Simulation et Applications Interactives

Physically Based Simulation

Caroline Larboulette

Fall 2018

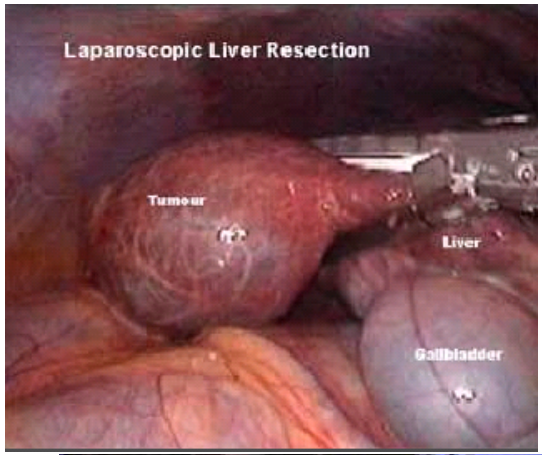
Geometry vs Physics

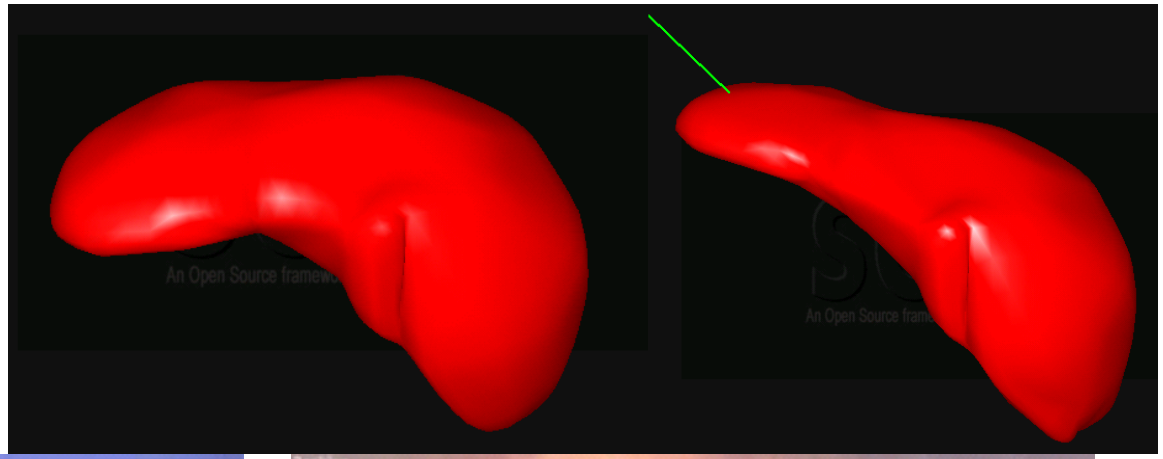
- Geometrically based animation
  - Describes the deformation by some mathematical function
- Physically based animation
  - Describes the causes of the deformation (rather than the deformation itself) by using the laws of physics
  - Also referred to as simulation

Pros & cons of Physics

- High level control for the animator
  - Easy to control a lot of objects
  - Hard to precisely obtain a specific effect
- Simulation is time-consuming
- Simulation might be unstable
- More realistic deformations
- Possibility to obtain dynamic deformations
- Possibility of interaction

Examples of Applications





Overview

- Continuum Mechanics
- Mass-Spring Systems
- Finite Elements
- Continuous Forces
- Particle Systems
- Introduction to Collision Detection and Response

Continuum Mechanics

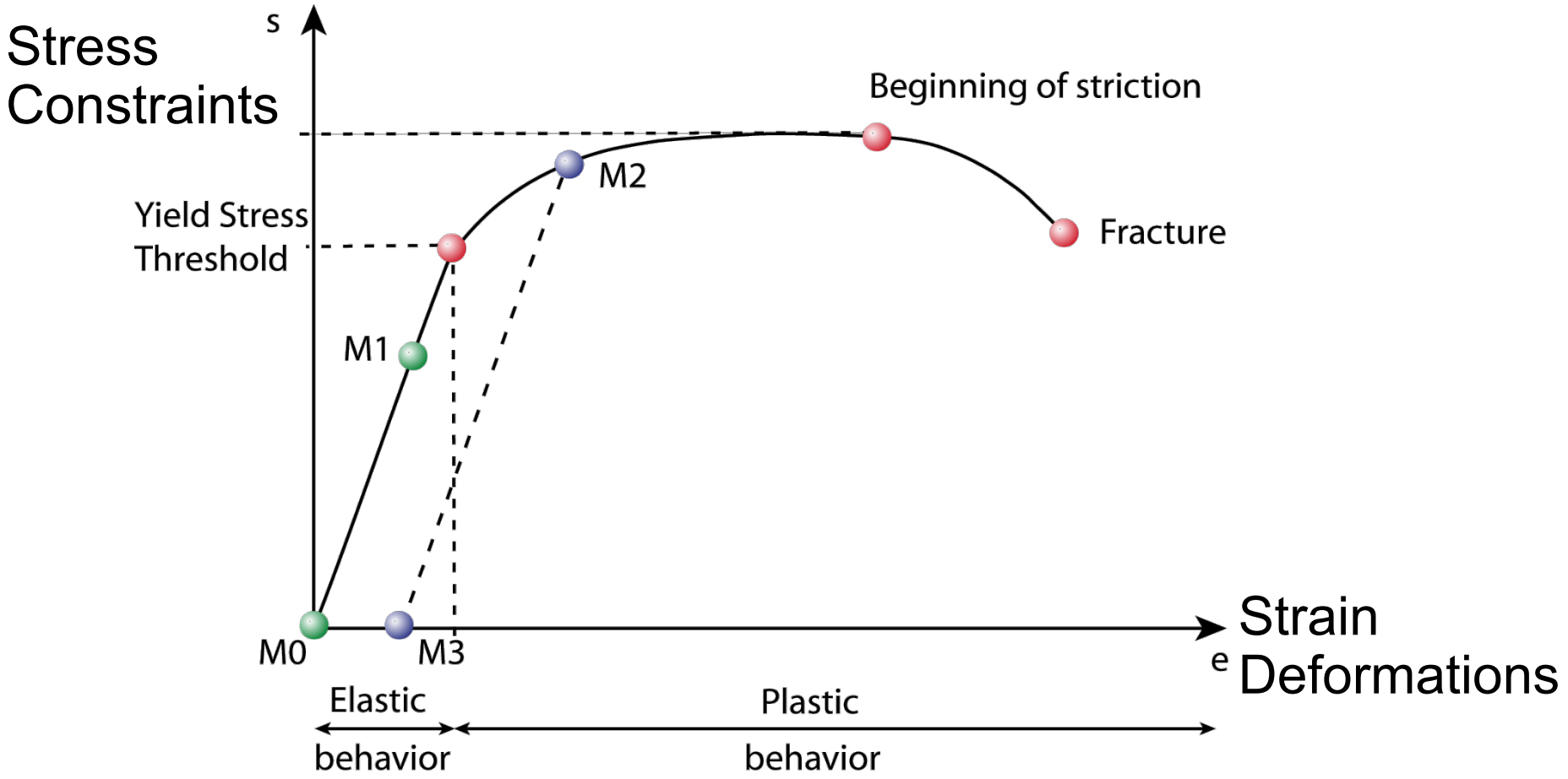
Continuum Mechanics

- Object first described in its reference position (rest state)
- Motion described as a continuous series of deformations between the rest state and the deformed state over time
- Two formalisms:
  - Lagrangian (solids)
  - Eulerian (fluids)

Lagrangian Description

- Object is composed of “particles” or “material points” i
- At each time t, the position of i is described a $P_i = P0_i + U_i$
- $P0_i$ = rest position of i
- U_i = displacement vector of i
- Function valid for any point in the solid
  - Best adapted if structure does NOT change over time (solid not fluid)
- Continuous description

Solid Deformation: Strain-Stress Curve



Deformation

Types

- Elasticity
- Viscosity
- Plasticity
- Fracture

Models

Continuous / Discrete

- Continuous model is not adapted to Computer Graphics (analytical solution difficult to compute -- if possible)
- Discretization in space
  - Mass-Springs, Finite elements
- Discretization in time
  - Time step and numerical integration

Particle

- Single point with a mass associated, usually represented as a sphere

- Position x in m (meters)
- Velocity v in $m.s^{-1}$
- Acceleration a in $m.s^{-2}$
- Mass m in Kg

Notations

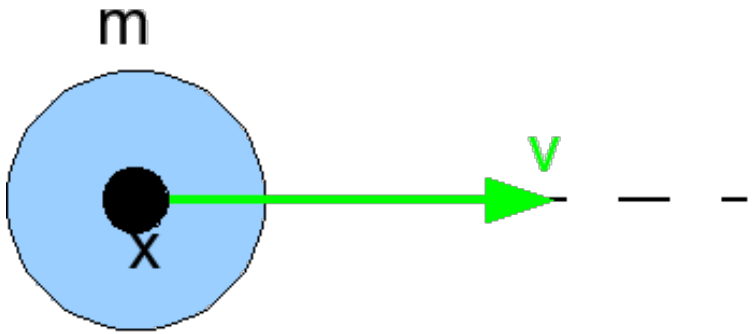
- Let $x(t)$ be the trajectory of a particle over time
- $\dot{x}(t) = \frac{dx}{dt}$ its velocity (and tangent to the trajectory)
- $\ddot{x}(t) = \frac{d^2x}{dt^2}$ its acceleration
- For simplification, we will also use:
- p(t), the position of a particle at time t
- v(t) its velocity at time t
- a(t) its acceleration at time t

Newton's Laws

First Law of Motion

Law of inertia

An isolated system (no forces applied) has a constant velocity

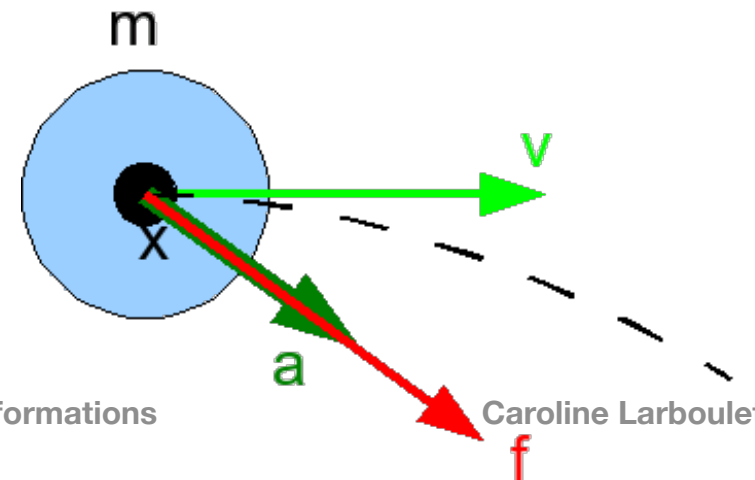


Newton's Laws

Second Law of Motion

$$\vec{F} = m \cdot \vec{a}$$

The acceleration of a particle only depends on the Forces (F is net force in Kg.m/s^2) applied to it and its mass m



Newton's Laws

Third Law of Motion

Interaction Principle

In every interaction, there is a pair of forces, with an equal and opposite direction, on both objects (i.e. the net force applied to an isolated system is null)

$$f_{1\to 2} = -f_{2\to 1}$$

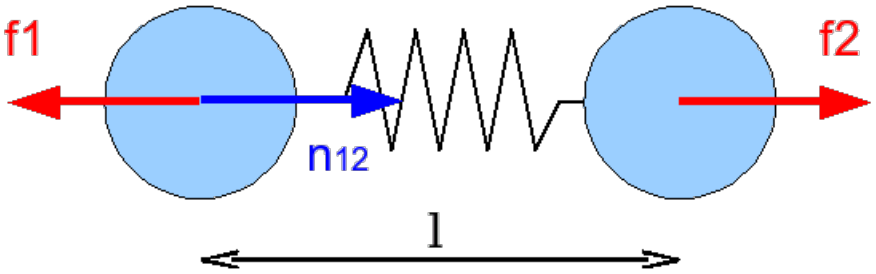
Overview

- Continuum Mechanics
- **Mass-Spring Systems**
- Finite Elements
- Continuous Forces
- Particle Systems
- Introduction to Collision Detection and Response

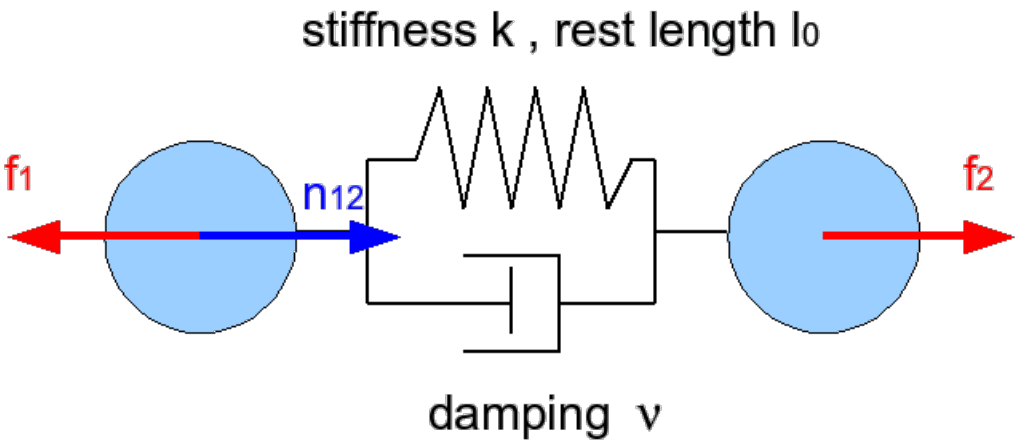
Mass-Spring Systems

Linear Spring (1D)

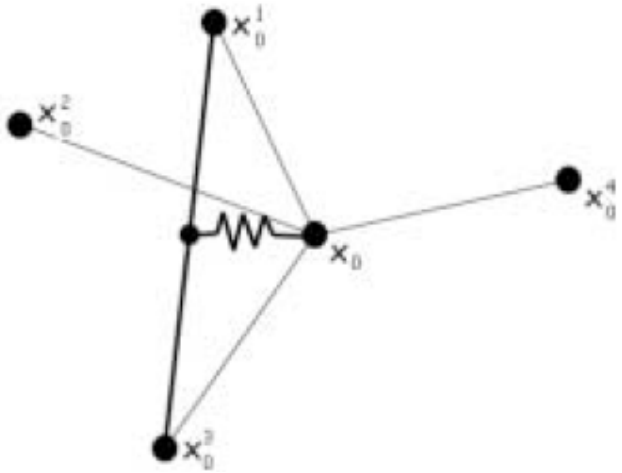
- F = - k u
- k = stiffness
- u = elongation



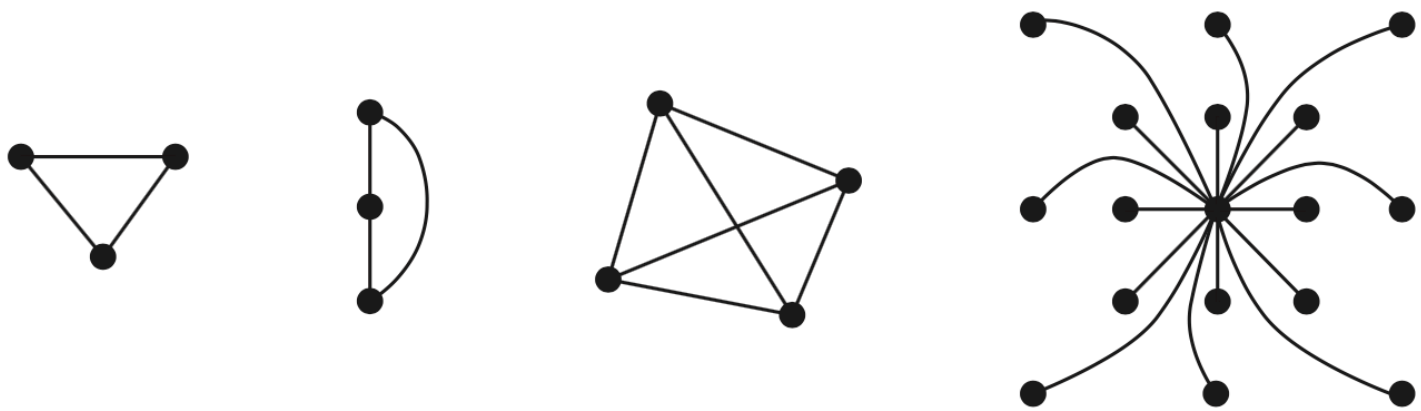
- If damping: acts on the velocity and in opposite direction
- F = - k u + d v
- !! To the vector dir in 3D !!

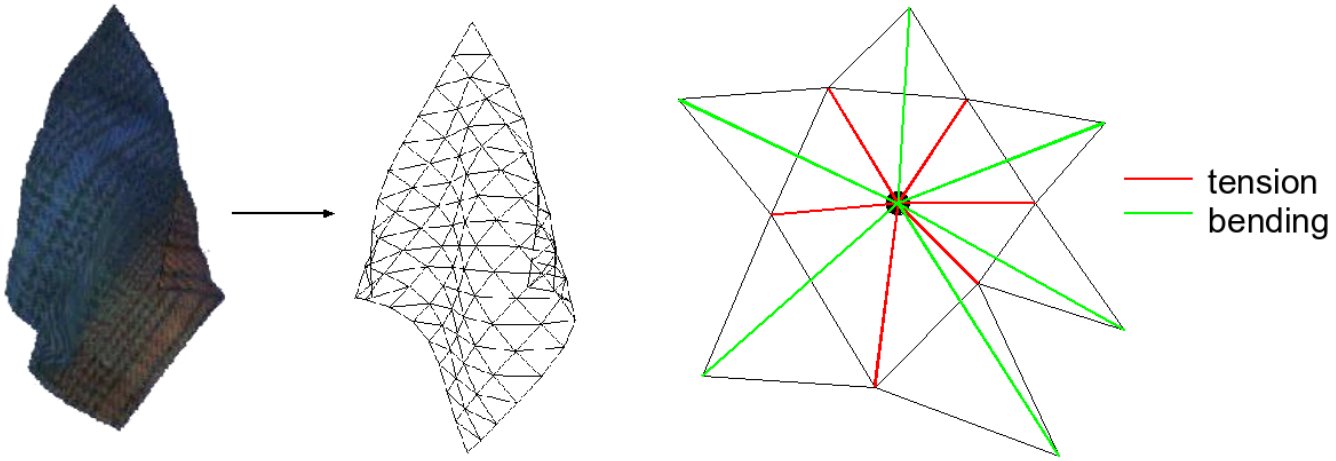


Angular Spring



Mass-Spring Systems





Continuous object Discrete model Particle neighborhood

Mass-Spring - Overview

- Animate() //runs at least at dt

{

  for each particle

  {

    - //Compute the forces (gravity, springs, other)
    - //Compute the acceleration (F=ma)
    - //Integrate velocities and positions (Euler, RK2 ...): a -> v -> p

  }

}

Forces

- Gravity
  - P = m G with m in Kg and G = 9.81 N.Kg-1
- Elastic

$$Fs = -(k(||P_2 - P_1|| - l_0)).n_{12}$$

$$Fd = (d(v_2 - v_1).n_{12}).n_{12}$$

$$F = (-k(||P_2 - P_1|| - l_0) + (d(v_2 - v_1).n_{12})).n_{12}$$

$$n_{12} = \frac{P_2 - P_1}{||P_2 - P_1||}$$

Numerical Integration

- Euler explicit

$$v_{t+dt} = v_t + dt \cdot a_t$$

$$p_{t+dt} = p_t + dt \cdot v_t$$

- Very unstable if more than one spring
  - Use the new Vt+1 for more stability

$$p_{t+dt} = p_t + dt \cdot v_{t+dt}$$

- Simple and fast but unstable
  - Use an implicit or hybrid scheme instead

Numerical Integration

- Euler Explicit
- Runge Kutta (RK2)
- Runge Kutta (RK4)

Example:

$$x(t) = 100\ e^{(0.1t)}$$
$$v(t) = dx/dt\ x(t) = 10\ e^{(0.1t)}$$
$$a(t) = dx/dt\ v(t) = e^{(0.1t)}$$

Euler Explicit = RK1

```
Runge Kutta 1
y'=f(y,t) and y(t0) = y0
h is the time-step
yn+1 = yn + h f(t, yn)

This corresponds to Euler:
p(t+dt) = p(t) + dt*v(t)
v(t+dt) = v(t) + dt*a(t)
```

- Example on board

Runge Kutta 2 (RK2)
Mid-point

- Example on board …

```
Runge Kutta 2

k1 = f(tn, yn)
k2 = f(tn+h/2, yn + h/2*k1)

yn+1 = yn + h*k2

At k1:
p_k1(t) = p(t)
v_k1(t) = v(t)  //slope k1
a_k1(t) = a(t)  //Computed with current p and v

At k2:
p_k2(t+dt/2) = p(t) + v_k1*dt/2
v_k2(t+dt/2) = v(t) + a_k1*dt/2    //slope k2
a_k2(t+td/2) //Computed with current p_k2 and v_k2
```


Runge Kutta 4 (RK4)

- **Example continued …**

```
k1 = f(tn, yn)
k2 = f(t + h/2, yn + h*k1/2)
k3 = f(t + h/2, yn + h*k2/2)
k4 = f(t + h, yn + h*k3)
yn+1 = yn + (k1 + 2 k2 + 2 k3 + k4)*h/6
p(t+dt) = p(t) + dt/6 ( k1 + 2 k2 + 2 k3 + k4 )
```

k1 is the slope at the beginning of the interval;
k2 is the slope at the midpoint of the interval, using slope k1 to determine the value of y at the point tn + h/2 using Euler's method;
k3 is again the slope at the midpoint, but now using the slope k2 to determine the y-value;
k4 is the slope at the end of the interval, with its y-value determined using k3. The easiest way to go it to store various positions, velocities and derivatives for each runge kutta step (1 to 4).

Implicit Schemes

- Computed using “future” values of variables
- Needs to solve a system of equations
- Euler implicit

$$\begin{aligned} \dot{x}(t+h) &= \dot{x}(t) + h\ddot{x}(t+h) = \dot{x}(t) + hf(t+h)/m \\ x(t+h) &= x(t) + h\dot{x}(t+h) \end{aligned}$$

Use of mass-springs

- Mushroom example
- Burning Paper
- Chadwick: FFD coupled with mass-springs

Mass-Spring under Maya

Overview

- Continuum Mechanics
- Mass-Spring Systems
- **Finite Elements**
- Continuous Forces
- Particle Systems
- Introduction to Collision Detection and Response

Finite Elements

Finite Elements

- More continuous model than mass-springs
- Discretization in elements, but continuous functions inside the elements to compute
  - Its mass M
  - Its stiffness K
  - Its damping D (if applicable)

FEM

- The basis functions have to be chosen carefully
  - Tradeoff between correct approximation of the material behavior
  - And the ease of computation (the displacements of the nodes need to be expressed as a linear combination of the basis functions)

Basis functions

- To have simple equations, it is common to use functions that are linear by piece
  - Value 1 at node i
  - Value 0 at all other nodes
- There are other possibilities (Hermite …)

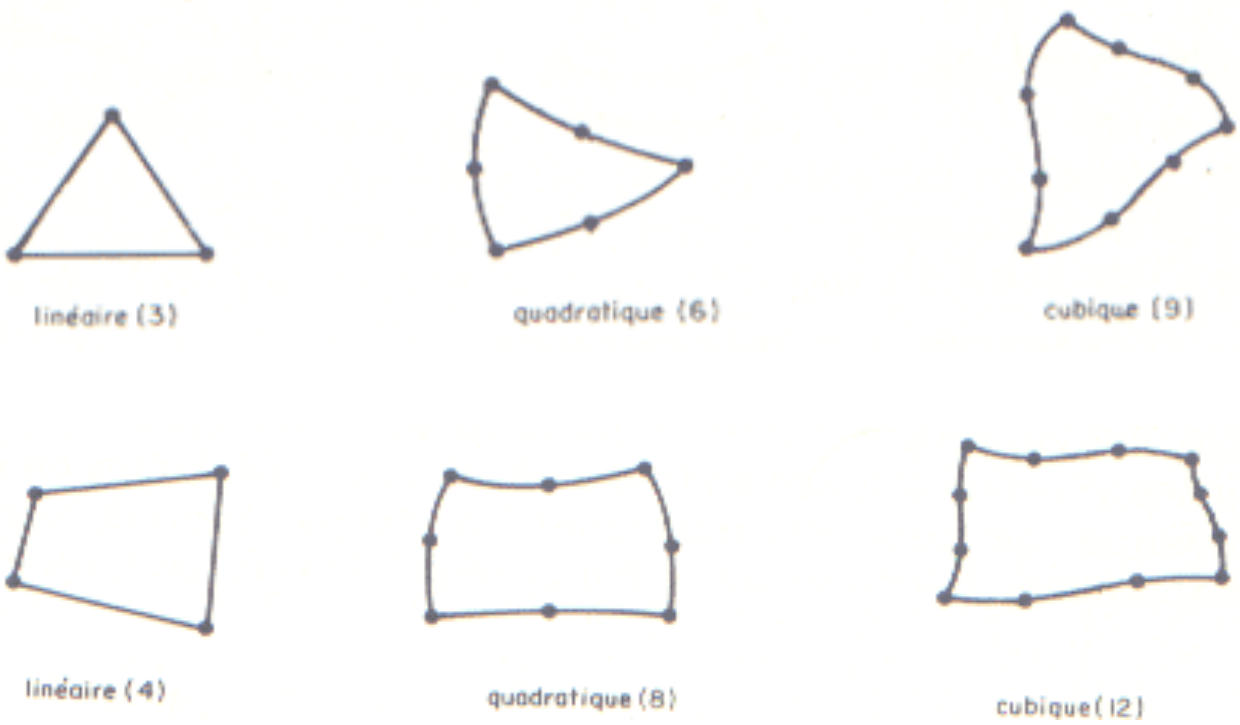
Basis Functions ...

- Over time, we need to compute the displacement u_i of the n nodes a_i
- Projected on the basis of functions, we can compute the position P of any point inside the element:

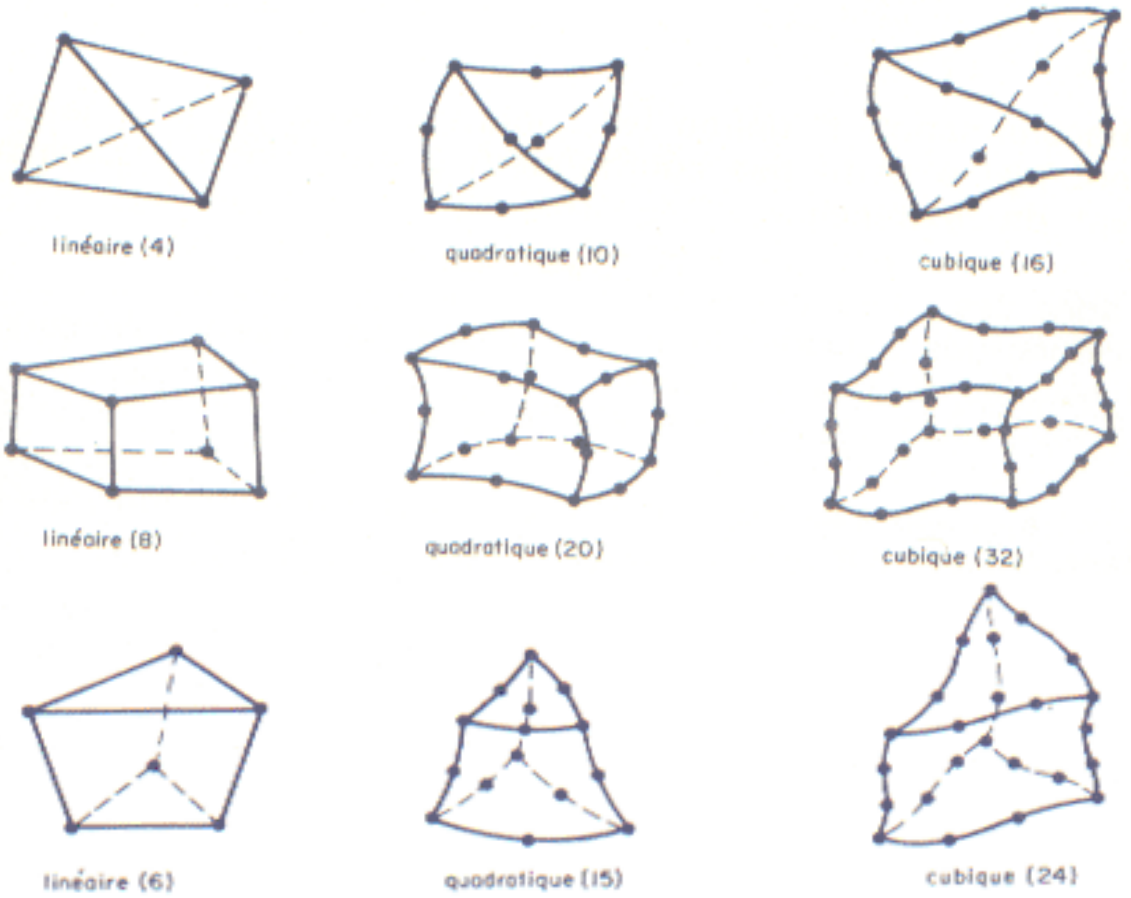
$$P = \sum_i a_i \Phi_i$$

Φ_i are the basis functions

Finite Elements 2D



Finite Elements 3D



Equilibrium Equation

- By trying to minimize the error wrt the behavior of the material (strain-stress curve), we obtain

$$F = K\,u$$

K is the stiffness matrix [n*n] that depends on the material expressed (Young, Poisson) in the basis

F is the vector of forces [n*1] expressed in the basis

u is the displacement vector [n*1] of all u_i's

Example of K matrix

Elongation $= \Delta \overrightarrow{u} = \overrightarrow{u_2} - \overrightarrow{u_1}$

$\overrightarrow{F_1} = k\overrightarrow{u_1} - k\overrightarrow{u_2}$

$\overrightarrow{F_2} = -k\overrightarrow{u_1} + k\overrightarrow{u_2}$

F_1, F_2, u_1, u_2 are respectively the forces and the displacements for each node.

Axial forces applied to the spring

$$\begin{bmatrix} F_1 \\ F_2 \end{bmatrix} = \begin{bmatrix} k & -k \\ -k & k \end{bmatrix} \cdot \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = K \,.\, u$$

To solve the equation

- K is the stiffness matrix

$$F = K \,.\, u$$

$$u = K^{-1} \,.\, F$$

- Computing K^{-1} is the costly step. It is common to pre-invert this matrix.

Same with Damping

- If damping is used, it modifies the velocities. It affects the strain-stress curve (acceleration != 0)
- The equilibrium equation becomes

$$F = Ma + Dv + Ku$$

M: mass matrix

D: damping matrix

a and v: acceleration and velocity

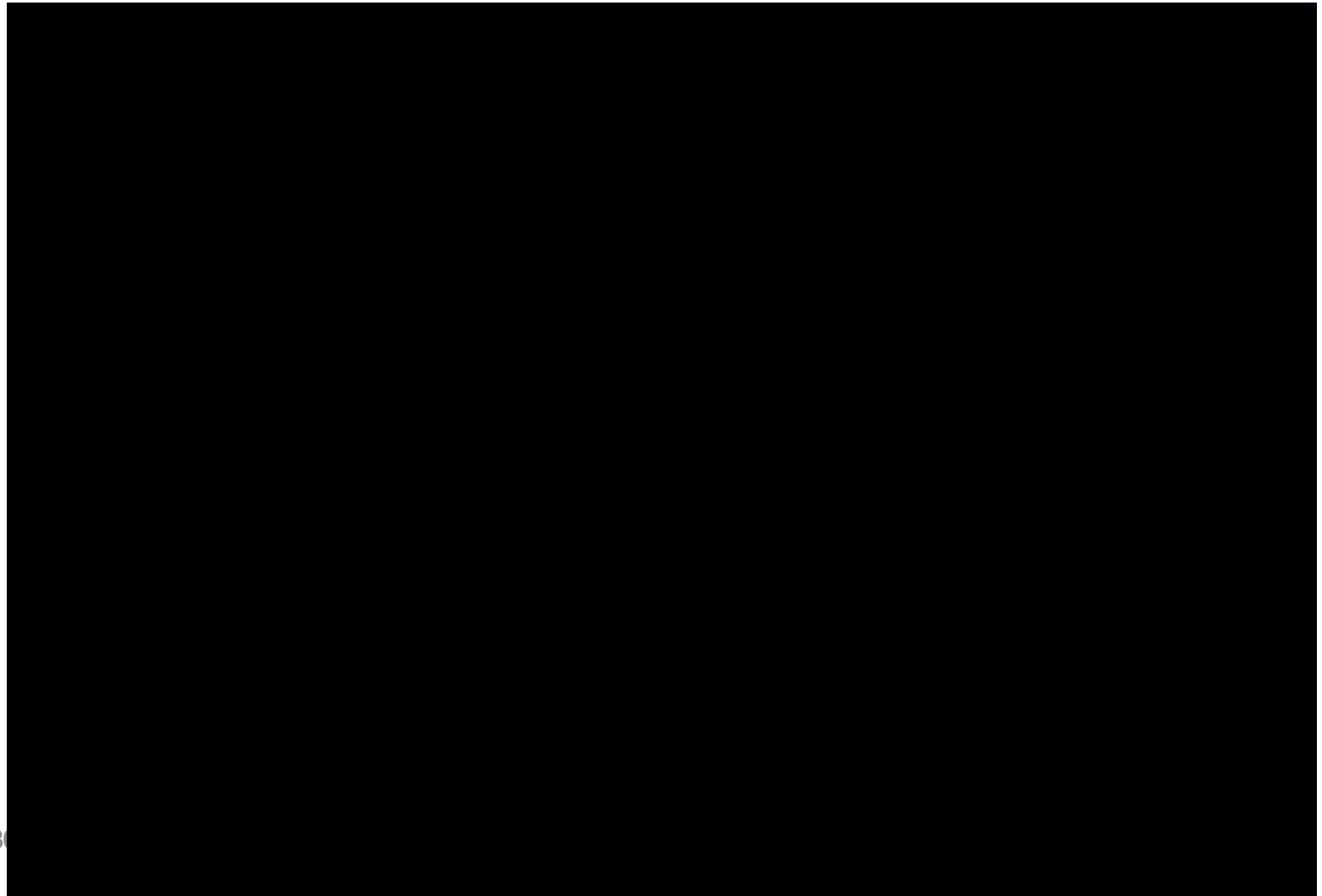
=> Necessary for simulating dynamics

FEM

Capell

FEM + Modal Analysis

DyRT



Overview

- Continuum Mechanics
- Mass-Spring Systems
- Finite Elements
- **Continuous Forces**
- Particle Systems
- Introduction to Collision Detection and Response

Continuous Forces

Singularities

- Continuous forces to create force fields
  - Source
  - Sink
  - Vortex
  - Directional

Singularities

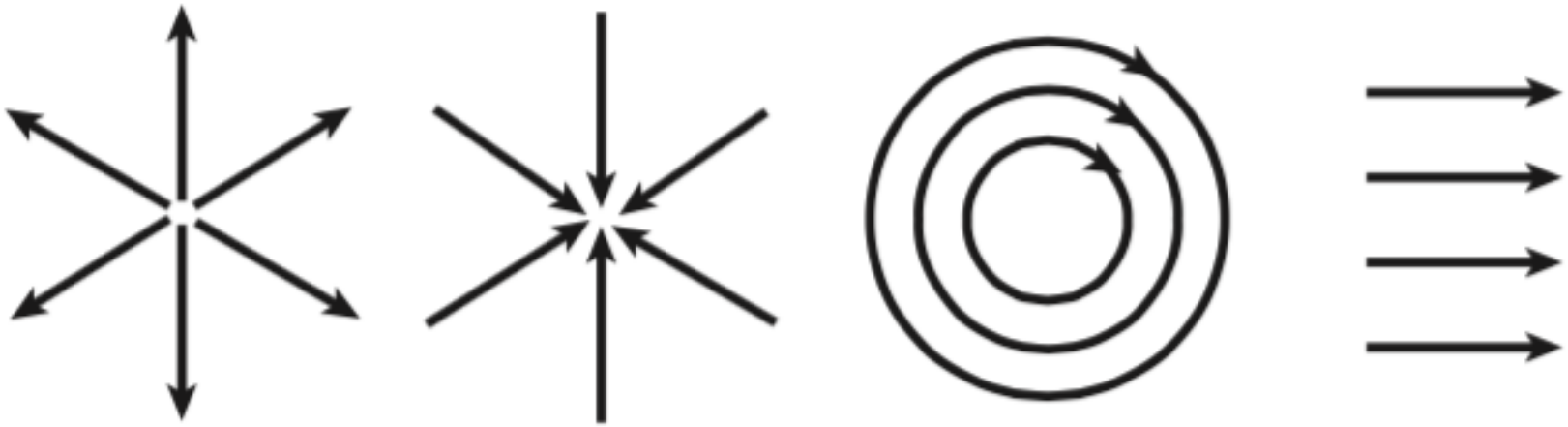
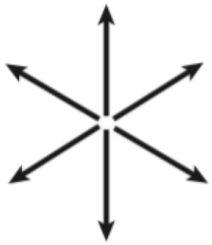


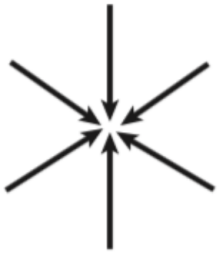
Figure 3: The four types of singularities used to model the fluid's velocity field. From left to right: Source, Hole, Whirlwind and Directional Field.

Source



- $$\mathbf{S}(\mathbf{p}) = \frac{\Phi}{\frac{\Phi}{\Phi_{max}} + d(\mathbf{p},\mathbf{C})^2} \frac{\vec{\mathbf{C}}\mathbf{p}}{||\vec{\mathbf{C}}\mathbf{p}||}$$

Sink

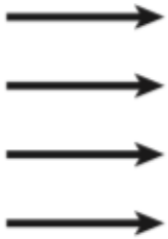


$$\mathbf{H}(\mathbf{p}) = -\mathbf{S}(\mathbf{p}) = -\frac{\Phi}{\frac{\Phi}{\Phi_{max}} + d(\mathbf{p},\mathbf{C})^2} \frac{\vec{\mathbf{Cp}}}{||\vec{\mathbf{Cp}}||}$$

Vortex

$$\mathbf{W}(\mathbf{p}) = \pm \frac{\Phi}{\frac{\Phi}{\Phi_{max}} + d(\mathbf{p},\mathbf{C})^2} \frac{\vec{\mathbf{C}}\mathbf{p} \times \vec{R}}{||\vec{\mathbf{C}}\mathbf{p} \times \vec{R}||}$$

Directional



- Force move things in a given direction
- Can be a sin or cos for wave effects

$$\mathbf{D}(\mathbf{p}) = \Phi(\mathbf{p}, t).\vec{v};$$

Example

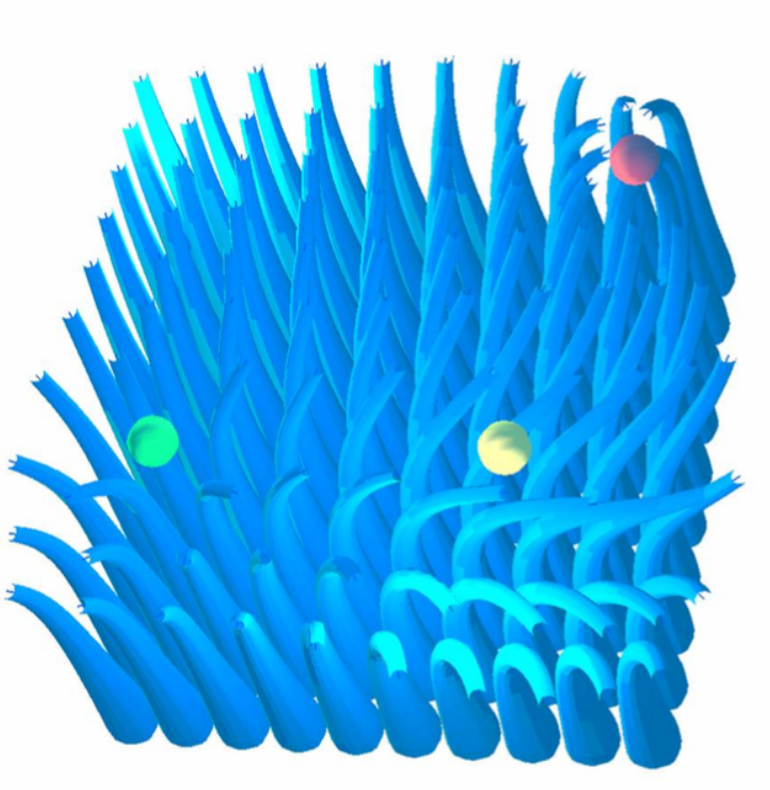


Figure 12: Influence of a Source (green), a Hole (red) and a Whirlwind (yellow) on a group of identical fibers.

Overview

- Continuum Mechanics
- Mass-Spring Systems
- Finite Elements
- Continuous Forces
- **Particle Systems**
- Introduction to Collision Detection and Response

Particle System

- Emitter emits particles
  - Spawning rate (how many particles are generated per time unit)

$$n = (t - t_{last}) / \text{ rate}$$

  - Spawning position (emitter pos)
  - Particles initial velocity
    - Magnitude and direction
  - Particles lifetime

Particle System

- Particles live their lives
- Before being animated, check if alive or dead
- If alive, update velocity / position
  - Can be very simple: vel -= 0.2
  - Or sophisticated: forces, a, v, p
- Check for collisions with other objects

Particle System

- Rendering of particles
  - As points (color / size according to age)
  - As billboards (quad always facing the user)
  - Metaballs (offline rendering) for liquids
  - 3D mesh (slower)

Particle System

- In practice: a lot of particles 10^5
- Use a table to store the particles
- Max number of particles
  - Fixed size table
  - Not need to create/destroy particles, just replace them

Overview

- Continuum Mechanics
- Mass-Spring Systems
- Finite Elements
- Continuous Forces
- Particle Systems
- **Introduction to Collision Detection and Response**

Introduction to Collision Detection and Response

Collision Detection

Motivation

Teschner, Kimmerle, Heidelberger, Zachmann, Raghupathi, et al., Collision Detection for Deformable Objects. Computer Graphics Forum, 2005.

- Why detect collisions and respond ?
  - To change object trajectories
  - To change object shapes
  - To prevent interpenetration of mesh (can lead to unstable simulation)
- Two types of collisions
  - Object vs object
  - Self-collision

Detection

- Collisions can be detected on any type of mesh, for any type of animation
  - What is tested is whether there are interpenetrations at a given time t
- However, depending on the type of surface/volume, it can be
  - Easy (fast) or hard (slow) to compute
  - Approximated or exact
  - Different types of responses

Polygonal Models

- Not feasible for each polygon versus each polygon (too many ray intersections to compute !)
  - Possibility of using Octrees, KD-trees
  - Approximation by bounding boxes or implicit surfaces (spheres) : BVH
  - Make use of a stochastic approach

Bounding Volume Hierarchies (BVH)

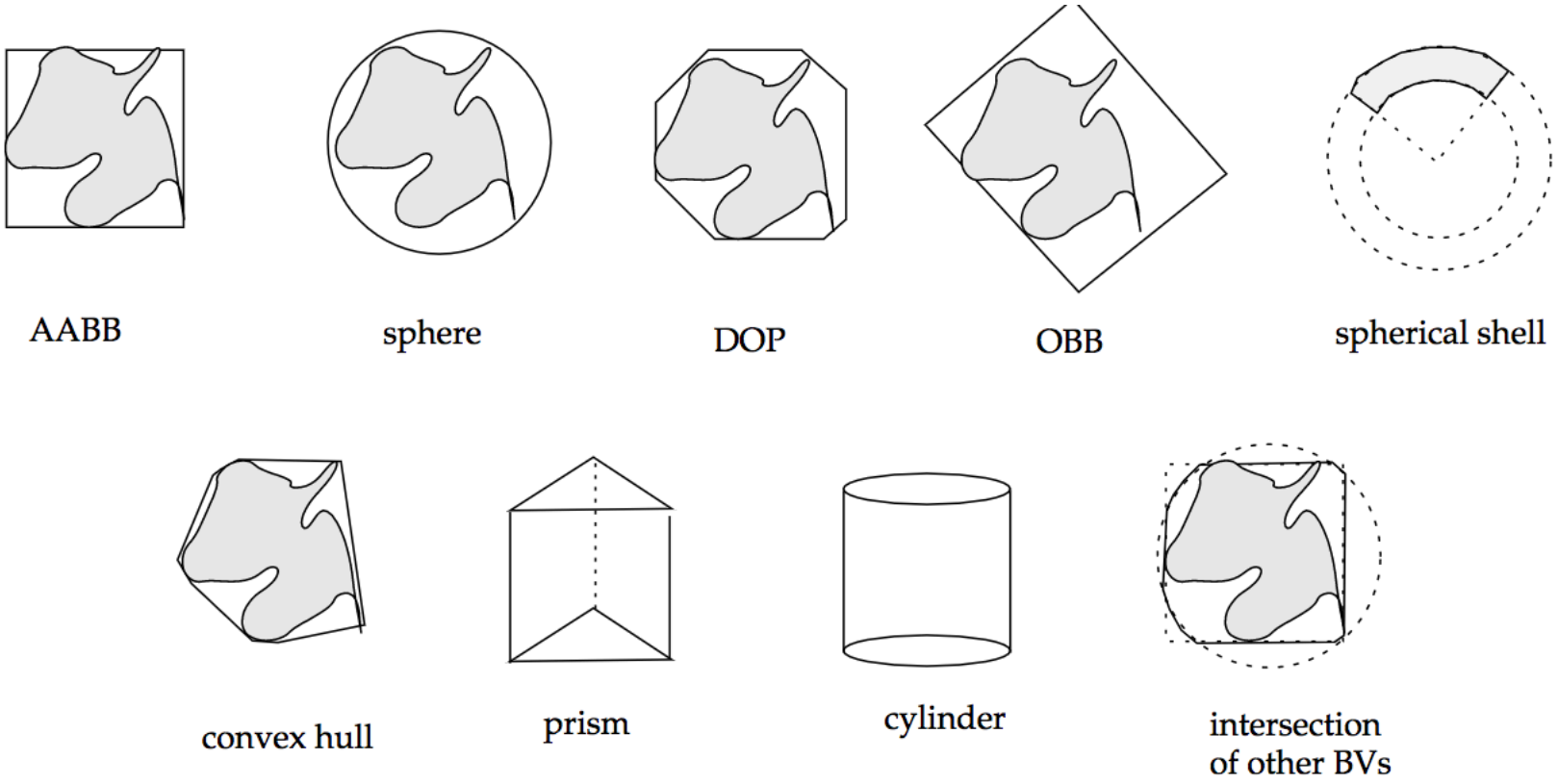
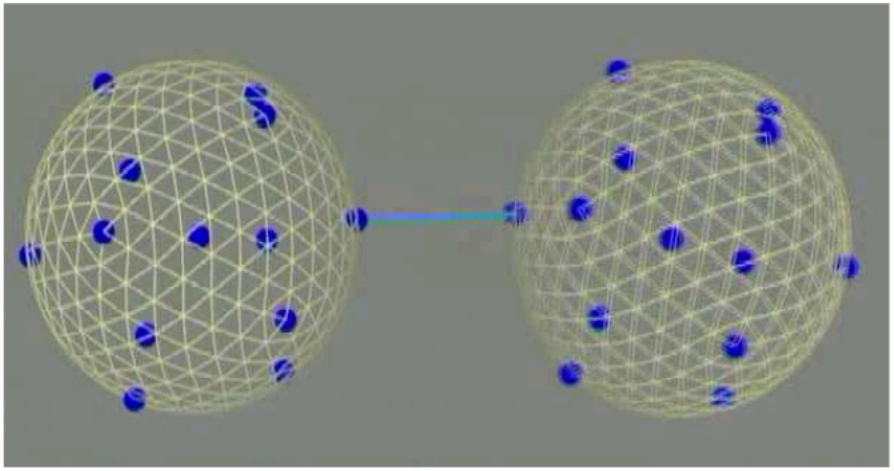
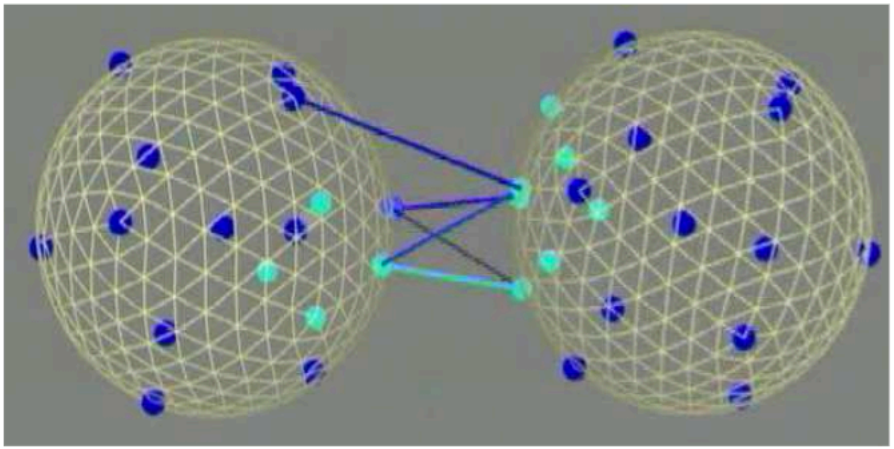


Figure 4: *A variety of bounding volumes has been proposed for hierarchy-based collision detection.*

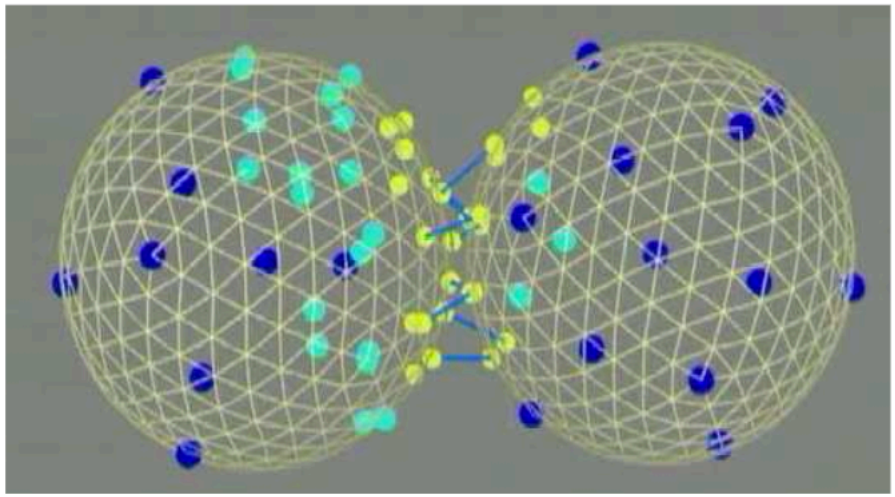
Stochastic Approach



(a)



(b)



(c)

Figure 10: *Illustration of stochastic collision detection between two volumetric bodies: Pairs of features (depicted by lines) across different resolutions (different colored points) being tracked as the objects approach each other.*

Subdivision Surfaces

- By nature, the collision detection and response can be done on the coarse polygonal mesh
  - Good approximation
  - Use same techniques as for polygons

Parametric Surfaces

- Curves are polynomial functions of degree 3:
  - Intersection not easy to compute

  Forget it !
- However, for normal, regular and positive splines (ex. B-spline), you can use the convex hull: probably the best approximation you can get !

Implicit Surfaces

- The type of surface that is maybe most appropriate to collision detection
  - Penetration can be computed exactly
  - Zone of contact can also be computed exactly
- However, not practical to render
  - Use implicit surfaces to approximate your object

Bounding Sphere

- Orientation independent
- Center = mesh center
- Radius = furthest vertex
- Effective for convex, oval bodies
- Bounding ellipsoid for more elongated bodies

Collision Response

Collision Response

- Apply repulsion forces
  - Penalty Method for physically based simulations: put a spring between the two colliding solids with a rest length of 0 (or epsilon) and high stiffness

Using Penalty Method

Selle et al., A mass spring model for hair simulation, SIGGRAPH 2008

Using Penalty Method

Selle et al., A mass spring model for hair simulation, SIGGRAPH 2008

Using Penalty Method

Selle et al., A mass spring model for hair simulation, SIGGRAPH 2008

Using Penalty Method

Selle et al., A mass spring model for hair simulation, SIGGRAPH 2008

Collision Response

- Modify speed for solids or particles
  - Elastic Collision
  - Soft Collision
  - Collision of solid/particle with a plane

Elastic Collision

- 2 particles with m_1 and m_2 collide
- v_1 and v_2 before collision
- v_1' and v_2' after collision
- Quantity of Motion is preserved:

$$m_1v_1 + m_2v_2 = m_1v_1' + m_2v_2'$$

- Kinetic Energy is preserved:

$$m_1v_1^2 + m_2v_2^2 = m_1v_1'^2 + m_2v_2'^2$$

Elastic Collision

$$m_1(v_1-v_1') = m_2(v_2'-v_2)$$

$$m_1(v_1^2-v_1'^2) = m_2(v_2'^2-v_2^2)$$

$$v_1+v_1' = v_2+v_2'$$

$$u' = -u$$

$u = v_2 - v_1$ (relative speed before collision)

$u' = v_2' - v_1'$ (relative speed after collision)

Elastic Collision

$$v'_1 = \frac{2m_2v_2 + v_1(m_1 - m_2)}{m_1 + m_2} \qquad v'_2 = \frac{2m_1v_1 + v_2(m_2 - m_1)}{m_1 + m_2}$$

- If $m_1 = m_2$, the solids exchange their velocities
- $v_1' = v_2$ and $v_2' = v_1$

Soft Collision

- If colliding solids interpenetrate, they glue together
- $v_2' = v_1'$
- There is a loss of kinetic energy
- Coefficient of restitution e (empirical)

$$e = \frac{v_1' + v_2'}{v_1 + v_2}$$

Soft Collision
with the Floor

- Let a ball fall from height h with a restitution coefficient e
- Relative speed u = speed v of the ball (the ground does not move)
- Let h' be the new height

$$\frac{1}{2}mu^2 = mgh$$

$$h' = e^2 h$$

$$\frac{1}{2}mu'^2 = mgh'$$

Collision with a Plane

- Collision detection
- P: point on the plane
- Q: particle
- n: normal of the plane

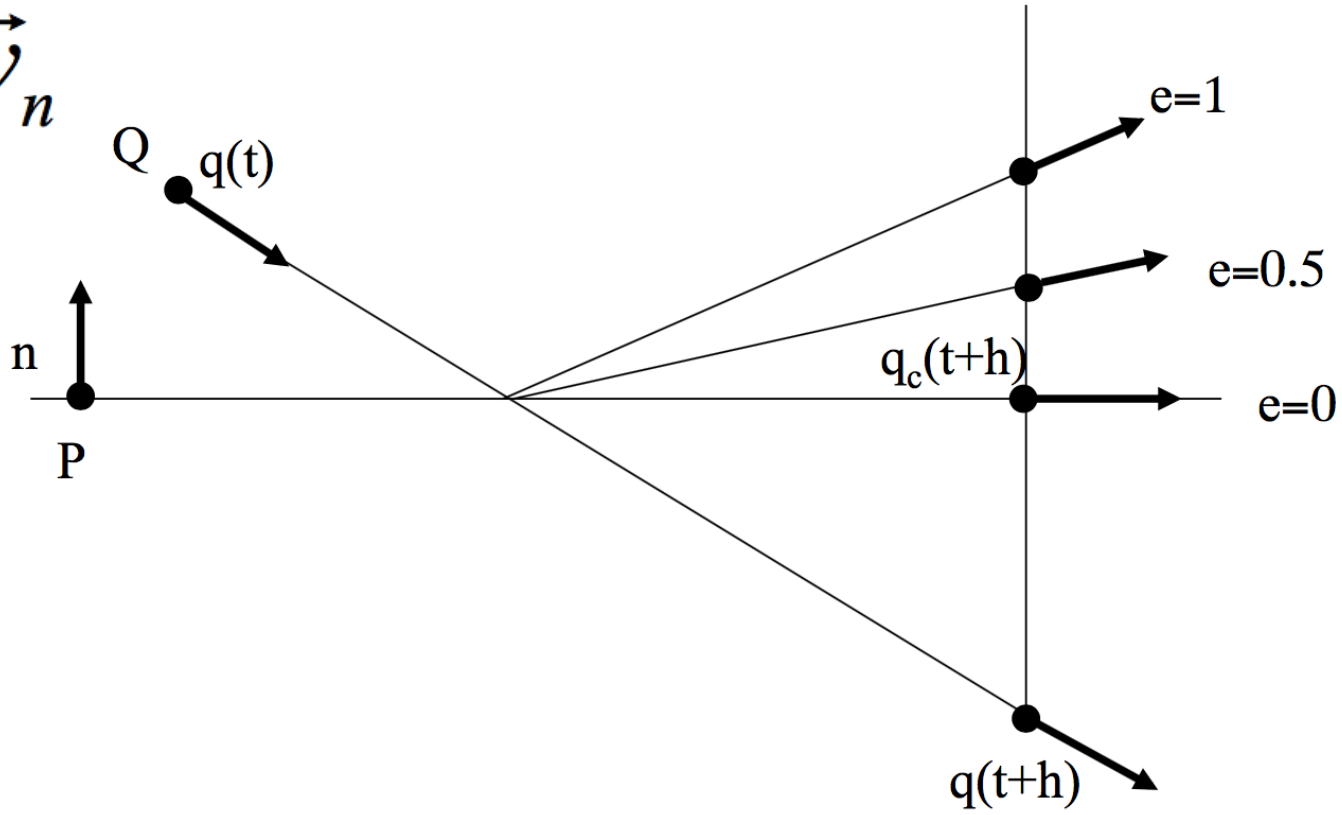
$$\vec{PQ}.\vec{n} < 0$$

- Tangential speed unchanged
- Normal speed inverted

Collision with a Plane

$$\vec{v} = \vec{v}_t + \vec{v}_n$$

$$\vec{v} \leftarrow \vec{v}_t - e\vec{v}_n$$



Collision with a Plane

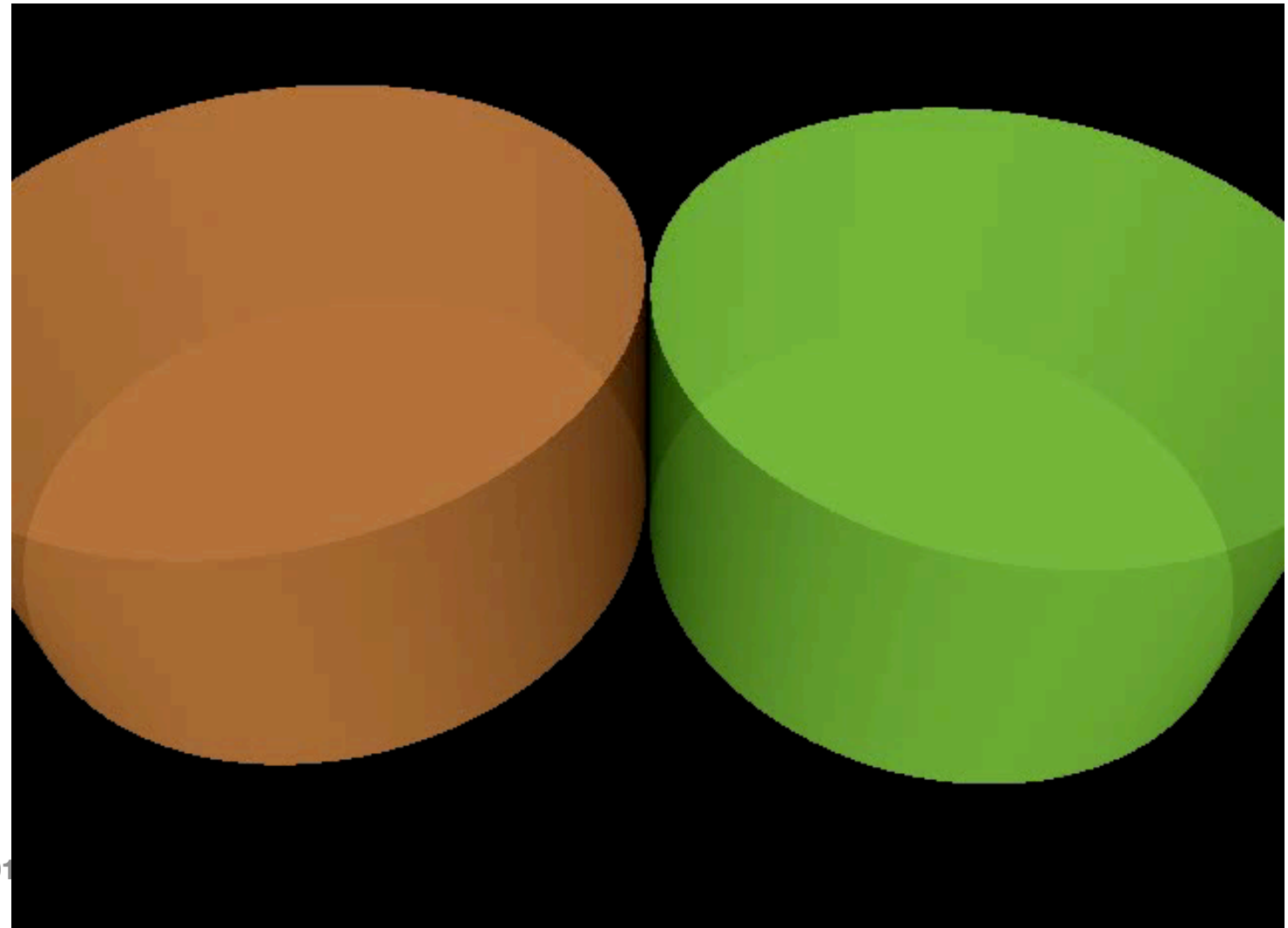
- e = 0: collision absorbed
- e = 1: perfectly elastic collision

Collision Response

- Apply some displacement along the colliding normal
  - For geometrically based techniques
  - Example with distance fields

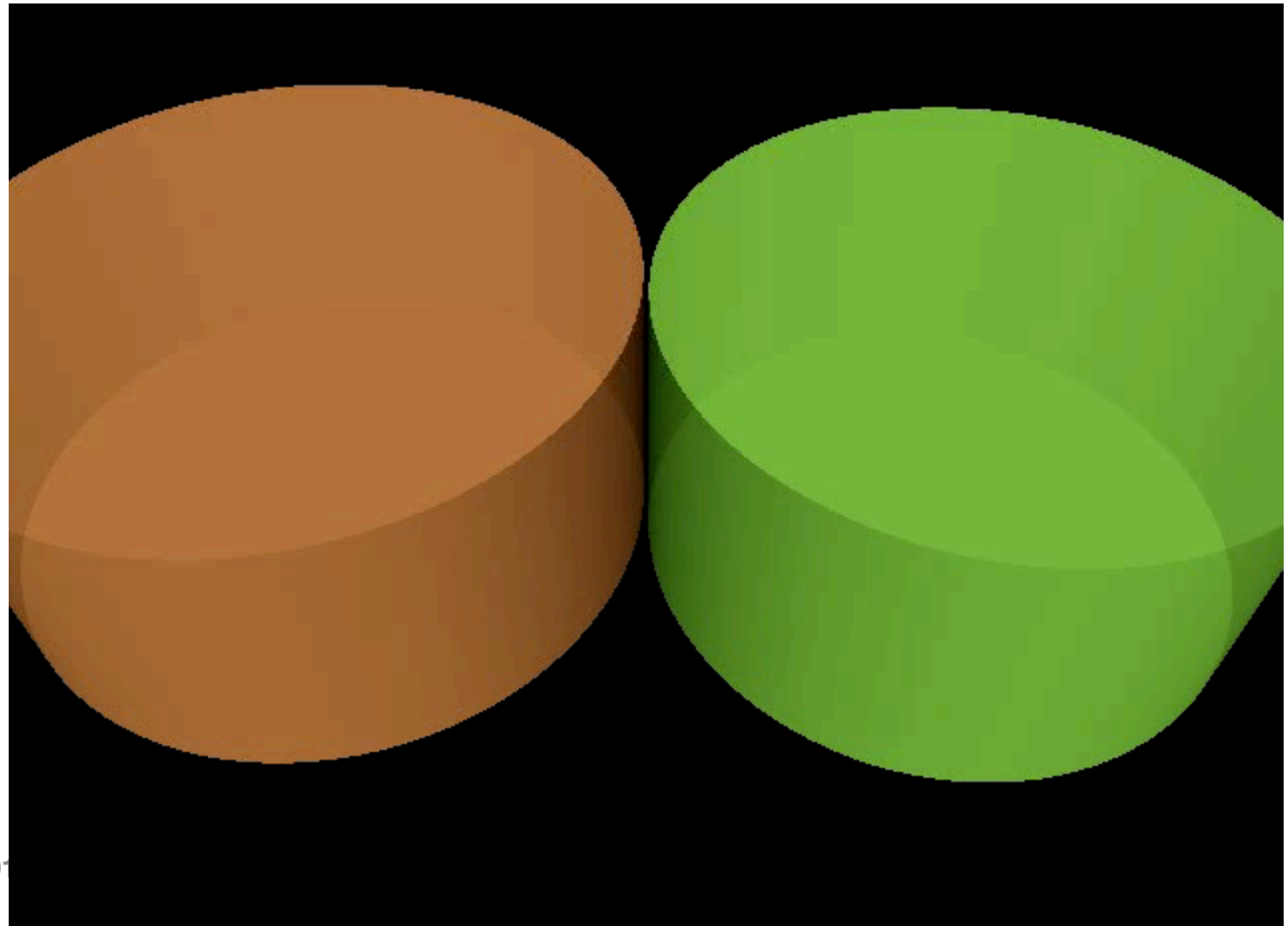
2D Distance Field

Hoff III et al. Fast and Simple 2D Geometric Proximity Queries Using Graphics Hardware, 2001



2D Distance Field

Hoff III et al. Fast and Simple 2D Geometric Proximity Queries Using Graphics Hardware, 2001



2D Distance Field

Hoff III et al. Fast and Simple 2D Geometric Proximity Queries Using Graphics Hardware, 2001