Computer Animation and Visualisation

Lecture 9

Physics-based Animation (Part 2)

Floyd M. Chitalu

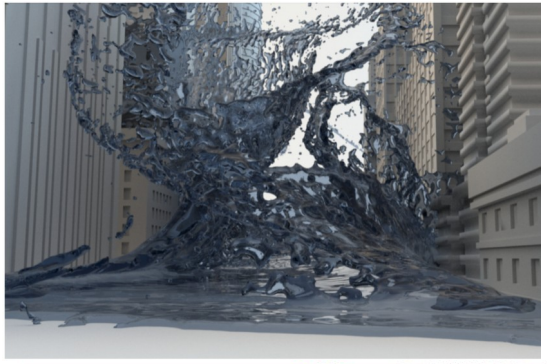
Lecture Overview

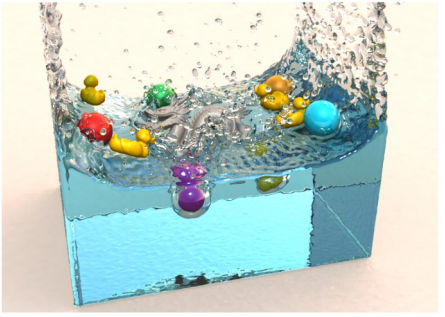
- Modelling Liquid
- Spatial Discretization
- Temporal Discretization

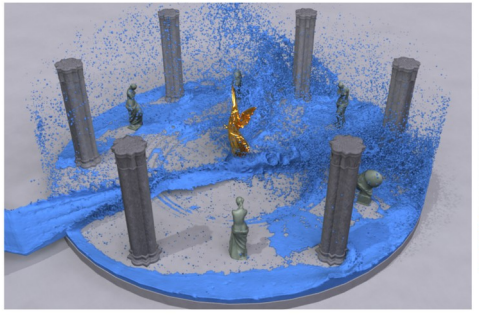
Lecture Overview

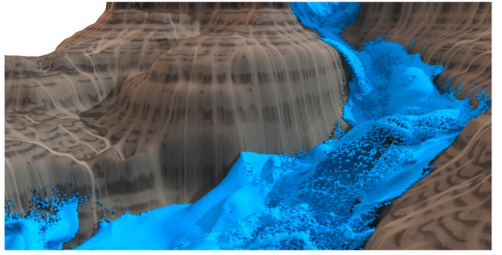
- Modelling Liquid
- Spatial Discretization
- Temporal Discretization

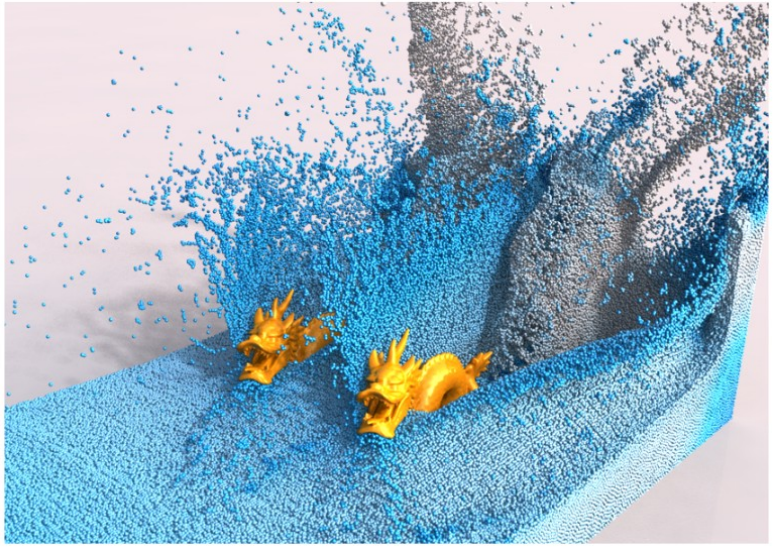
Modelling Liquid

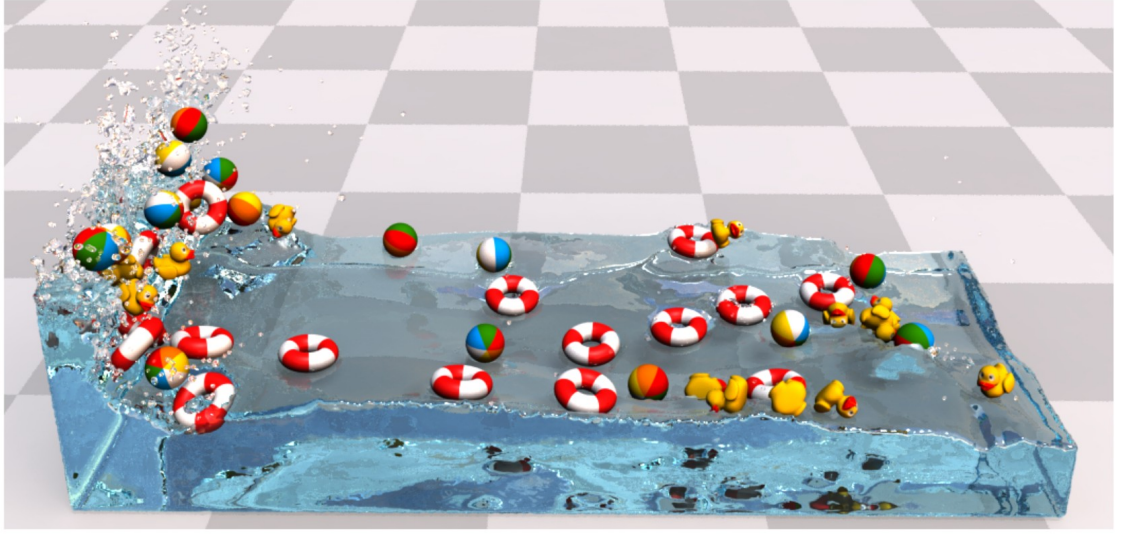












Modelling Liquid

- Two approaches to simulating liquids
  1) Lagrangian (*Today*)
     - Particles carry data samples and travel with the flow
     - i.e. Smoothed Particle Hydrodynamics (SPH)
  2) Eulerian
     - Samples are fixed in a grid, and information flows past.
     - i.e. Grid-based view of fluid motion

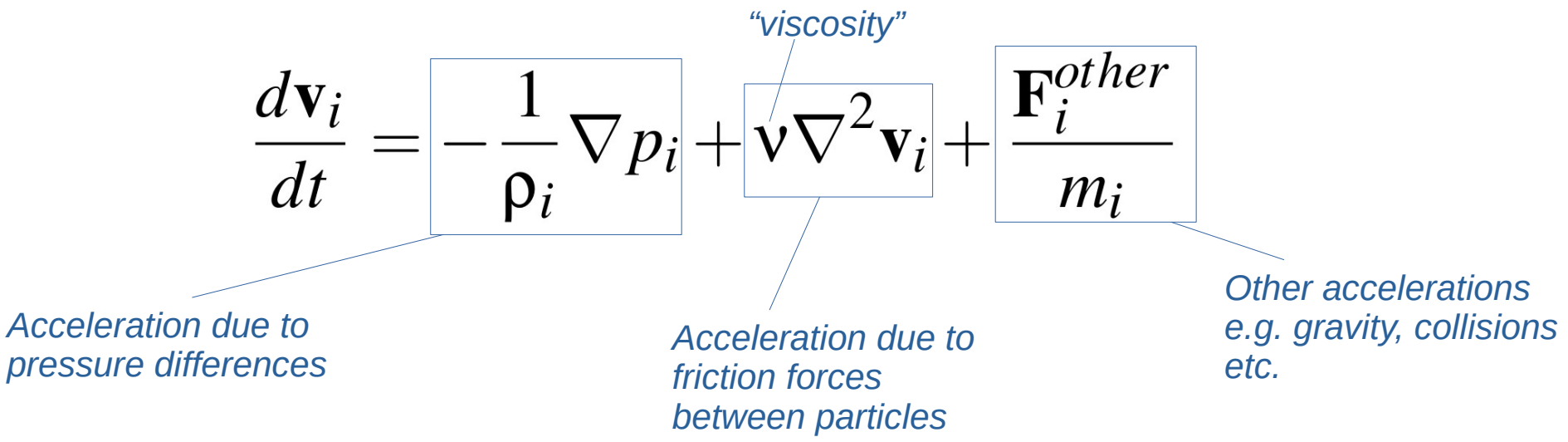
Governing Equations

- We consider a fluid that consists of a set of small moving fluid elements (particles)
  - Each particle *i* has a mass m_i and carries attributes such as density ρ_i , pressure p_i or volume V_i
  - Over time *t*, particle positions $\mathbf{x}_i$ and the respective attributes are advected with the local fluid velocity $\mathbf{v}_i$

$$\frac{d\mathbf{x}_i}{dt} = \mathbf{v}_i.$$

Governing Equations (Navier-Stokes equation)

- The time rate of change of the velocity is governed by the *Navier-Stokes equation (Lagrange form)*



Smoothed Particle Hydrodynamics (SPH)

- The SPH concept is used for two purposes
  1) To *interpolate* fluid quantities at arbitrary positions.
  2) To *approximate* the spatial derivatives in the Navier-stokes equation.

SPH (interpolation)

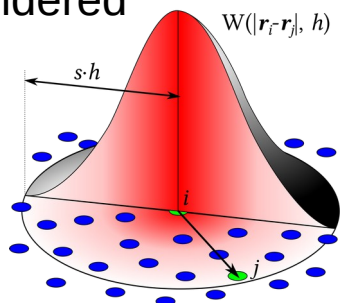
- A quantity A_i at an arbitrary position $\boldsymbol{x}_i$ is approximately computed with a set of known quantities A_j at neighboring particle positions $\mathbf{x}_j$

$$A_i = \sum_j \frac{m_j}{\rho_j} A_j W_{ij} \qquad W_{ij} = W\left(\frac{\|\mathbf{x}_i - \mathbf{x}_j\|}{h}\right) = W(q) = \frac{1}{h^d} f(q)$$

Smoothing kernel — *Smoothing length*

$$f(q) = \frac{3}{2\pi} \begin{cases} \frac{2}{3} - q^2 + \frac{1}{2}q^3 & 0 \le q < 1 \\ \frac{1}{6}(2-q)^3 & 1 \le q < 2 \\ 0 & q \ge 2 \end{cases}$$

- The number of adjacent particles that are considered is dependant on three factors
  - 1) The dimensionality *d*; *2)* the support of the kernel function; and 3) the particle spacing which is typically close to *h*
- Three factors influencing the accuracy of the summation
  - 1) Choice of the kernel function; 2) The number and 3) The disorder of considered particles



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

SPH (Spatial derivatives)

- But how do we solve for variables like:

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

- Common solution

Used to compute accelerations

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

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

Used to predict density changes

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

  - … with:

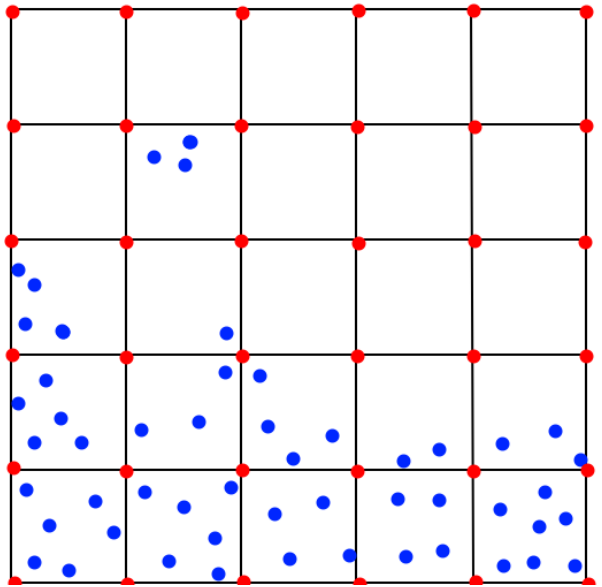
$$A_{ij} = A_i - A_j, \quad \mathbf{A}_{ij} = \mathbf{A}_i - \mathbf{A}_j, \quad \mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j \qquad \nabla W_{ij} = \left(\frac{\partial W_{ij}}{\partial x_{i,x}}, \frac{\partial W_{ij}}{\partial x_{i,y}}, \frac{\partial W_{ij}}{\partial x_{i,z}} \right)^T$$

SPH Fluid Solver

- Three basic building blocks of SPH-based fluid solvers
  - Neighborhood search
  - Pressure computation
  - Time integration

SPH Fluid Solver (Neighborhood search)

- The neighborhood search is typically accelerated by a spatial access structure, e.g. a uniform grid
  - cell size is normally equal to the kernel support, e.g., $2h$



SPH Fluid Solver (Pressure computation)

- Preasure is computed from density to compute the preasure gradient (in NS eq.),

Stiffness constant to scale preasure (forces)

$$p_i = k\left(\left(\frac{\rho_i}{\rho_0}\right)^7 - 1\right)$$

Desired rest density of the fluid

- In practice, a larger stiffness constant reduces the compressibility of the fluid, but demands smaller integration time steps.

SPH Fluid Solver (Algorithm)

Algorithm 1 SPH with state equation.

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

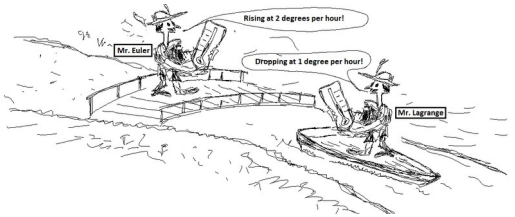
Lecture Overview

- Modelling Liquid
- Spatial Discretization
- Temporal Discretization

Spatial Discretization

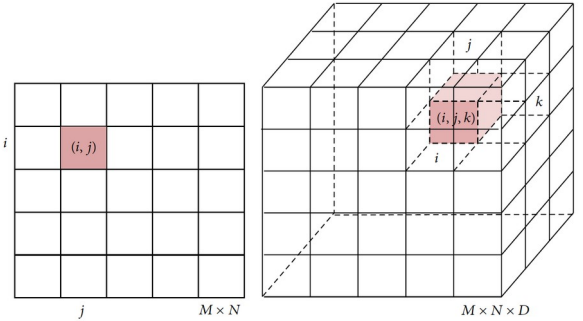
- Lagrangian and Eulerian reference frames
- Spatial data structures
- Discretizing equations of motion

Lagrangian vs Eulerian view



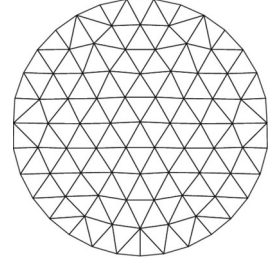
- Lagrangian
  - The reference frame moves with the material
  - Often used for solids and employ tetrahedral meshes or particles and finite element methods.
    - Easier to construct the mapping (**x**(**u**)) from rest space to world space if we explicitly track points in the material through time.
- Eulerian
  - The point of measurement, the reference frame, is fixed in space
  - Eulerian reference frames are often used for fluids and often employ regular grids
    - No need for a `mapping`, and fixed reference frames offer many computational advantages

Grids

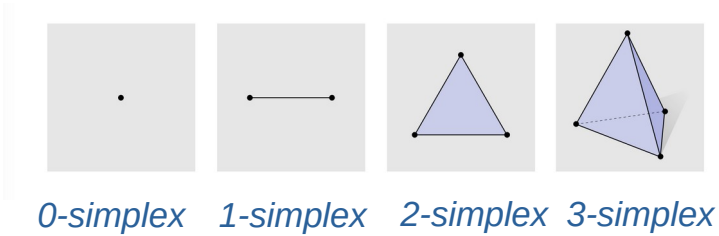


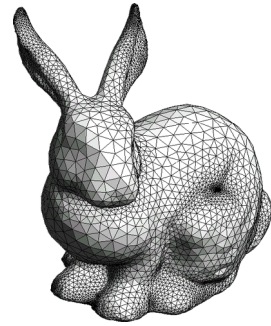
- The regular grid
  - All edges have the same length, called the grid spacing
  - Cubes in the grid are referred to as cells which have 8 vertices, 12 edges, and 6 faces
  - Can be described by a few redundant parameters:
    - The grid spacing, the grid resolution (i.e. the number of cells in each dimension), and the upper and lower extent of the grid.
- Grids are typically fixed in space and do not change shape, thus typically an Eulerian frame is adopted.

Meshes

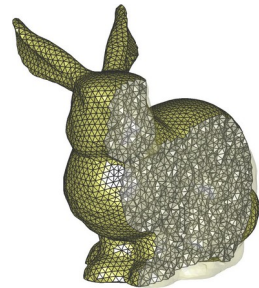


- A mesh is a `simplicial complex`
  - A simplicial complex is decomposes a domain into a set of disjoint ‘simplices’, e.g. triangles in 2D and tetrahedra in 3D
  - A k-simplex contains k + 1 vertices that are all connected.



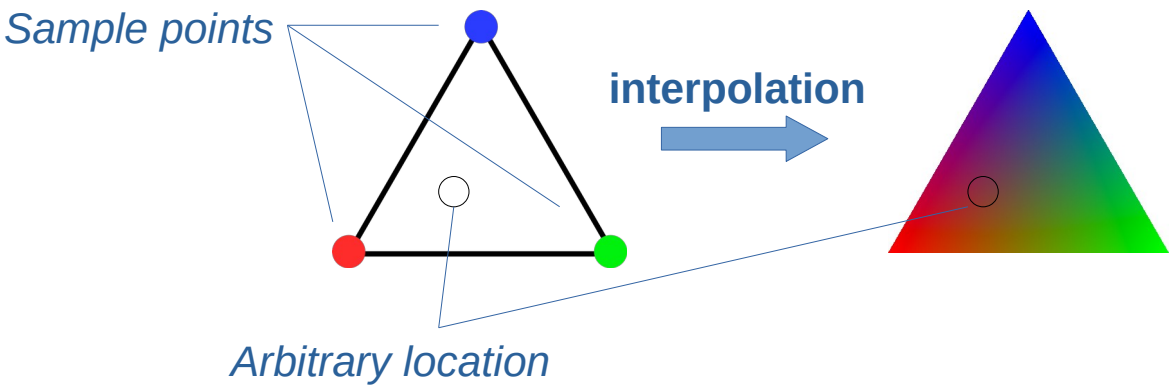


  - Useful if Lagrangian reference frame is adopted.
  - Automatic (tetrahedral) meshing is a hard problem!

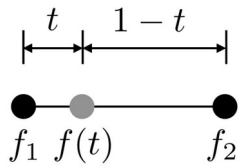


Interpolation

- Sometimes the value of the field is required at a location other than the sample points.

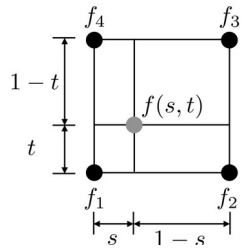


Interpolation



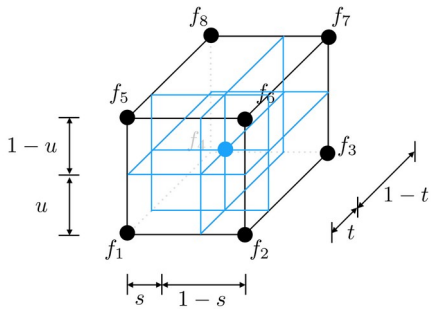
Linear

$$f(t) = (1-t)f_1 + tf_2$$



Bilinear

$$f(s,t) = (1-s)(1-t)f_1 + s(1-t)f_2 + stf_3 + (1-s)tf_4$$

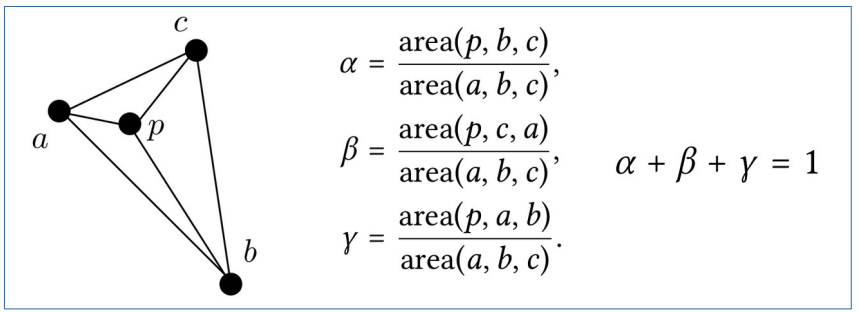


Trilinear

$$\begin{aligned} f(s,t,u) =& (1-s)(1-t)(1-u)f_1 + s(1-t)(1-u)f_2 + st(1-u)f_3 + (1-s)t(1-u)f_4 \\ &+ (1-s)(1-t)uf_5 + s(1-t)uf_6 + stuf_7 + (1-s)tuf_8. \end{aligned}$$

Interpolation (barycentric)

- Triangles (and more generally simplicial complexes) are also used frequently in physics-based animation
- To interpolate to a point **p**, we associate with the vertices a, b, and c the weights α, β, and γ:



$$\alpha = \frac{\text{area}(p,b,c)}{\text{area}(a,b,c)}, \quad \beta = \frac{\text{area}(p,c,a)}{\text{area}(a,b,c)}, \quad \gamma = \frac{\text{area}(p,a,b)}{\text{area}(a,b,c)}. \qquad \alpha+\beta+\gamma = 1$$

Interpolated values

$$f_p = f(\alpha,\beta,\gamma) = \alpha f_a + \beta f_b + \gamma f_c$$

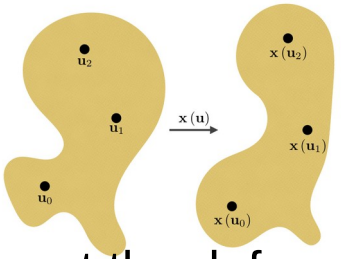
- For more efficient code, we can also write the following, which is useful for methods like FEM

$$f_p = f(\alpha,\beta) = (1-\alpha-\beta)f_a + \alpha f_b + \beta f_c = f_a + \alpha(f_b - f_a) + \beta(f_c - f_a)$$

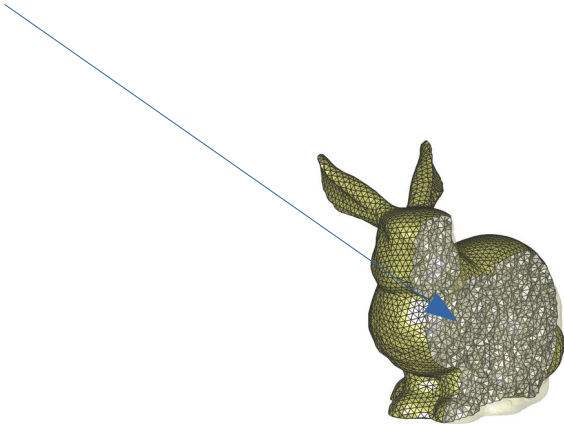
Finite Element Method

Finite Element Method

- For animating elastic bodies we must compute *the deformation gradient* **F** from the deformation function **x**(**u**)
  - $$\mathbf{x}(\mathbf{u}) = \mathbf{x}(\mathbf{u}_0) + \mathbf{A}\,(\mathbf{u} - \mathbf{u}_0)$$



- To compute **F**, we use *a piecewise linear basis* to represent the deformation function. Two prerequsites:
  - Breaking an object up into a finite set of disjoint elements.
  - Defining the basis (shape) functions over these elements.
- **F** is constant over an element.
  - And so will be stress and strain.



$$f_p = f(\alpha, \beta) = (1 - \alpha - \beta)f_a + \alpha f_b + \beta f_c = f_a + \alpha(f_b - f_a) + \beta(f_c - f_a)$$

Finite Element Method

- Recall that an arbitrary point inside a triangle can be represented with barycentric coordinates:

material/rest/undeformed space

$$\mathbf{u} = \mathbf{u}_0 + \alpha(\mathbf{u}_1 - \mathbf{u}_0) + \beta(\mathbf{u}_2 - \mathbf{u}_0)$$

$$\mathbf{u} = \mathbf{u}_0 + \begin{pmatrix}\mathbf{u}_{10} & \mathbf{u}_{20}\end{pmatrix}\begin{pmatrix}\alpha \\ \beta\end{pmatrix}$$

world/deformed space

$$\mathbf{x} = \mathbf{x}_0 + \alpha(\mathbf{x}_1 - \mathbf{x}_0) + \beta(\mathbf{x}_2 - \mathbf{x}_0)$$

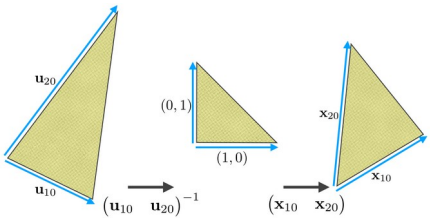
$$\mathbf{x} = \mathbf{x}_0 + \begin{pmatrix}\mathbf{x}_{10} & \mathbf{x}_{20}\end{pmatrix}\begin{pmatrix}\alpha \\ \beta\end{pmatrix}$$

- Thus, we can write the deformation function as

$$\mathbf{x}(\mathbf{u}) = \mathbf{x}_0 + \begin{pmatrix}\mathbf{x}_{10} & \mathbf{x}_{20}\end{pmatrix}\begin{pmatrix}\mathbf{u}_{10} & \mathbf{u}_{20}\end{pmatrix}^{-1}(\mathbf{u} - \mathbf{u}_0)$$

- The gradient of this function w.r.t $\mathbf{u}$, is the deformation gradient:

$$\mathbf{F} = \frac{\partial \mathbf{x}}{\partial \mathbf{u}} = \begin{pmatrix}\mathbf{x}_{10} & \mathbf{x}_{20}\end{pmatrix}\begin{pmatrix}\mathbf{u}_{10} & \mathbf{u}_{20}\end{pmatrix}^{-1}$$



Finite Element Method

- Finally the nodal forces are computed by

$$\mathbf{f} = \boldsymbol{\sigma}\mathbf{n}_i$$

- $\mathbf{n}_i$ is the area weighted normal of the edge (2D) or face (3D) opposite the node the force is acting upon in the rest configuration.
  - This force is evenly distributed among the vertices of the edge (2D) or face (3D).

Finite Element Method (Example Algorithm)

```
1:  for Particle p : particles do
2:      p.frc = 0
3:      p.frc += p.mass*gravity
4:  end for
5:  for Element e : elements do
6:      Matrix3x3 F = Matrix3x3 (x1-x0, x2-x0, x3-x0) * inverse(Matrix3x3(u1-u0, u2-u0, u3-u0))
7:      PolarDecomp (F, Q, Ftilde)
8:      Matrix3x3 strain = 1/2 * (Ftilde + transpose (Ftilde)) - I
9:      Matrix3x3 stress = lambda * trace(strain) * I + 2 * mu * strain
10:     for i = 0 to 3 do
11:         particles[e.node[i]].frc += Q * stress * e.normal[i]
12:     end for
13: end for
14: for Particle p : particles do
15:     p.vel += dt*(p.frc / p.mass)
16:     p.pos += dt*(p.vel)
17: end for
```

Lecture Overview

- Modelling Liquid
- Spatial Discretization
- Temporal Discretization

Temporal Discretization

- Explicit Integration
- Implict Integration

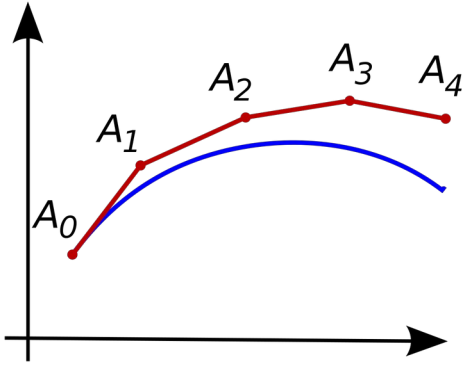
Explicit Integration

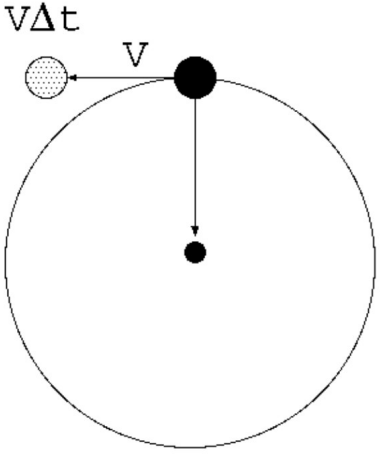
- Explicit integration refers to integration techniques where the updated state at time t + Δt is described solely in terms of quantities computed at time t.
- Explicit Euler

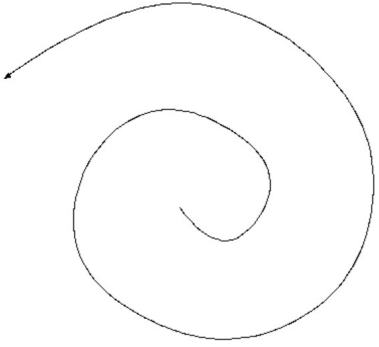
$$\mathbf{x}_p(t + \Delta t) = \mathbf{x}_p(t) + \Delta t \cdot \mathbf{v}(\mathbf{x}_p, t)$$

Problem with Euler Method

- The system tends to blow up (diverge) very quickly when the time step is too large.
  - Can be mitigated with smaller timestep but there is a higher computational cost!

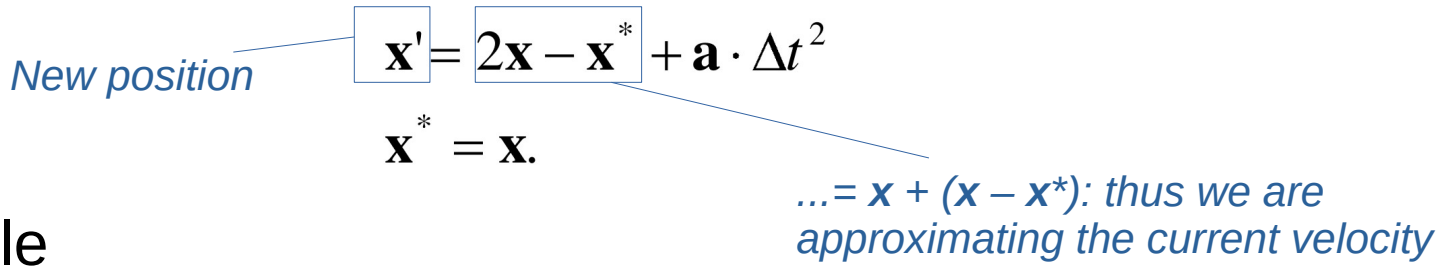






Verlet Integration

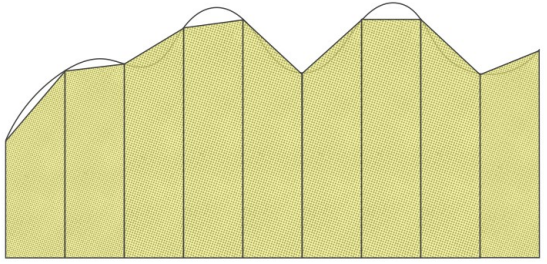
- Instead of storing each particle's position and velocity, store its current position **x** and its previous position **x***

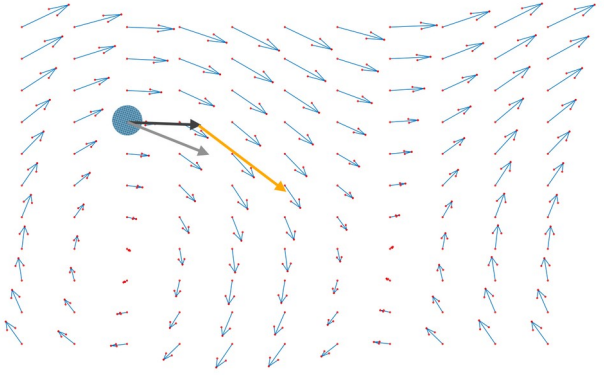


- Very stable
  - Velocity is implicitely given, thus velocity and position do not come out of sync
- It's fast

Trapezoidal Rule

- Evaluate the velocity at the particle's position, pretend to move the particle a full timestep and evaluate the velocity again, then use the average of these two evaluations to update the particle's position

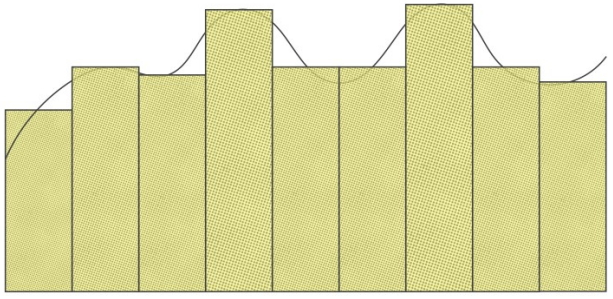


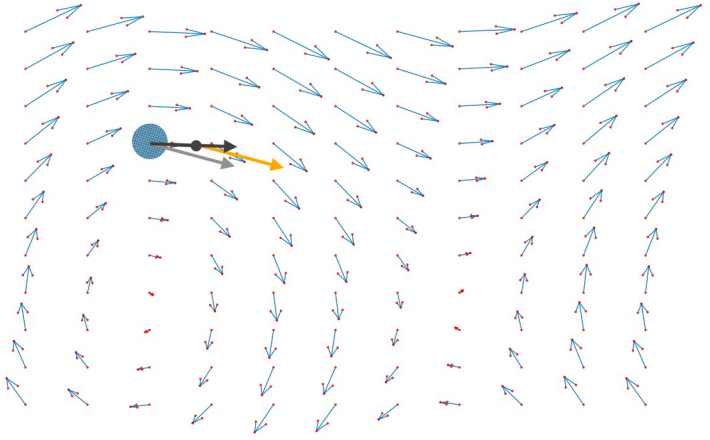


$$\mathbf{x}(t+\Delta t) = \mathbf{x}(t) + \frac{\Delta t}{2}\left(\mathbf{v}(\mathbf{x}(t), t) + \mathbf{v}(\mathbf{x} + \Delta t \mathbf{v}(\mathbf{x}, t), t)\right)$$

Midpoint Method

- Evaluate the velocity at the particle's position, pretend to move the particle a half step and evaluate the velocity again, then use the second evaluation to update the particle's position

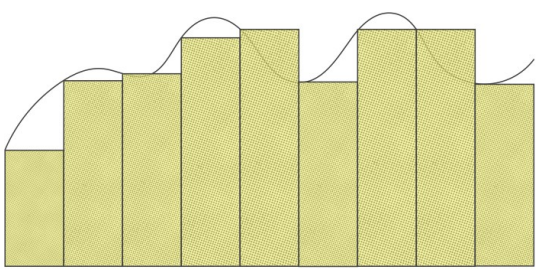




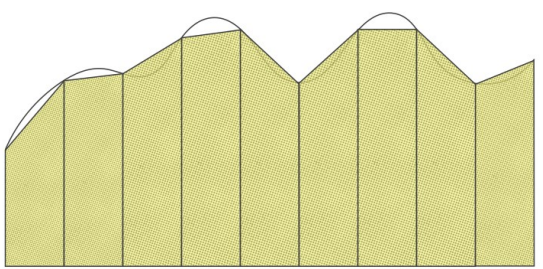
$$\mathbf{x}(t+\Delta t) = \mathbf{x}(t) + \Delta t \left(\mathbf{v}(\mathbf{x} + \frac{\Delta t}{2}\mathbf{v}(\mathbf{x}, t), t) \right)$$

Trapezoidal Rule vs. Midpoint Method

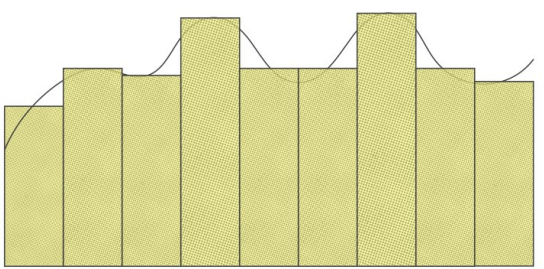
- The trapezoidal rule tends to produce a smoother solution since it averages to velocity estimates.
- The midpoint method is more susceptible to noise or aliasing



Explicit Euler



Trapezoidal Rule



Midpoint method

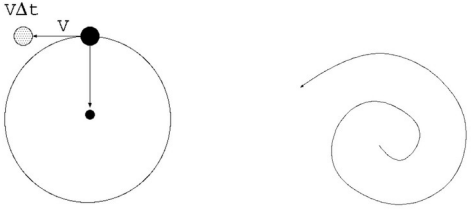
Symplectic Euler (revisited)

$$\mathbf{x}_p(t+\Delta t) = \mathbf{x}_p(t) + dt \cdot \mathbf{v}_p(t)$$ *"Primitive euler"*

$$\mathbf{x}_p(t+\Delta t) = \mathbf{x}_p(t) + \frac{dt}{2} \cdot \left(\mathbf{v}_p(t) + \mathbf{v}_p(t+\Delta t)\right)$$ *"Improved euler"*

$$\mathbf{x}_p(t+\Delta t) = \mathbf{x}_p(t) + dt \cdot \mathbf{v}_p(t+\Delta t).$$ *"Symplectic euler"*

- The only difference is where the velocity is evaluated
  - at the beginning of the step, end of the step, or an average of both.
- For some problems (e.g. pure elasticity), "improved Euler" may not converge at all.
  - It is *unconditionally unstable* as the solution can diverge irrespective of timestep size.
- Symplectic Euler is the preferred explicit integrator
  - despite its lower order accuracy



Implicit Integration

- Sometimes we wish to solve ‘stiff’ problems
  - materials that have very strong resistance to deformation
- Explicit integrators require smaller and smaller timesteps in order to remain stable (which is not guarranteed!).

Implicit Integration

- Explicit symplectic Euler integrator

$$\mathbf{v}(t + \Delta t) = \mathbf{v}(t) + \Delta t \cdot \mathbf{M}^{-1}\mathbf{f}(\mathbf{x}(t), t)$$
$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \Delta t \cdot \mathbf{v}(t + \Delta t).$$

- The implicit formulation

$$\mathbf{v}(t + \Delta t) = \mathbf{v}(t) + \Delta t \cdot \mathbf{M}^{-1}\mathbf{f}(\mathbf{x}(t + \Delta t), t + \Delta t)$$
$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \Delta t \cdot \mathbf{v}(t + \Delta t).$$

- Since we can’t directly compute the forces at t+Δt without knowing the system state at t+Δt the new state is defined *implicitly* as one which satisfies this equation.

Linearly Implicit Euler

(1) $$\mathbf{K}(\mathbf{x} - \mathbf{x}_0) + \mathbf{D}(\dot{\mathbf{x}}) + \mathbf{M}\ddot{\mathbf{x}} = \mathbf{f}_{ext}$$

Non-linear dynamic system for soft bodies

(2) $$\mathbf{K}\mathbf{x} - \mathbf{K}\mathbf{x}_0 + \mathbf{D}\dot{\mathbf{x}} + \mathbf{M}\ddot{\mathbf{x}} = \mathbf{f}_{ext}.$$

*Linearize system around current state **x***

Our forces $$\mathbf{f}_{ext} - \mathbf{K}\mathbf{x} + \mathbf{K}\mathbf{x}_0 - \mathbf{D}\dot{\mathbf{x}}.$$ (3)

*Substitute forces and eq. of **x**(t + Δt) into eq. of **v**(t + Δt)*

(4)
$$\mathbf{v}(t + \Delta t) = \mathbf{v}(t) + \Delta t \cdot \mathbf{M}^{-1}\mathbf{f}(\mathbf{x}(t + \Delta t), t + \Delta t)$$
$$\mathbf{v}(t + \Delta t) = \mathbf{v}(t) + \Delta t \cdot \mathbf{M}^{-1}\left[-\mathbf{K}\left(\mathbf{x}(t) + \Delta t\mathbf{v}(t + \Delta t)\right) + \mathbf{K}\mathbf{x}_0 - \mathbf{D}\mathbf{v}(t + \Delta t) + \mathbf{f}_{ext}\right]$$

$$\left(\mathbf{M} + \Delta t^2\mathbf{K} + \Delta t\mathbf{D}\right)\mathbf{v}(t + \Delta t) = \mathbf{M}\mathbf{v}(t) + \Delta t\left(-\mathbf{K}\left(\mathbf{x}(t) - \mathbf{x}_0\right) + \mathbf{f}_{ext}\right)$$ (5)

*The linear system of the form Ax=b which we solve to find **v**(t + Δt)*

Summary

- Modelling fluids with particles
  - Lagrangian SPH is very common
- Spatial Discretization
  - Two fundamental concepts when discussing spatial discretization are the Lagrangian and Eulerian reference frames
  - Mesh and grid data structures are common ways of storing simulation variables
  - We can simulate elasticity by using FEM to discretize equations of motion (FEM)
- Temporal Discretization
  - Explicit integration is fast and suitable for most simulations e.g. particles
  - Implicit integration is unconditionally stable and thus useful for “stiff problems”

Reading

- Advanced Character Physics, Thomas Jakobsen (2001) : http://www.cs.cmu.edu/afs/cs/academic/class/15462-s13/www/lec_slides/Jakobsen.pdf
- SPH Fluids in Computer Graphics M. Ihmsen, J. Orthmann, B. Solenthaler, A. Kolb, M. Teschner Proceedings of Eurographics (Strasbourg, France, April, 2014), Computer Graphics Forum, vol. , no. , pp. 21-42
- Eric Lengyel. 2011. Mathematics for 3D Game Programming and Computer Graphics, Third Edition (3rd. ed.). Course Technology Press, Boston, MA, USA.
- Bargteil, A., Shinar T. An introduction to physics-based animation, ACM SIGGRAPH 2018 Courses, 2018
- FEM Simulation of 3D Deformable Solids: A practitioner's guide to theory, discretization and model reduction, ACM SIGGRAPH 2012 Courses
- David Baraff. 2001. Physically based modeling: Rigid body simulation. SIGGRAPH Course Notes, ACM SIGGRAPH 2, 1 (2001)
- Matthias Müller, Jos Stam, Doug James, and Nils Thürey. 2008. Real time physics: class notes. In ACM SIGGRAPH 2008 classes (SIGGRAPH ’08). Association for Computing Machinery