Animation Models

3D Graphics

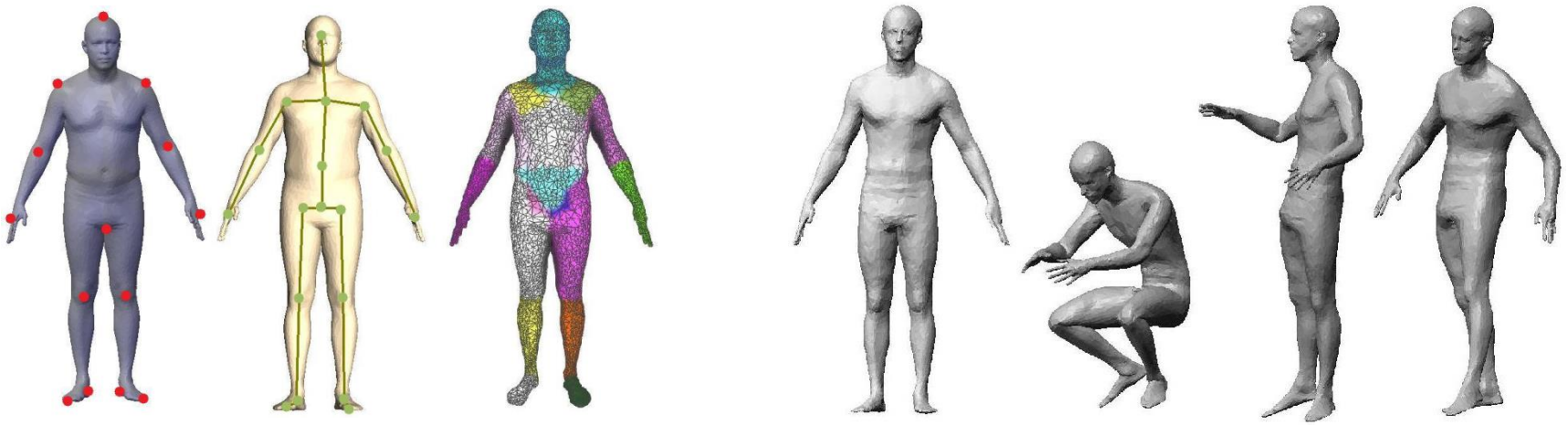
What we see today

- Skeleton-based animations
  - Suitable to animate human-like models
- Blendshape-based animations
  - Suitable to animate faces
- Data-driven animations
  - Allow to model motion of human-like characters
- Physics-based animations & collisions
  - Suitable for unstructured natural phenomena: water, gravel, …

Skeleton-Based Animations

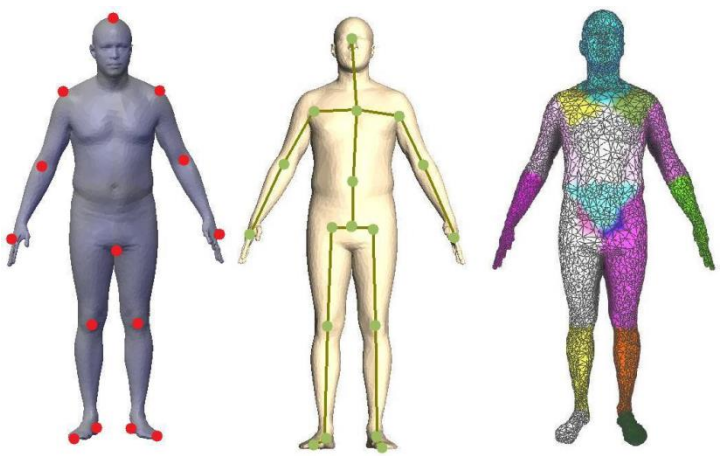
Main Idea (1)

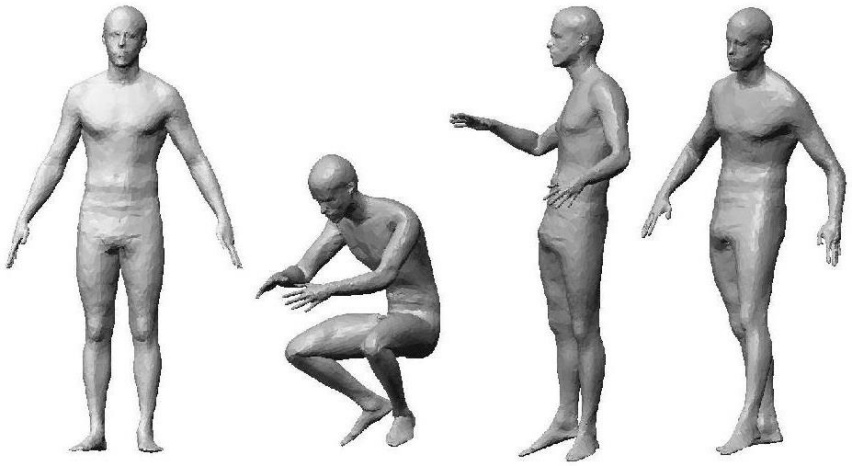
- Use piecewise as-rigid-as-possible deformation
  - “Pieces” are defined using “bones” of a skeleton
  - Pieces overlap at joints
  - Bone b_j rigidly transformed with matrix T_j
  - T_j composed by hierarchical structure



Main Idea (2)

- Use piecewise as-rigid-as-possible deformation
  - Point p_i is assigned one weight $\omega_{i,j}$ per bone
  - Point p_i transformed as $p_i^* = (\sum_j \omega_{i,j} T_j)\, p_i$





Properties

- Method called **linear blend skinning**
- Simple to use
- Linear
- Creating artifacts close to joint locations for large deformations

Extensions

- Many extensions exist to improve on artifacts
- Popular models include
  - Spline skinning
  - Dual quaternion skinning

Applications

- Use model to deform
  - Humans
  - Animals with skeleton
- Used for
  - Animation
  - Correspondence computation
  - Tracking deforming models

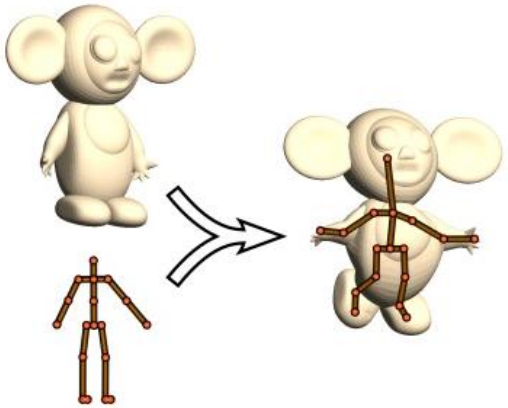
Standard Use in Animation

- Manually define skeleton and hierarchy
- Manually define rigging weights
- Animation artists carefully choose these quantities to achieve desired effects

Extensions

- Automatic computations exist for
  - Skeleton computation
  - Skinning weight computation
  - Statistical models of body shape variation due to change of identity and/or motion
- Used for scenes that are not shown as close-ups (e.g. crowds)

Image from [Stolpner, Siddiqi, Whitesides, Approximating the Medial Axis by Shooting Rays: 3D Case, CCCG 2011]



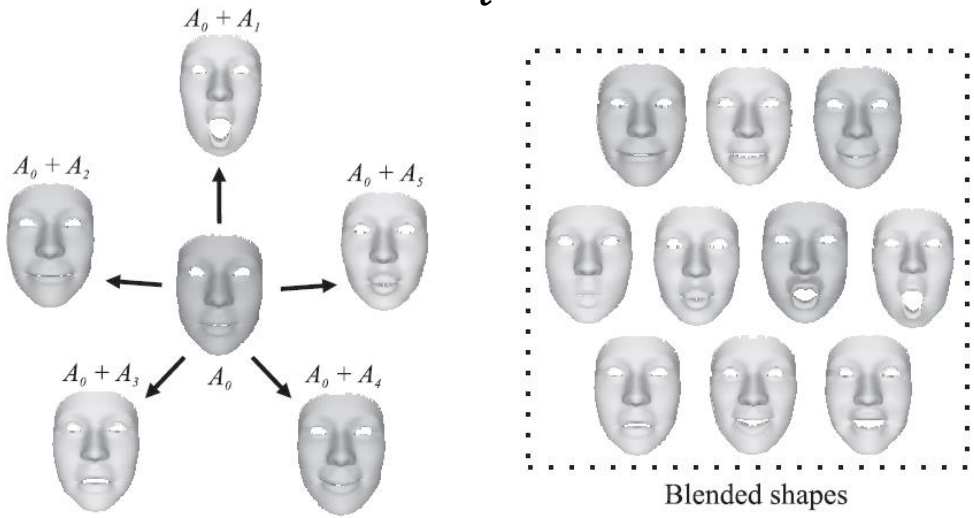
Method and Figure from [Baran, Popovic, Automatic Rigging and Animation of 3D Characters, SIGGRAPH 2007]

Blendshape-based animations

Main Idea

- Similar to skeleton-based model but for faces
  - Instead of “pieces”, we have a set of example poses called blendshapes $A_0, \dots, A_0 + A_i, \dots, A_0 + A_n$
  - A_0 is the neutral rest pose
  - The final result is a linear combination of the examples

$$F = \sum_i \alpha_i A_i$$



Standard Use in Animation

- Manually define blendshapes
- Define a motion using keyframes that are interpolated

Extension

- Automatic computations
  - Analyze possible shape variation of face based on database of 3D scans of population
  - Compute statistical models of shape variation due to change of identity and/or motion
  - Use to extract blendshapes A_i
- Active area of research

Data-driven animation

1. Motion capture – based on sparse keypoints
2. Fitting to dynamic scan data – based on dense 4D scans

Motion capture

- Traditional approach used throughout industry today
- Based on a sparse set of keypoints recorded over time
- Static model is fitted to this

Motion capture
A well suited method for humans

- Capture of markers on an actor
  - Magnetic or optic: set of synchronized cameras
  - Difficulty: reconstruct motion despite of occlusions
- Replay similar motion
  - Curves of angle values over time

Examples of use:

- Feature films
- Sport video games (library of typical motion)

Motion capture

Problems to be solved

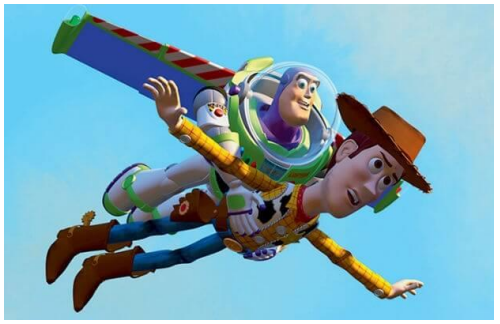
To make it generally applicable

- Adapt to different morphologies
  - monsters, aliens…
- Combining different motions
  - walk while raising arms
  - motion graphs for transitions (walk, fall, get up, run….)
- Editing at various levels of detail
  - walk on uneven ground

Physics-Based Animation & Collisions

Reminder: Descriptive animation

- Describes a single motion with manual control
  Ex: direct kinematics with key-frames, inverse kinematics



- ***Advantage:***
  - Skilled artist can create what they want
- ***Problems:***
  - Defining a new motion is very tedious
  - The user gets no help towards realism
  - Objects may collide and intersect each other, etc.

Towards methods that **generate motion** ?

- The user defines the **laws of motion**

  Examples :

  - Physical laws (gravity, collisions…)
  - Behavioral laws (artificial intelligence)

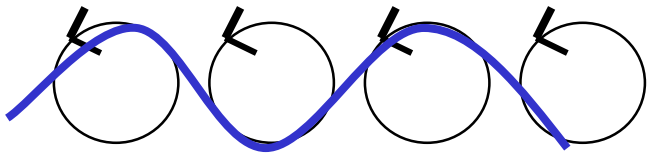
- The system generates motion from
  - The initial conditions
  - The laws to be applied

« Procedural animation »

- Describes a family of motions
- Indirect control

Procedural animation : Examples

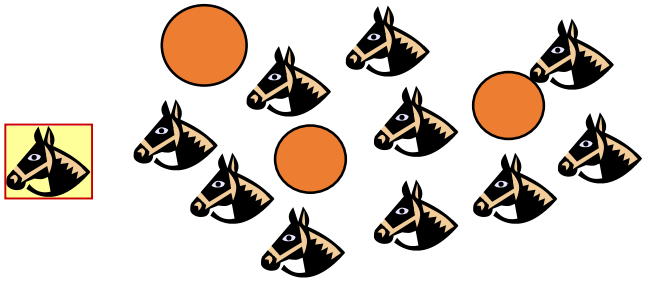
- Procedural virtual ocean



- Particle systems

(fire, smoke, rain, bees, fishes…)

  - Points : X (x,y,z), V (v_x, v_y, v_z)
  - V given by a “law”
  - Birth and death of particles



Physically-based models

Laws of motion from mechanics

- Model (mass etc) + initial conditions + applied forces
  $\rightarrow$ Motion & deformations

Advantage: a help towards realism!

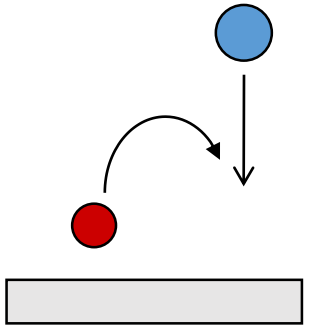
- useful when dynamics plays an important part
- easier for passive models!

Examples :

- *Toy-Story, Shrek*

Physically-based models

Standard animation algorithm

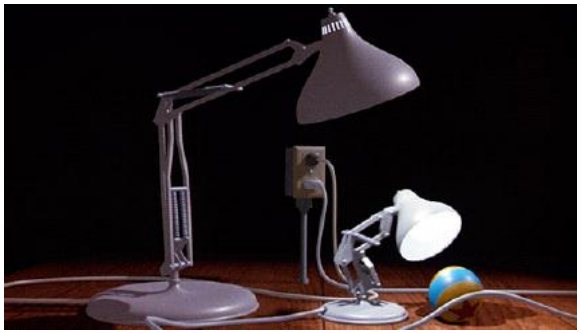


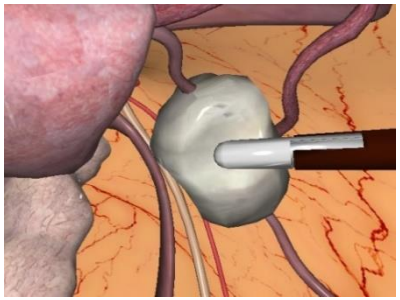
- Loop: $t := t+\Delta t$
  - For each object
    1. Compute new speed (use law & applied forces)
    2. Compute new position & deformation
    3. Display
  - For each pair of objects
    1. Detect collisions
    2. Compute new applied forces

Exercise:
- Where is the approximation?
- Can you improve the loop?

Physically-based models

Which laws of motion do we need?





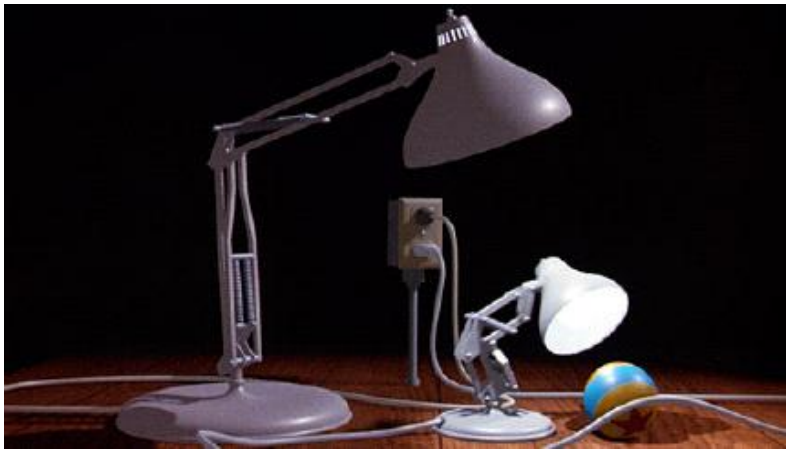
We need…

Rigid bodies

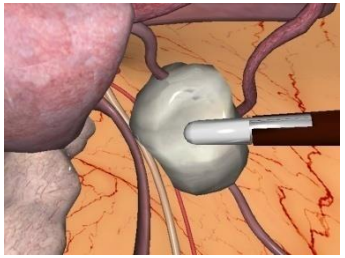
- Solids
- Articulated solids

Ex:

- Rolling ball?
- Lamps?
- Wire?



We need…
Deformable bodies



Structured

- Elasticity
  - Deformation under forces
  - Back to equilibrium
- Visco-elasticity
  - Speed of deformation
- Fractures
  - If distortion is too large

Ex : ball, organ, cloth, paper…

Un-structured

- Neighbors change!

- Plasticity
  - Absorbs deformations

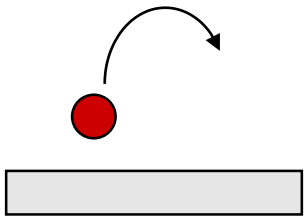
- Fluids
  - Navier-Stokes

Ex : mud, clay, liquids, smoke...

Main motions laws
used in Computer Graphics

Point-based physics

- Model *[m, X, V]*
- Law: $F = \sum Forces = m\,A = m\,dV/dt$

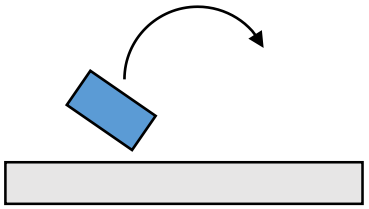


Solid physics

- Model *[m, I* inertia matrix*, X, V,* ω angular speed *]*
- Laws: $\sum F = m\,(dV/dt)$

$$\sum M = I\,(d\omega/dt) + \omega \wedge I\,\omega$$

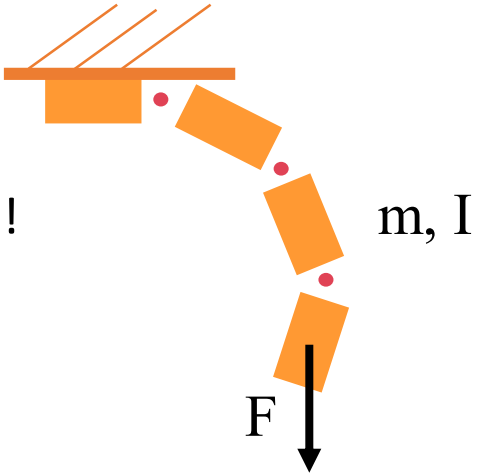
Difficulty: representation of orientations!



Main motions laws
used in Computer Graphics

Articulated solids

- Solid dynamics + unknown internal forces at joints!
  (Lagrange multipliers..)



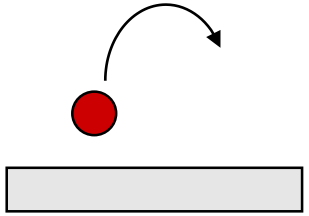
Deformable models

- Linear & non-linear elasticity, plasticity
- Navier-Stokes for fluids

NB: Eulerian vs Lagrangian discretization

This lecture:
Do it **all** with point-based physics!

- Physically-based model: Particles *[m, X, V]*
- Motion law : $\sum Forces = m\,A$



Animation algorithm

- At each time step, for each particle

$$V(t+dt) = V(t) + \sum F(t)/m \; dt$$

$$X(t+dt) = X(t) + V(t)\, dt$$

Integration:
- *Explicit Euler : may diverge!*
- *Implicit integration (next year)*

- From a model to another one
  - Choose the appropriate forces
  - Render with adapted geometry!

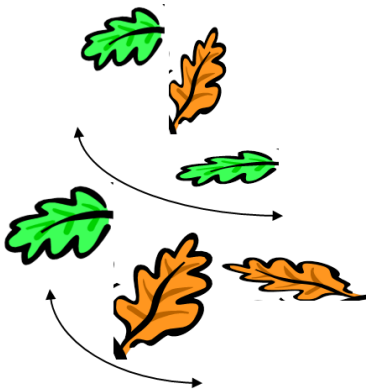
Do it all with point-based physics?

Lots of simple objects

Physically-based **particle systems**

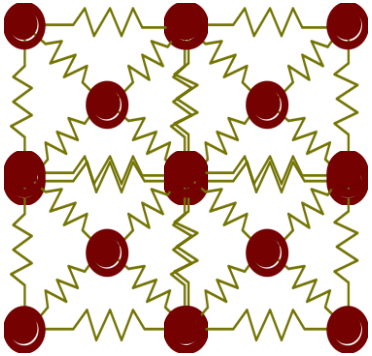
- Example : gravels, cereals
  - Gravity
  - Spheres for collisions detection
  - Random individual geometry

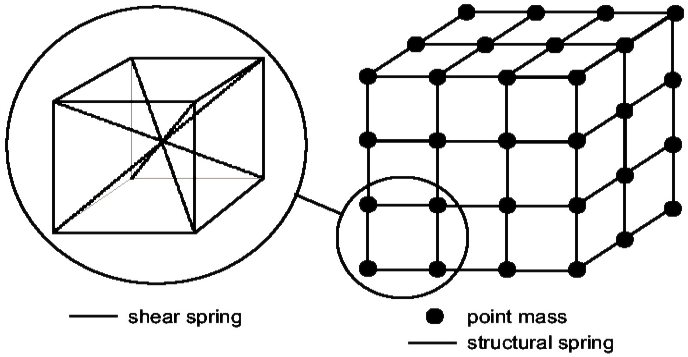
- Example: animating autumn leaves
  - Leaves = particle + local frame
  - Wind primitives
  - Gravity
  - Friction force strong in normal direction, weak in tangential one



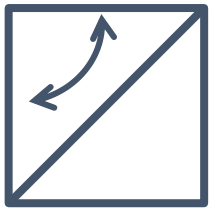
Do it all with point-based physics?
Structured deformable bodies

1D, 2D, 3D mass-spring networks





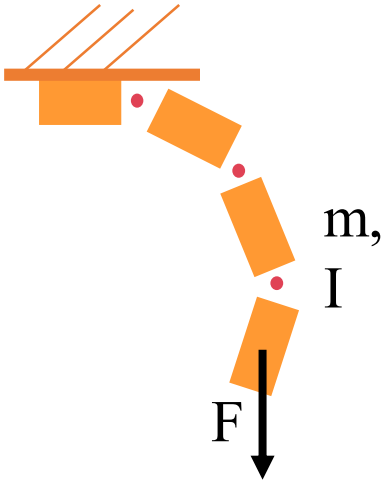
- Spring: $F = k (x - x_0)$ (where x is length)
- Angular spring: $F = k (\alpha - \alpha_0)$
- Damping: or air friction: $F = - \lambda v$

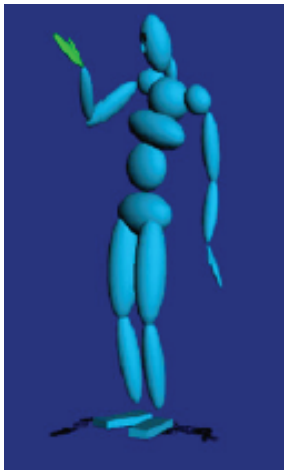


Do it all with point-based physics?
Articulated solids

- Articulated solids?

Joint = spring of zero length



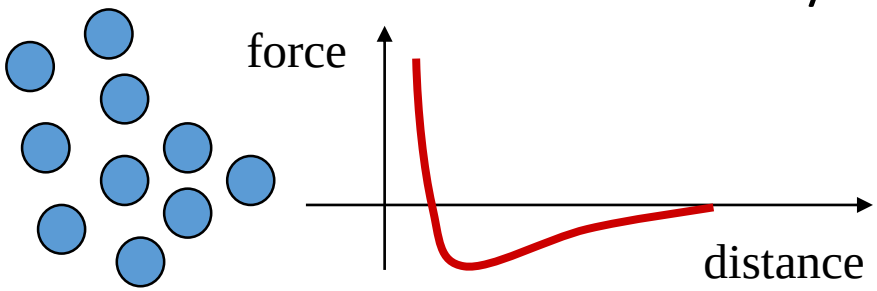


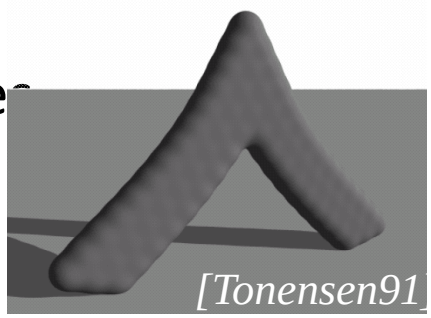
Exercise :

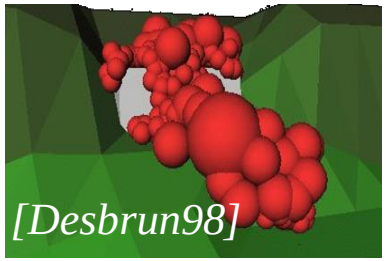
- How would you model the solids within an articulated solid with springs? (to enable rotation)
- Drawbacks compared to more accurate physics?

Do it all with point-based physics?
Unstructured objects

- Particle systems inspired from molecular dynamics
  - Lennard-Jones attraction/repulsion force

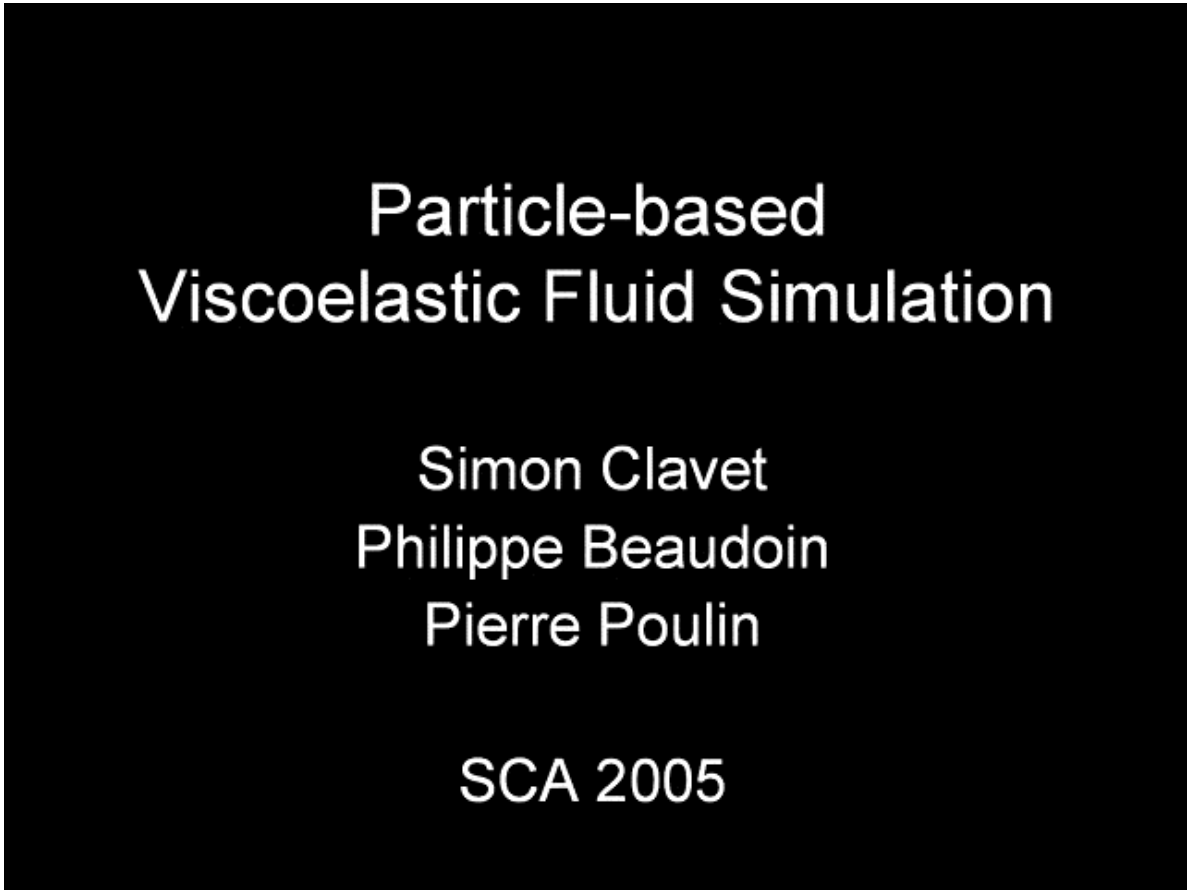






Do it all with point-based physics?

Unstructured objects [Clavet, Beaudoin, Poulin, SCA'2005]



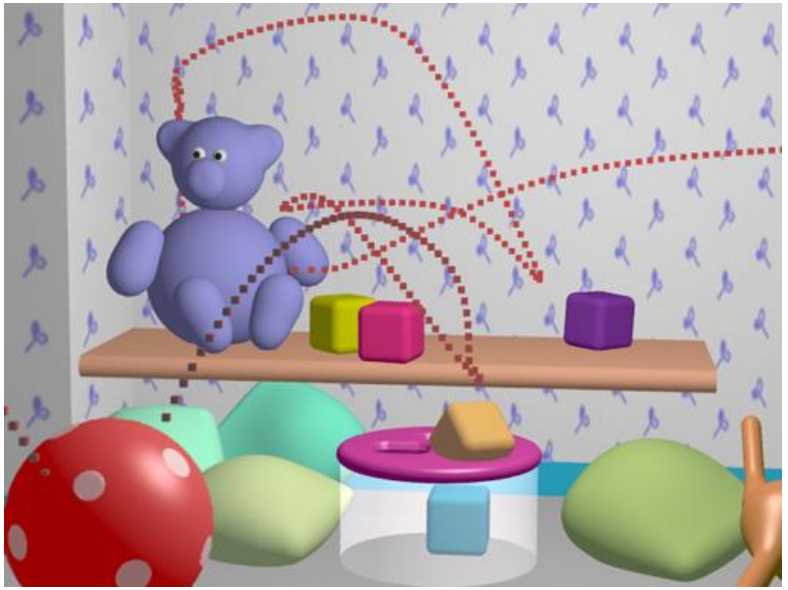
Collisions

Physically-based models

Interactions (collisions) between objects

Processing them: an advantage of physically-based models!

- Continuous solutions
  - Intersections of trajectories
  - Back to the contact time!
- Discrete time solutions
  1. Detect interpenetrations
  2. Model contact
  3. Respond to collisions



Physically-based models

Interactions between objects

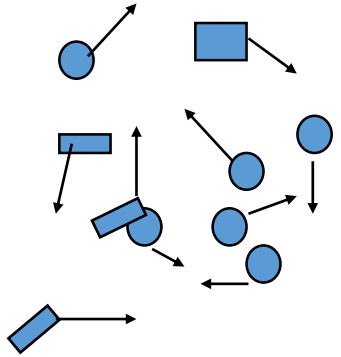
1. Detect interpenetrations

- Broad phase
  - Goal: eliminate quickly couples of objects that cannot intersect
- Narrow phase
  - Intersection of geometry

Physically-based models

Interactions between objects

Broad phase of collision detection



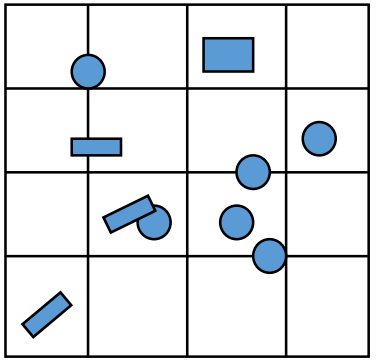
Method 1: Event-based detection

- Guarantee that a pair cannot collide before …
- Use a temporal queue to store the next tests
  - Only works for rigid solids with bounded acceleration

Physically-based models

Interactions between objects

Broad phase of collision detection



Method 2: Space grid

- Each cell store list of objects intersecting it
- Tests: pairs of objects in the same cell

Exercise:

- *Good choice of cell size compared to size of objects?*
- *Propose a solution with a grid of adaptive size*

Physically-based models

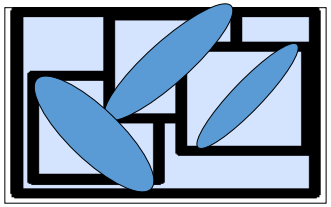
Interactions between objects

Broad phase of collision detection

Method 3: Bounding volumes

- Spheres

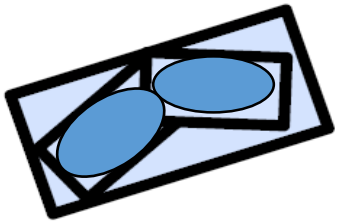
  distance < $R_1 + R_2$?



- Axis-aligned bounding boxes (AABB)

  (X_1-*max* > X_2-*min*) & (Y_1-*max* > Y_2-*min*) & (Z_1-*max* > Z_2-*min*)

- Oriented bounding boxes (OBB)

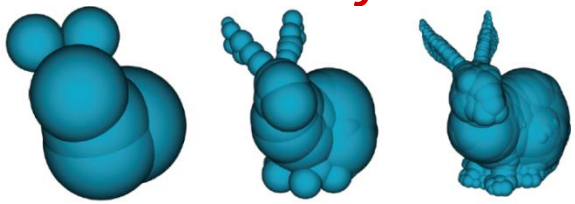


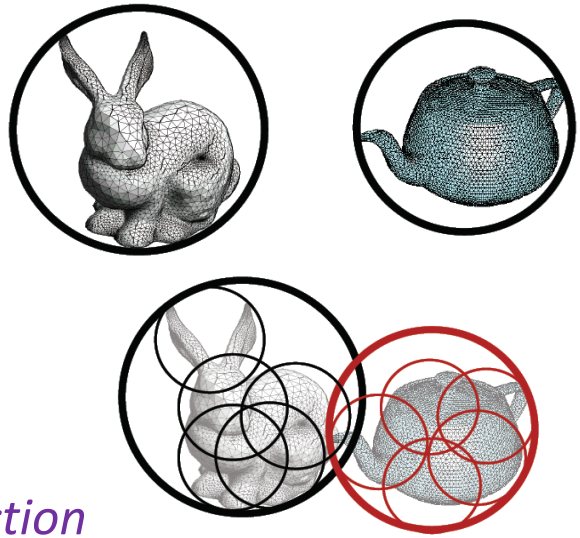
Physically-based models

Interactions between objects

Broad phase of collision detection

Method 4: Hierarchies of bounding volumes





- Detection: Divide & conquer approach
  - Refine if the parent volumes intersect

Exercise: *Write the animation loop*

Extend to constant time, approximate detection

Physically-based models

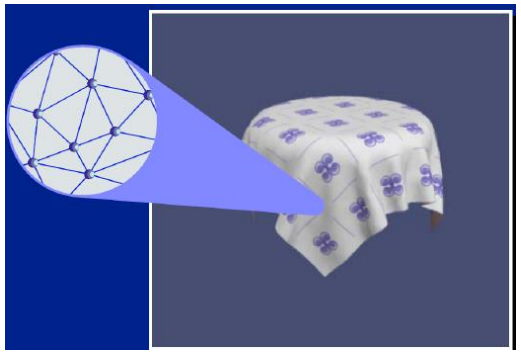
Interactions between objects

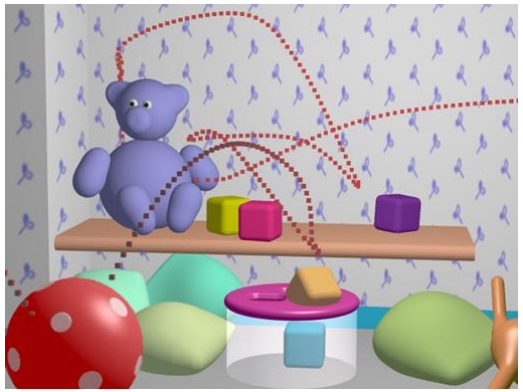
Narrow phase = test object geometry

- For each pair of object

  Use the geometric description

  - Polygonal models: intersection between pairs of faces (O(n^2))





Done only for faces of the other object lying in the bounding volume of the first one!

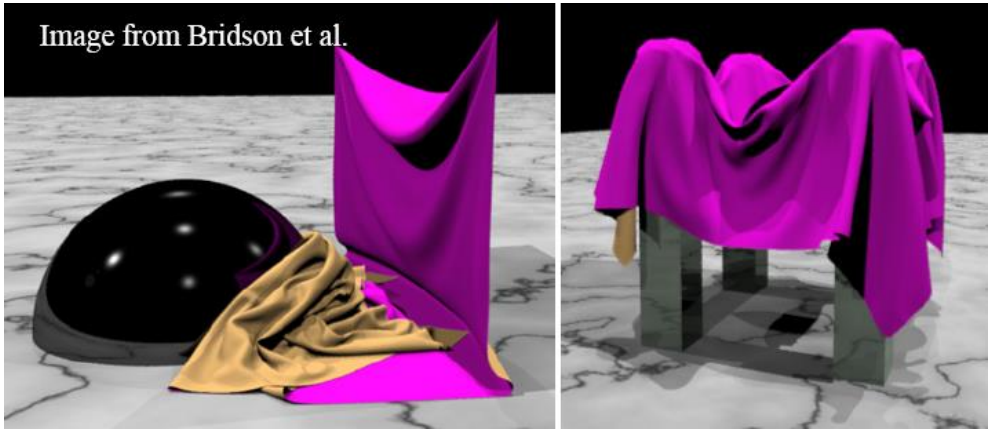
Physically-based models

Interactions between objects

Narrow phase: remarks

- Many recent methods based on the graphics hardware (GPU)
- Thin objects: Some collisions can be missed

Untangling cloth *[Baraff03]*

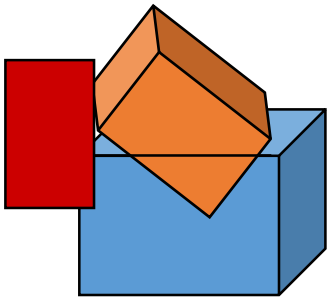


Physically-based models

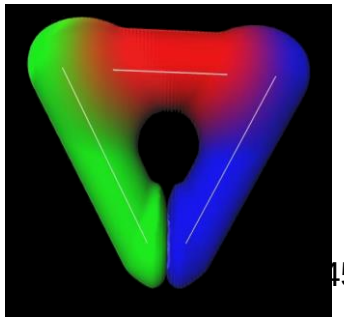
Interactions between objects

Step 1: Collision detection

Step 2: Contact modeling ?



- Rigid objects
  - Back to a « valid configuration »?
    - Inequalities expressing non-penetration
    - Global system to be solved
  - Virtual reality: fast solution for a single collision
    - Display a non-penetrating proxy (god-object)
- Deformable models
  - Deform objects without moving them?



Physically-based models

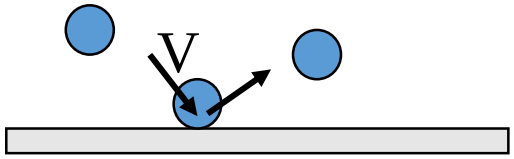
Interactions between objects

Step 3: Response to collisions

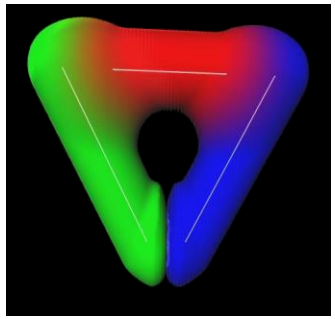
- Rigid bodies : 2 possible solutions
  - **Impulses**

    $V = V_t + V_n$

    Modified speed: $V := V_t - k\, V_n$

    (mirror with energy decay in normal direction)
  - **Contact forces**



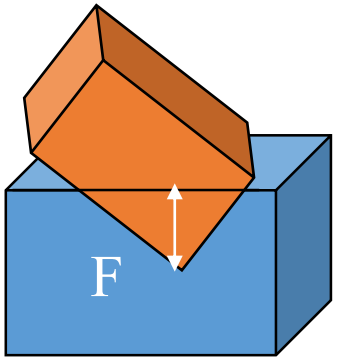
- Soft objects
  - **Contact forces**

Physically-based models

Interactions between objects

Step 3: Response to collisions

- "Penalty method" for response forces
  - Normal force function of penetration
  - + Friction forces (viscous, dry…)



Virtual spring

$F = kl$

Deep penetration: overshooting problem!

- Go back in time?
- Project the object to the closest point?
- Control energy after bouncing?
- Use adaptive time-steps?