Simulation in Computer Graphics

Particle-based Fluid Simulation

Matthias Teschner

Computer Science Department
University of Freiburg
Outline

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

- fluid is represented with a set of small moving fluid elements, i.e. particles
- particles have attributes, e.g. position $x_i$, velocity $v_i$, mass $m_i$, volume $V_i$, density $\rho_i$, pressure $p_i$

representation  

**typical visualization**
Exemplary Simulation

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

- particles / small fluid elements / sample positions $\mathbf{x}_i$ are advected with their velocity $\mathbf{v}_i$

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

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

- simulation approaches compute the velocity $\mathbf{v}_i$ over time $t$
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 $\mathbf{v}_i$ 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 efficiently handled by the advection of the particles / sample positions (in Eulerian approaches, sample positions are not advected with the flow)
Accelerations

- $-\frac{1}{\rho_i} \nabla p_i$: acceleration due to pressure differences
  - pressure acts in normal direction at the surface of the fluid element
  - preserves the fluid volume / density
  - small and preferably constant density deviations are important for high-quality simulation
  - typically the largest acceleration

- $\nu \nabla^2 v_i$: acceleration due to friction forces between particles with different velocities
  - friction forces act in tangential direction at the surface of the fluid element
  - $\nu \approx 10^{-6} \text{m}^2 \cdot \text{s}^{-1}$: larger friction is less realistic, but can improve the stability

- $\frac{F_{\text{other}}}{m_i}$: e.g., boundary handling, gravity
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
    \[ \frac{dx_i}{dt} = v_i \quad \text{and} \quad \frac{dv_i}{dt} = -\frac{1}{\rho_i} \nabla p_i + \nu \nabla^2 v_i + \frac{F_{other}}{m_i} \]
    \[-\frac{1}{\rho_i} \nabla p_i \quad \text{and} \quad \nu \nabla^2 v_i \quad \text{can be computed with SPH} \]
- 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 (not necessarily the particle distance or the size of the compact support of $W_{ij}$)
Kernel Function

- close to a Gaussian, but with compact support
  - support typically between $h$ and $3h$
- e.g. cubic spline (1D: $\alpha = \frac{1}{h}$ 2D: $\alpha = \frac{15}{7\pi h^2}$ 3D: $\alpha = \frac{3}{2\pi h^3}$)

$$W(q) = \alpha \begin{cases} 
  \frac{2}{3} - q^2 + \frac{1}{2}q^3 & 0 \leq q < 1 \\
  \frac{1}{6}(2 - q)^3 & 1 \leq q < 2 \\
  0 & q \geq 2
\end{cases}$$

- 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 is a trade-off between performance and interpolation accuracy
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**
  \[ \nabla A_i = \rho_i \sum_j m_j \left( \frac{A_i}{\rho_i^2} + \frac{A_j}{\rho_j^2} \right) \nabla W_{ij} \]
  \[ \nabla^2 A_i = 2 \sum_j \frac{m_j}{\rho_j} A_{ij} \frac{x_{ij} \cdot \nabla W_{ij}}{x_{ij} \cdot x_{ij} + 0.01h^2} \]
  \[ \nabla \cdot A_i = -\frac{1}{\rho_i} \sum_j m_j A_{ij} \nabla W_{ij} \]
  \[ A_{ij} = A_i - A_j \]
  \[ A_{ij} = A_i - A_j \]
  \[ x_{ij} = x_i - x_j \]
Kernel Derivatives

- **first derivative**
- \[ \nabla W_{ij} = \left( \frac{\partial W_{ij}}{\partial x_i, x}, \frac{\partial W_{ij}}{\partial x_i, y}, \frac{\partial W_{ij}}{\partial x_i, z} \right)^T \]

- \[ \nabla W_{ij} = \frac{\partial W(q)}{\partial q} \nabla q = \frac{\partial W(q)}{\partial q} \frac{x_{ij}}{\|x_{ij}\| h} \quad q = \frac{\|x_i - x_j\|}{h} \]

- e.g. cubic spline (1D: \( \alpha = \frac{1}{h} \) 2D: \( \alpha = \frac{15}{7\pi h^2} \) 3D: \( \alpha = \frac{3}{2\pi h^3} \))

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

- **second derivative**

- \[ \nabla^2 W_{ij} = \frac{\partial^2 W(q)}{\partial q^2} \frac{1}{h^2} + \frac{\partial W(q)}{\partial q} \frac{2}{\|x_{ij}\| h} \]
Density Computation

- explicit form
  \[ \rho_i = \sum_j \frac{m_j}{\rho_j} \rho_j W_{ij} = \sum_j m_j W_{ij} \]
  - 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
  - $F_{i,\text{pressure}} = -\frac{m_i}{\rho_i} \nabla p_i$
  - $F_{i,\text{viscosity}} = m_i \nu \nabla^2 v_i$
  - $F_{i,\text{other}} = m_i g$
  - $F_i(t) = F_{i,\text{pressure}} + F_{i,\text{viscosity}} + F_{i,\text{other}}$
- for all particle $i$ do
  - $v_i(t + \Delta t) = v_i(t) + \Delta t F_i(t)/m_i$
  - $x_i(t + \Delta t) = x_i(t) + \Delta t v_i(t + \Delta t)$
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**
  $a_{i, \text{pressure}} = -\frac{1}{\rho_i} \nabla p_i$
  $a_{i, \text{viscosity}} = \nu \nabla^2 v_i$
  $a_{i, \text{other}} = g$
  $a_i(t) = a_{i, \text{pressure}} + a_{i, \text{viscosity}} + a_{i, \text{other}}$

- **for all particle** $i$ **do**
  $v_{i}(t + \Delta t) = v_{i}(t) + \Delta t a_{i}(t)$
  $x_{i}(t + \Delta t) = x_{i}(t) + \Delta t v_{i}(t + \Delta t)$
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 $\lambda = 0.4$ and particle spacing $h$
  - motivation: for $\lambda = 1$, a particle moves less than its diameter per time step
Outline

- concept of an SPH fluid simulator
- momentum equation
- SPH basics
- neighborhood search
- incompressibility
- boundary handling
- surface reconstruction
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, e.g. gravity
- surface forces
  - based on shear and normal stress distribution on the surface due to deformation of the fluid element
  - normal stress related to volume changes
  - 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**

- force at particle $i$
  \[
  \mathbf{F}_{i}^{\text{pressure}} = - \left( \begin{array}{c}
  \frac{\partial p_i}{\partial x_{i,x}} \\
  \frac{\partial p_i}{\partial x_{i,y}} \\
  \frac{\partial p_i}{\partial x_{i,z}} 
  \end{array} \right) \quad V_i = -\nabla p_i \quad V_i = -\frac{m_i}{\rho_i} \nabla p_i
  \]

- respective acceleration
  \[
  \mathbf{a}_{i}^{\text{pressure}} = \frac{\mathbf{F}_{i}^{\text{pressure}}}{m_i} = -\frac{1}{\rho_i} \nabla p_i
  \]
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 \( \tau = \nu \begin{pmatrix} \frac{\partial u}{\partial x} + \frac{\partial u}{\partial x} & \frac{\partial v}{\partial x} + \frac{\partial u}{\partial y} & \frac{\partial w}{\partial x} + \frac{\partial u}{\partial z} \\ \frac{\partial v}{\partial x} + \frac{\partial v}{\partial y} & \frac{\partial v}{\partial y} + \frac{\partial v}{\partial y} & \frac{\partial w}{\partial y} + \frac{\partial v}{\partial z} \\ \frac{\partial w}{\partial x} + \frac{\partial w}{\partial z} & \frac{\partial w}{\partial y} + \frac{\partial w}{\partial y} & \frac{\partial w}{\partial z} + \frac{\partial w}{\partial z} \end{pmatrix} \)

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

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

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

- 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
  \[ \int_\Omega W(x - x', h)dx' = 1 \]
- 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
- should be differentiable
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

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

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

\[ W \text{ is symmetric} \]

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

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

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

Gauss theorem

\[ S \text{ is the surface of } \Omega \]

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

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

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

\[ \nabla_x A(x_i) \approx \sum_j \frac{m(x_j)}{\rho(x_j)} A(x_j) \nabla W(x_i - x_j, h) \]
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 pressure forces do not preserve momentum and are not necessarily zero for constant pressure
Gradient

- momentum-preserving form

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

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

- SPH approximation

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

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

- resulting pressure forces preserve linear and angular momentum for arbitrary samplings
Laplacian

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

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

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

$$x_{ij} = x_i - x_j$$
Outline

- concept of an SPH fluid simulator
- momentum equation
- SPH basics
- neighborhood search
- incompressibility
- boundary handling
- 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, 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 $O(1)$
- Verlet lists
  - 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
Characteristics

- 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
- incompressibility
- boundary handling
- surface reconstruction
Basic Grid

- particle is stored in a cell with coordinates \((k, l, m)\)
- 27 cells are queried in the neighborhood search \((k\pm1, l\pm1, m\pm1)\)
- 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
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
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
- incompressibility
- boundary handling
- surface reconstruction
Spatial Hashing

- hash function maps a grid cell to a hash cell
  - infinite domain is mapped to a finite list
  - in contrast to index sort, infinite domains can be handled
- 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
- incompressibility
- boundary handling
- 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)
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
Outline

- concept of an SPH fluid simulator
- momentum equation
- SPH basics
- neighborhood search
- incompressibility
- boundary handling
- 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.
  - \( p_i = k (\rho_i - \rho_0) \)
  - \( p_i = k \rho_i \)
  - \( p_i = k \left( \frac{\rho_i}{\rho_0} - 1 \right) \)
  - \( k \) is user-defined
  - in graphics referred to as compressible SPH
  - \( p_i = k_1 \left( \left( \frac{\rho_i}{\rho_0} \right)^{k_2} - 1 \right) \)
  - \( k_1, k_2 \) are user-defined
  - in graphics referred to as weakly compressible SPH
- compressibility is governed by the stiffness constant(s) and limits the time step
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 $F_{i}^{\text{nonp}}(t)$
- compute intermediate velocity $v_{i}^{*} = v_{i}(t) + \Delta t \frac{F_{i}^{\text{nonp}}}{m_{i}}$
- 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**
  - \(\mathbf{F}_{i}^{\text{viscosity}} = m_i \nu \nabla^2 \mathbf{v}_i\)
  - \(\mathbf{F}_{i}^{\text{other}} = m_i \mathbf{g}\)
  - \(\mathbf{v}_i^* = \mathbf{v}_i(t) + \Delta t \frac{\mathbf{F}_{i}^{\text{viscosity}} + \mathbf{F}_{i}^{\text{other}}}{m_i}\)

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

- **for all particle** \(i\) **do**
  - \(\mathbf{F}_{i}^{\text{pressure}} = -\frac{m_i}{\rho_i^*} \nabla p_i\)

- **for all particle** \(i\) **do**
  - \(\mathbf{v}_i(t + \Delta t) = \mathbf{v}_i^* + \Delta t \frac{\mathbf{F}_{i}^{\text{pressure}}}{m_i}\)
  - \(\mathbf{x}_i(t + \Delta t) = \mathbf{x}_i(t) + \Delta t \mathbf{v}_i(t + \Delta t)\)

- follows from the continuity equation
- avoids neighbor search
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
  $F_{i}^{\text{viscosity}} = m_i \nu \nabla^2 v_i$
  $F_{i}^{\text{other}} = m_i g$
  $v_i^* = v_i(t) + \Delta t \frac{F_{i}^{\text{viscosity}} + F_{i}^{\text{other}}}{m_i}$
  $x_i^* = x_i(t) + \Delta t v_i^*$

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 $\rho_{err}$, e.g. average or maximum
  for all particle $i$ do
    $F_{i}^{\text{pressure}} = -\frac{m_i}{\rho_i^*} \nabla p_i$
    $v_i^* = v_i^* + \Delta t \frac{F_{i}^{\text{pressure}}}{m_i}$
    $x_i^* = x_i^* + \Delta t^2 \frac{F_{i}^{\text{pressure}}}{m_i}$

until $\rho_{err} < \eta$  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)
    \[
    \Delta x_i = -\frac{1}{\rho_0} \sum_j \left( \frac{p_i}{\beta_i} + \frac{p_j}{\beta_j} \right) \nabla W_{ij} \quad \beta \text{ is a pre-computed constant}
    \]
- different EOS and stiffness constants are used
  - \( k = \frac{\rho^*_i r_i^2}{2\rho_0 \Delta t^2} \) in local Poisson SPH
  - \( k = \frac{m_i^2 \Delta t^2 \sum_j \nabla W^0_{ij} \cdot \sum_j \nabla W^0_{ij} + \sum_j (\nabla W^0_{ij} \cdot \nabla W^0_{ij})}{2\rho_0^2} \) in PCISPH \( W^0 \) is precomputed
  - \( k = 1 \) in PBF \( (p_i = \frac{\rho_i}{\rho_0} - 1) \)
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 leads 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:
  \[ \nabla^2 p_i = \frac{\rho_0}{\Delta t} \nabla \cdot \mathbf{v}_i^* = \frac{1}{\Delta t^2} (\rho_0 - \rho_i^*) \]
- \( \mathbf{v}_i^* \) 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

- 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
  \[ \frac{d\rho_i(t)}{dt} = -\rho_i(t) \nabla \cdot \mathbf{v}_i(t) \]
to
  \[ \rho_i(t+\Delta t) - \rho_i(t) = \sum_j m_j \mathbf{v}_{ij}(t + \Delta t) \nabla W_{ij}(t) \]
  forward difference

- unknown velocity \( \mathbf{v}_i(t + \Delta t) \) can be rewritten
  using known non-pressure forces \( \mathbf{F}_{i}^{nonp}(t) \) and
  unknown pressure forces \( \mathbf{F}_{i}^{p}(t) \)

  \[ \mathbf{v}_i(t + \Delta t) = \mathbf{v}_i(t) + \Delta t \frac{\mathbf{F}_{i}^{nonp}(t) + \mathbf{F}_{i}^{p}(t)}{m_i} \]

  \[ \mathbf{v}_{i}^{nonp} = \mathbf{v}_i(t) + \Delta t \frac{\mathbf{F}_{i}^{nonp}(t)}{m_i} \]
  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 $\rho_{i,\text{nonp}} = \rho_i(t) + \Delta t \sum_j m_j \mathbf{v}_{ij}^{\text{nonp}} \nabla W_{ij}(t)$

predicted density, if only non-pressure forces are applied

the discretized continuity equation can be written as

$$\frac{\rho_0 - \rho_{i,\text{nonp}}(t)}{\Delta t} = \Delta t \sum_j m_j \left( \frac{\mathbf{F}_{i,\text{p}}^p(t)}{m_i} - \frac{\mathbf{F}_{j,\text{p}}^p(t)}{m_j} \right) \nabla W_{ij}(t)$$

with unknown pressure forces $\mathbf{F}_{i,\text{p}}^p(t)$

- unknown pressure forces $\mathbf{F}_{i,\text{p}}^p(t)$ can be rewritten using unknown pressures $p_i(t)$

$$\mathbf{F}_{i,\text{p}}^p(t) = -m_i \sum_j m_j \left( \frac{p_i(t)}{\rho_i^2(t)} - \frac{p_j(t)}{\rho_j^2(t)} \right) \nabla W_{ij}(t)$$

resulting in a linear system with unknown pressures
Implicit Incompressible SPH
Linear System

- one equation per particle
  \[ \sum_j a_{ij} p_j = b_i = \rho_0 - \rho_i^{nonp} \]
- iterative solver, e.g.,
  \[ p_i^{l+1} = (1 - \omega) p_i^l + \omega \frac{\rho_0 - \rho_i^{nonp} - \sum_{j \neq i} a_{ij} p_j^l}{a_{ii}} \]
  - relaxed Jacobi
  - matrix-free implementation
  - user-defined \( \omega (= 0.5) \)
  - \( p_i^l \) is the computed pressure in iteration \( l \)
  - \( p_i^0 \) is initialized, e.g., \( p_i^0 = 0 \) or \( p_i^0 = p_i(t - \Delta t) \)
  - system is not necessarily symmetric
    (a prerequisite for Conjugate Gradient)
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></th>
<th></th>
<th>IISPH</th>
<th></th>
<th></th>
<th>PCISPH / IISPH</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>avg. iter.</td>
<td>total comp. time [s]</td>
<td>pressure</td>
<td>overall</td>
<td>avg. iter.</td>
<td>total comp. time [s]</td>
<td>pressure</td>
<td>overall</td>
</tr>
<tr>
<td>0.0005</td>
<td>4.3</td>
<td>540</td>
<td>1195</td>
<td></td>
<td>2.2</td>
<td>148</td>
<td>978</td>
<td></td>
</tr>
<tr>
<td>0.00067</td>
<td>7.2</td>
<td>647</td>
<td>1145</td>
<td></td>
<td>2.9</td>
<td>149</td>
<td>753</td>
<td></td>
</tr>
<tr>
<td>0.001</td>
<td>14.9</td>
<td>856</td>
<td>1187</td>
<td></td>
<td>4.9</td>
<td>164</td>
<td>576</td>
<td></td>
</tr>
<tr>
<td>0.0025</td>
<td>66.5</td>
<td>1495</td>
<td>1540</td>
<td></td>
<td>18.4</td>
<td>242</td>
<td>410</td>
<td></td>
</tr>
<tr>
<td>0.004</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>33.5</td>
<td>273</td>
<td>379</td>
<td></td>
</tr>
<tr>
<td>0.005</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>45.8</td>
<td>297</td>
<td>383</td>
<td></td>
</tr>
</tbody>
</table>

- largest possible time step does not necessarily results 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
- incompressibility
- boundary handling
- surface reconstruction
Concept

- Rigid objects are sampled with particles
  - Sufficiently dense
  - Oversampling is not an issue
- Adapted density computation of fluid particles that incorporates contributions from a rigid
- Compute pressure forces between fluid and rigid properties
  - Works with arbitrary samplings
  - Volumetric objects, planes, lines, even one particle
  - Preserves a smooth pressure field
Boundary Forces

- **rigid particle volume**
  \[ V_{bi} = \frac{m_b}{\rho_{bi}} = \frac{m_b}{\sum_k m_b W_{ik}} = \frac{1}{\sum_k W_{ik}} \]
  volume estimation is independent from mass and density

- **adapted fluid density**
  \[ \rho_{fi} = \sum_j m_{fi} W_{ij} + \sum_k \rho_0 V_{bk} W_{ik} \]
  rigid particles are considered as fluid particles in the fluid density computation

- **rigid-fluid pressure forces**
  \[ F_{fi \leftarrow bj}^p = -F_{bj \leftarrow fi}^p = -m_{fi} \rho_0 V_{bj} \left( \frac{p_{fi}}{\rho_{fi}^2} \right) \nabla W_{ij} \]
  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
- incompressibility
- boundary handling
- surface reconstruction