} W_{f_f} \rho_{f_f} + \sum_{f_b} m_{f_b} W_{f_f} \)
  - **predicted velocity** \( v_f^* = v_f + \Delta t a^{nonp}_f \)
  - **source term** \( s_f = \rho_0 - \rho_f - \Delta t \sum_{f_f} m_{f_f} \left( v_f^* - v_{f_f}^* \right) \nabla W_{f_f} - \Delta t \sum_{f_b} m_{f_b} \ldots \)
  - **pressure** \( p_f^0 = 0 \)
  - **diagonal element of matrix** \( A \)

\[
\begin{align*}
  a_{ff} &= \Delta t^2 \sum_{f_f} m_{f_f} \left( - \sum_{f_f} \frac{m_{f_f}}{\rho_0^2} \nabla W_{f_f} - 2\gamma \sum_{f_b} \frac{m_{f_b}}{\rho_0^2} \nabla W_{f_f} \right) \nabla W_{f_f} \\
  &\quad + \Delta t^2 \sum_{f_f} m_{f_f} \left( \frac{m_{f_f}}{\rho_0^2} \nabla W_{f_f} \right) \nabla W_{f_f} \\
  &\quad + \Delta t^2 \sum_{f_b} m_{f_b} \left( - \sum_{f_f} \frac{m_{f_f}}{\rho_0^2} \nabla W_{f_f} - 2\gamma \sum_{f_b} \frac{m_{f_b}}{\rho_0^2} \nabla W_{f_f} \right) \nabla W_{f_f} \\
  &\quad + \Delta t^2 \sum_{f_b} m_{f_b} \left( \frac{m_{f_b}}{\rho_0^2} \nabla W_{f_f} \right) \nabla W_{f_f}
\end{align*}
\]
Implicit Incompressible SPH Implementation with Boundary Handling

- iteration \( l \)
  - first particle loop
    - predicted pressure acceleration
      \[
      (\mathbf{a}_f^p)^l = - \sum_{f_f} m_{f_f} \left( \frac{p_f^l}{\rho_f^2} + \frac{p_{f_f}^l}{\rho_{f_f}^2} \right) \nabla W_{f_f} - \gamma \sum_{f_b} m_{f_b} 2 \frac{p_f^l}{\rho_f^2} \nabla W_{f_b}
      \]
  - second particle loop
    - predicted density change due to pressure acceleration
      \[
      (\mathbf{A}p_f^l) = \Delta t^2 \sum_{f_f} m_{f_f} \left( (\mathbf{a}_f^p)^l - (\mathbf{a}_{f_f}^p)^l \right) \nabla W_{f_f} + \Delta t^2 \sum_{f_b} m_{f_b} (\mathbf{a}_f^p)^l \nabla W_{f_b}
      \]
    - pressure update
      \[
      p_f^{l+1} = \max \left( p_f^l + \omega \frac{s_f - (\mathbf{A}p_f^l)}{a_{f_f}}, 0 \right)
      \]
  - predicted density deviation per particle
    \[
    (\rho_f^{\text{error}})^l = (\mathbf{A}p_f^l)^l - s_f
    \]
Comparison with Iterative SESPH

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

<table>
<thead>
<tr>
<th>$\Delta t$ [s]</th>
<th>PCISPH</th>
<th>IISPH</th>
<th>PCISPH / IISPH</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>avg. iter.</td>
<td>pressure</td>
<td>overall</td>
</tr>
<tr>
<td>0.0005</td>
<td>4.3</td>
<td>540</td>
<td>1195</td>
</tr>
<tr>
<td>0.00067</td>
<td>7.2</td>
<td>647</td>
<td>1145</td>
</tr>
<tr>
<td>0.001</td>
<td>14.9</td>
<td>856</td>
<td>1187</td>
</tr>
<tr>
<td>0.0025</td>
<td>66.5</td>
<td>1495</td>
<td>1540</td>
</tr>
<tr>
<td>0.004</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0.005</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

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

- state equation SPH (SESPH)

- iterative SESPH

- incompressible SPH
Outline

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