Computer Animation and Visualisation

Lecture 8

Physics-based Animation

Floyd M. Chitalu

University of Edinburgh

Me

- Floyd M. Chitalu
  - Contact: *floyd.m.chitalu <at> ed.ac.uk (Room INF.G12)*
  - Research Topics (PhD): *Physics-based animation + collision detection + GPU computing ...*
  - B.Sc. in Games Tech, Univ of West of Scotland (2010-15)
  - M.Sc. in CompSci, Edinburgh Univ. (2015-16)
  - Ph.D. Student, Edinburgh Univ. (2016-19)
  - Edinburgh Univ. PostDoc. (2020-now)

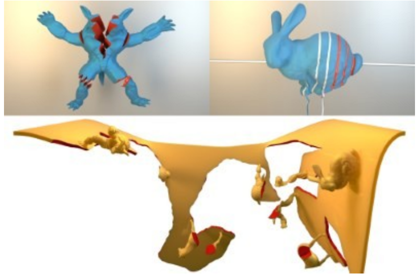
Lecture Overview

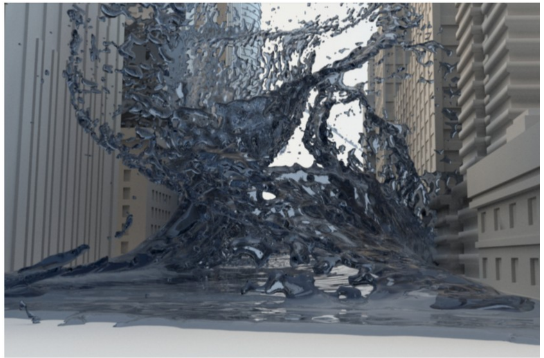
- Introduction to “Physics-based Animation”
- Particle Dynamics
- Modelling Materials

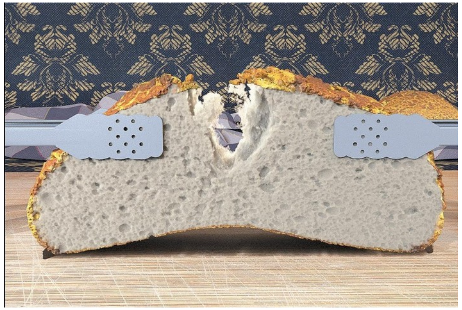
Lecture Overview

- Introduction to “Physics-based Animation”
- Particle Dynamics
- Modelling Materials

Intro. to “Physics-based Animation”







Intro. to “Physics-based Animation”

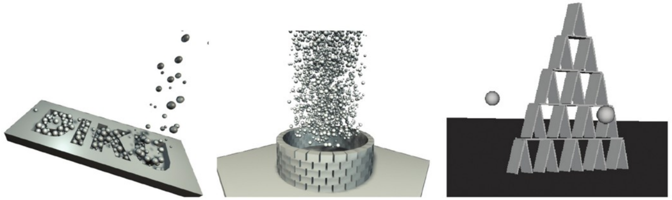
- Real-time vs. Off-line physics
- Introduction to Solids

Off-line vs. Real-time physics

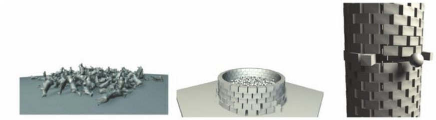
Off-line vs. Real-time physics

- In off-line physical simulation, the main concern is visual quality
  - Computational *efficiency* is important because simulations are in high resolution.
  - Visual *quality* of the output is more important than performance.
  - Require's powerful computers which work for hours.
- Off-line simulations are also *predictable*
  - it is possible to re-run the process, adapt the time step in case of numerical instabilities or change parameters if the outcome does not meet the specifications or expectations.

Off-line vs. Real-time physics



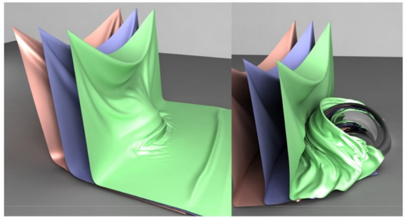
- Real-time physics is useful in *interactive systems*
  - running at a fixed frame rate (e.g. 30 or 60 FPS)
- Strict *time budget*
  - ~30 or ~15 milliseconds per frame (which are shared with e.g. AI, rendering)
  - Only a few milliseconds remain for physics.
- In contrast to off-line simulations, the outcome of interactive scenarios is *not predictable*
- Contraints of real-time physics
  - *Time*: resolution and visual quality have to be adjusted to meet time constraints!
  - *Stability*: it is essential that simulations are unconditionally stable, i.e. stable under all circumstances.

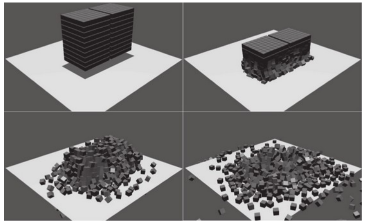


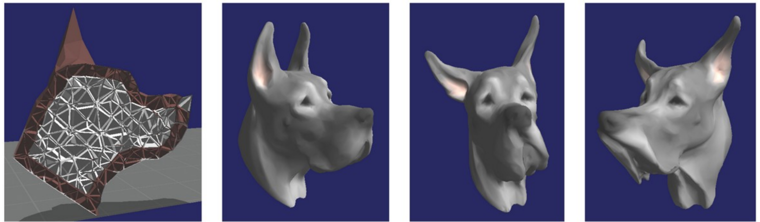
Intro. to Solids

Intro. to Solids

- Solid objects are typically divided into three main groups:
  - Cloth, Rigid bodies and Soft bodies
- Makes sense to handle those three types separately from an algorithmic and simulation point of view.
  - E.g. treating objects made of stone as infinitely rigid leads to no visual artifacts but simplifies the handling and simulation of such objects significantly.
  - For cloth, simulating it as a 2D rather than a 3D object reduces simulation time and memory consumption.





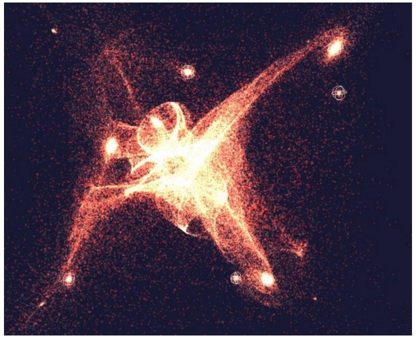
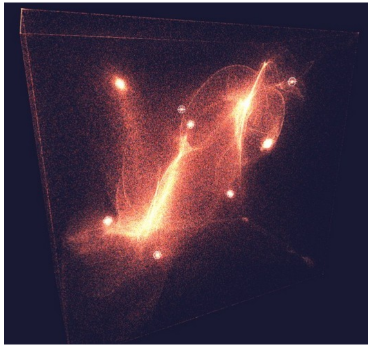
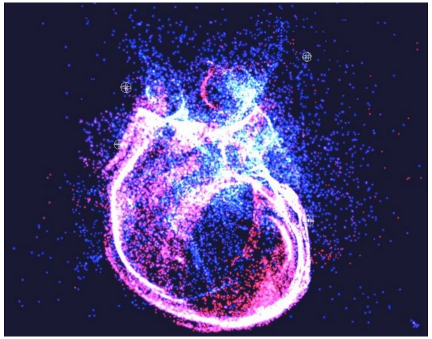


Lecture Overview

- Introduction to “Physics-based Animation”
- Particle Dynamics
- Modelling Materials

Particle Dynamics

- Particles in a velocity field
- Particles with mass
- Spring-mass systems

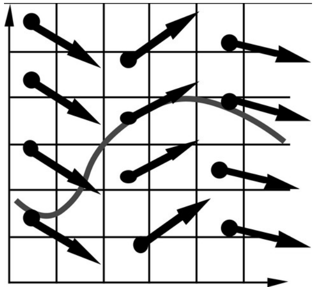





Particles in a velocity field

- Mass-less particles are advected (transferred) through velocity fields
  - Simplest particle system: we only need to track the particle’s position through time.
- A particle’s position, which varies over time, is then the solution of an *Initial Value Problem* (IVP):

$$\mathbf{x}_p(0) = \mathbf{x}_0$$

$$\frac{d\mathbf{x}_p(t)}{dt} = \dot{\mathbf{x}}_p(t) = \mathbf{v}(\mathbf{x}_p, t),$$



$$\mathbf{x}_p(0) = \mathbf{x}_0$$

$$\frac{d\mathbf{x}_p(t)}{dt} = \dot{\mathbf{x}}_p(t) = \mathbf{v}(\mathbf{x}_p, t),$$

Particles in a velocity field

- In the general case, we will need to use *numerical integration* to solve our IVP.
  - Analytical solutions exist only for very simple velocity fields
- Euler Integration
  - Simplest numerical integration method, based on continous definition of derivative

$$\frac{d\mathbf{x}_p(t)}{dt} = \lim_{\epsilon \longrightarrow 0} \frac{\mathbf{x}_p(t+\epsilon) - \mathbf{x}_p(t)}{\epsilon}.$$

$$\frac{d\mathbf{x}_p(t)}{dt} \approx \frac{\mathbf{x}_p(t+\Delta t) - \mathbf{x}_p(t)}{\Delta t}.$$

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

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

Position update rule

Particles with mass

Particles with mass

- Motion of real-world objects is governed by internal and external forces via Newton's 2nd law of motion:

$$\mathbf{f} = m\mathbf{a}.$$

- Thus to introduce forces, such as gravity, into our particle systems we must also introduce mass.
- Our IVP then becomes:

Particle position

$$\mathbf{x}_p(0) = \mathbf{x}_0$$

$$\frac{d^2\mathbf{x}_p(t)}{dt^2} = \ddot{\mathbf{x}}_p(t) = \frac{\mathbf{f}(\boxed{\mathbf{x}_p}, t)}{\boxed{m_p}}.$$

Particle mass

Particles with mass (updating the system)

- For convinience, we re-write our 2nd-order DE as a coupled system of 1st-order DEs:

$$
\mathbf{x}_p(0) = \mathbf{x}_0
$$
$$
\frac{d^2\mathbf{x}_p(t)}{dt^2} = \ddot{\mathbf{x}}_p(t) = \frac{\mathbf{f}(\mathbf{x}_p, t)}{m_p}.
$$

$\Rightarrow$

$$
\begin{aligned}
\mathbf{x}_p(0) &= \mathbf{x}_0 \\
\mathbf{v}_p(0) &= \mathbf{v}_0 \\
\frac{d\mathbf{x}_p(t)}{dt} &= \dot{\mathbf{x}}_p(t) = \mathbf{v}_p(t) \\
\frac{d\mathbf{v}_p(t)}{dt} &= \dot{\mathbf{v}}_p(t) = \frac{\mathbf{f}(\mathbf{x}_p, t)}{m_p}.
\end{aligned}
$$

- We can then (numerically) solve this IVP by integrating forward in time using Euler's method:

new velocity
$$
\boxed{\mathbf{v}_p(t + \Delta t)} = \mathbf{v}_p(t) + \Delta t \cdot \frac{\mathbf{f}(\mathbf{x}_p, t)}{m_p}
$$
new position
$$
\boxed{\mathbf{x}_p(t + \Delta t)} = \mathbf{x}_p(t) + \Delta t \cdot \mathbf{v}_p(t + \Delta t).
$$

- This particular integration scheme is commonly referred to as <u>"Symplectic Euler"</u>

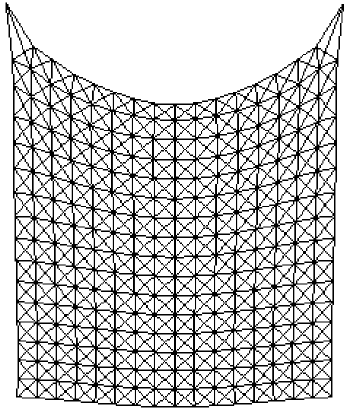
Spring-mass systems

Spring-mass systems

- So far described, the particles can be subjected to a wide range of forces, but they do not interact
  - Interaction between particles enables animation of a wide range of physical phenomena like *hair* and *cloth*

Spring-mass systems

- A mass-spring system is a set of N particles, where each particle has
  - mass m_i, position $\boldsymbol{x}_i$ and velocity $\boldsymbol{v}_i$
- Particles are connected by a set S of strings (p,q, r, k_s, k_d) where
  - p and q are indices of adjacent particles, r is the rest length k_s is the stiffness constant and k_d is the damping coefficient



- The spring forces acting on the particles are

$$\mathbf{f}_p = \Bigg[\underbrace{k_s\left(\frac{\|\mathbf{x}_q - \mathbf{x}_p\|}{r} - 1\right)}_{\text{spring}} + \underbrace{k_d\left(\frac{(\mathbf{v}_q - \mathbf{v}_p)\cdot(\mathbf{x}_q - \mathbf{x}_p)}{r\|\mathbf{x}_q - \mathbf{x}_p\|}\right)}_{\text{damping}}\Bigg]\overbrace{\frac{\mathbf{x}_q - \mathbf{x}_p}{\|\mathbf{x}_q - \mathbf{x}_p\|}}^{\text{direction}}.$$

$$\mathbf{f}_q = -\mathbf{f}_p.$$

From Newton's 3rd law

Spring-mass systems

```
for Particle p : particles do
    p.frc = 0
    p.frc += p.mass*gravity
end for
for Spring s : springs do
    Vec3 d = particles[s.j].pos - particles[s.i].pos
    double l = mag(d)
    Vec3 v = particles[s.j].vel - particles[s.i].vel
    Vec3 frc = (k_s*((l / s.r) - 1.0) + k_d*dot(v / s.r, d / l)) * (d / l)
    particles[s.i].frc += frc
    particles[s.j].frc -= frc
end for
for Particle p : particles do
    p.vel += dt*(p.frc / p.mass)
    p.pos += dt*(p.vel)
end for
```

Lecture Overview

- Introduction to “Physics-based Animation”
- Particle Dynamics
- Modelling Materials

Modelling Materials

- Rigid Bodies
- Soft Bodies

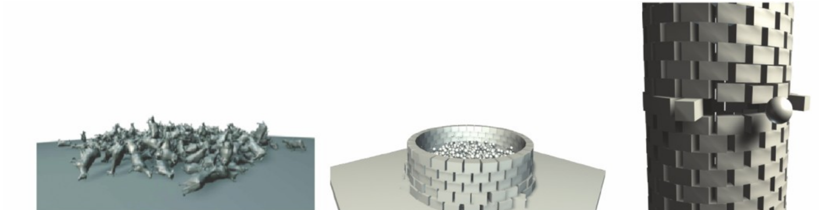
Rigid Bodies

Rigid Bodies

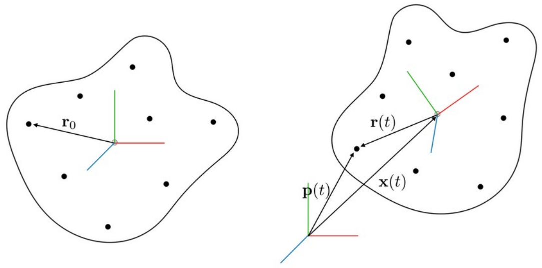
- In many circumstances, we interested in modeling <u>very stiff objects</u> for which we are not concerned with elastic deformation
- We use a rigid-body approximation since mass-spring system is relatively inefficient.
  - Points in an object are constrained to be at a fixed distance from one another.
- The position of all points on the object can be described with <u>six degrees of freedom</u>
  - *Position*: Center of mass can be at any point in three-dimensional space

$$\mathbf{x}_{com} = \frac{\sum_{i=1}^{N} m_i \mathbf{p}_i}{\sum_{i=1}^{N} m_i}.$$

  - *Rotation*: Object can be oriented in any way about that center of mass



Rigid Bodies (state & linear velocity)



(a) *Object space.* **(b)** *World space.*

- Its convinient to work with two coordinate systems to describe motion
  - A fixed "object space" transformation (located at centre of mass), and a "world space" transformation.
- The world space position of a particle at time t is given by

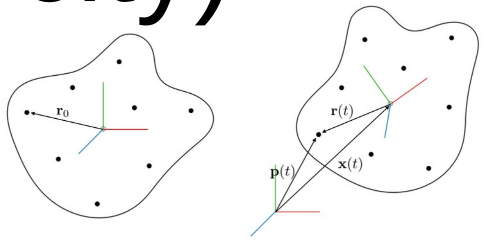
$$\mathbf{p}(t) = \mathbf{x}(t) + \boxed{\mathbf{R}(t)}\mathbf{r}_0.$$

Rotation about the centre of mass

- The *linear* velocity of a particle is found by differentiating **p**(t):

$$\mathbf{v}(t) = \dot{\mathbf{p}}(t) = \dot{\mathbf{x}}(t) + \dot{\mathbf{R}}(t)\mathbf{r}_0.$$

Rigid Bodies (Angular velocity)



- Angular velocity
  - For a rigid body that is rotating → $\dot{\mathbf{R}}(t) \neq 0.$
    - i.e. we have a non-zero component of motion of a particle (due to the instantaneous rotation of the body about its center of mass)
  - This instantaneous rotation is equivalent to a rotation about a single axis that runs through the center of mass.

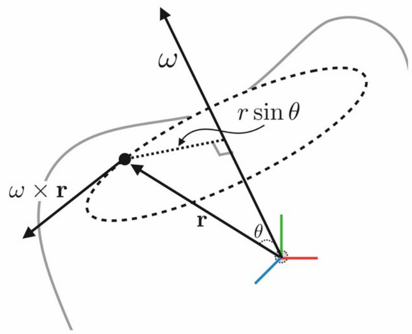


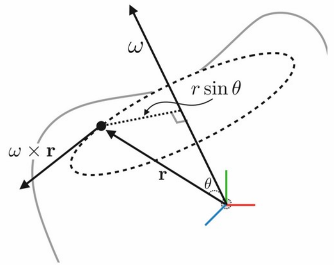
Fig. 4. *For a body with angular velocity $\boldsymbol{\omega}$, a point at $\mathbf{r}$ on the body relative to the body center of mass rotates about about the center of mass with velocity $\boldsymbol{\omega} \times \mathbf{r}$.*

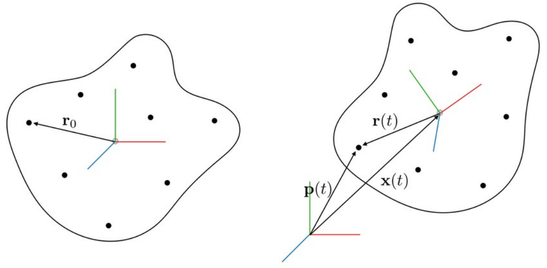
Rigid Bodies (Angular velocity)

- Thus, angular velocity is given by $\dot{\mathbf{R}}(t)\mathbf{r}_0 = \boldsymbol{\omega}(t) \times \mathbf{r}(t).$
  - … where $\mathbf{r}(t) = \mathbf{R}(t)\mathbf{r}_0$
- Velocity of particle is therefore

$$\mathbf{v}(t) = \dot{\mathbf{p}}(t) = \underbrace{\dot{\mathbf{x}}(t)}_{\text{Linear velocity}} + \underbrace{\boldsymbol{\omega}(t) \times \mathbf{r}(t)}_{\text{Angular velocity}}.$$

  - Particle rotates ‖ω‖ radians/sec on a circle of radius r*sin(θ)
  - Therefore, the <u>speed of motion</u> of the particle is ‖ω‖‖r‖*sin(θ), and <u>direction</u> of particle motion is perpendicular to ω and r.





(a) *Object space.* **(b)** *World space.*

Rigid Bodies (Linear Momentum)

- The concept of linear momentum lets us express the effect of the total force on a rigid body quite simply.
  - $\mathbf{P}(t) = M\dot{\mathbf{x}}(t).$
- Linear momentum of the rigid body is given by the sum of the momenta of its constituent particles
  - $$\mathbf{P}(t) = \sum_{i=1}^{N} m_i \mathbf{v}_i(t). \Rightarrow \mathbf{P}(t) = \sum_{i=1}^{N} m_i \left(\dot{\mathbf{x}}(t) + \boldsymbol{\omega}(t) \times \mathbf{r}_i(t)\right)$$
    $$= \sum_{i=1}^{N} m_i \dot{\mathbf{x}}(t) + \boldsymbol{\omega}(t) \times \boxed{\left(\sum_{i=1}^{N} m_i \mathbf{r}_i(t)\right)}$$

Equates to zero since $\boldsymbol{r}_i$ *are defined relative to centre of mass!*

- Thus, our linear momentum tells us nothing about the rotational velocity of a body, which is good, because force also conveys nothing about the change of rotational velocity of a body

Rigid Bodies (Angular Momentum)

- While the concept of linear momentum is pretty intuitive, angular momentum is not!
  - But we need angular momentum because it lets us write simpler equations.
- The total angular momentum is given by
  - $\mathbf{L}(t) = \mathbf{I}(t)\boldsymbol{\omega}(t).$
- Inertia tensor **I**(t) describes mass distribution:
  - $$\begin{aligned}\mathbf{I}(t) &= \sum_{i=1}^{N} m_i \boxed{\mathbf{r}_i^\star(t)} \mathbf{r}_i^{\star T}(t) \\ &= \sum_{i=1}^{N} m_i \left(\boxed{\mathbf{r}_i^T \mathbf{r}_i \boldsymbol{\delta} - \mathbf{r}_i \mathbf{r}_i^T}\right) \\ &= \mathbf{R}(t) \boxed{\sum_{i=1}^{N} m_i \left(\mathbf{r}_{0i}^T \mathbf{r}_{0i} \boldsymbol{\delta} - \mathbf{r}_{0i} \mathbf{r}_{0i}^T\right)} \mathbf{R}(t)^T \\ &= \mathbf{R}(t)\mathbf{I}_0\mathbf{R}(t)^T.\end{aligned}$$

$$\mathbf{r}^\star = \begin{pmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{pmatrix}$$

$$\mathbf{r}^\star \mathbf{r}^{\star T} = \mathbf{r}^T \mathbf{r} \boldsymbol{\delta} - \mathbf{r}\mathbf{r}^T$$

$$\mathbf{r} = (r_x, r_y, r_z)^T$$

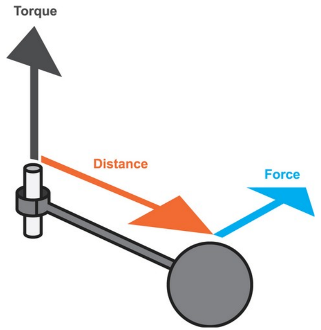
constant object-space inertia tensor

Rigid Bodies (Force and Torque)

- In the case of a rigid body, Newton's second law takes the form

$$\frac{d}{dt}\begin{pmatrix}\mathbf{P}(t)\\ \mathbf{L}(t)\end{pmatrix} = \begin{pmatrix}\mathbf{f}(t)\\ \boldsymbol{\tau}(t)\end{pmatrix}$$

- If a force **f** is applied to a rigid body at its center of mass
  - then the body responds as if it was a particle with mass *M* (its particles undergo an acceleration of $\mathbf{a} = \mathbf{f}/M$)
- When a force is applied to the body at a point other than its center of mass, this may generate a torque as well
  - $\boldsymbol{\tau} = \mathbf{r} \times \mathbf{f}.$
- Torque has magnitude $\|\boldsymbol{\tau}\| = \boxed{\|\mathbf{r}\|\|\mathbf{f}\| \sin\theta}$ *Note: two ways to increase the magnitude of torque!*
  - Where θ is angle between **r** and **f**



Rigid Bodies (simulation)

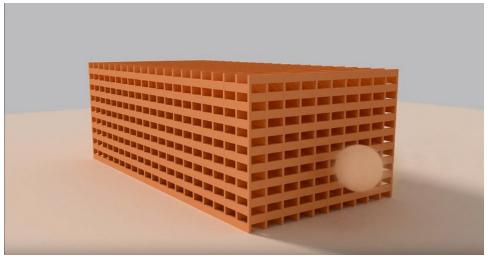
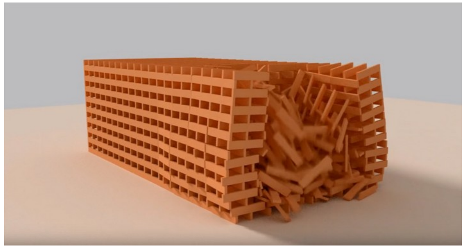
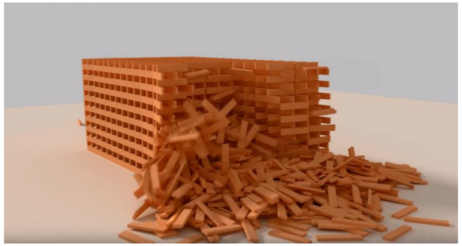
- Define the auxiliary quantities

Computed using Euler angles or quaternions

$$\mathbf{v}(t) = \frac{\mathbf{P}(t)}{M}, \qquad \mathbf{I}(t) = \boxed{\mathbf{R}(t)}\mathbf{I}_0\mathbf{R}(t)^T, \qquad \text{and} \qquad \boldsymbol{\omega}(t) = \mathbf{I}(t)^{-1}\mathbf{L}(t)$$

- Then, update rigid body state by

$$\frac{d}{dt}\begin{pmatrix}\mathbf{x}(t)\\ \mathbf{R}(t)\\ \mathbf{P}(t)\\ \mathbf{L}(t)\end{pmatrix} = \begin{pmatrix}\mathbf{v}(t)\\ \boldsymbol{\omega}^{\star}(t)\mathbf{R}(t)\\ \mathbf{f}(t)\\ \boldsymbol{\tau}(t)\end{pmatrix}$$

Soft bodies

Soft bodies (equation of motion)

- To model soft bodies we incorporate elastic and damping forces into Newton's second law

displacement *Newton's 2nd law*

$$\mathbf{K}(\boxed{\mathbf{d}}) + \mathbf{D}(\dot{\mathbf{d}}) + \boxed{\mathbf{M}\ddot{\mathbf{d}} = \mathbf{f}_{ext}}$$

- **K** is the stiffness matrix
  - determines the magnitude of elastic forces
  - encodes the elastic relationships between particles in the system
- **D**amping matrix
- **M**ass matrix

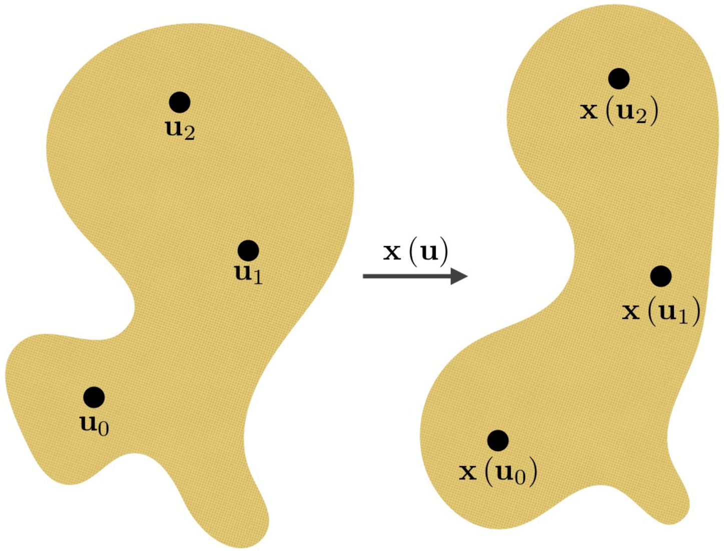
Soft Bodies (deformation gradient)

- The deformation function maps points in the rest/material space to world space.
  - This mapping may be arbitrarily complicated, but we can always linearize about a point
  - $\boxed{\mathbf{x}(\mathbf{u})} = \mathbf{x}(\mathbf{u}_0) + \mathbf{A}\left(\mathbf{u} - \mathbf{u}_0\right)$

Our deformation function

- The *deformation gradient* **F** is the derivative of the deformation map $\partial\mathbf{x}/\partial\mathbf{u}$
  - It describes how infinitesimal vectors/ lengths/ displacements in rest space are mapped to world space
  - It measures stretch, thus volume is conserved if

$$\det(\mathbf{F}) = 1$$



Soft bodies (strain)

- Strain is a dimension-less (or unit-less) quantity that *measures the amount of deformation*
  - Computed from the deformation gradient
- Three types of strain metrics commonly used in computer graphics
  - Green's finite strain (a.k.a Cauchy-Green strain)
  - Cauchy's infinitesimal strain
  - Co-rotated strain

Recall our spring force equation..?

Spring strain

$$\mathbf{f}_p = \left[k_s \left(\boxed{\frac{\|\mathbf{x}_q - \mathbf{x}_p\|}{r} - 1} \right) + k_d \left(\frac{(\mathbf{v}_q - \mathbf{v}_p) \cdot (\mathbf{x}_q - \mathbf{x}_p)}{r\|\mathbf{x}_q - \mathbf{x}_p\|} \right) \right] \frac{\mathbf{x}_q - \mathbf{x}_p}{\|\mathbf{x}_q - \mathbf{x}_p\|}.$$

Soft bodies (Green's finite strain)

$$\epsilon_{ij} = \frac{1}{2}\left(\frac{\partial \mathbf{x}}{\partial u_i} \cdot \frac{\partial \mathbf{x}}{\partial u_j} - \mathbf{I}\right) = \frac{1}{2}\left(\mathbf{F}^T\mathbf{F} - \mathbf{I}\right)$$

- Advantages
  - Straightforward and simple
  - Accounts for world-space rotations without artefacts, which led to widespread early adoption in computer graphics.
- Disadvantages
  - It is quadratic in positions
  - Always results in a non-constant stiffness matrix **K**, which makes various pre-computations invalid and results in significant computational cost

Soft bodies (Cauchy's infinitesimal strain)

- Derived from Green's finite strain.

Displacements component

- Let us re-write the deformation gradient as $\mathbf{F} = \mathbf{I} + \boxed{\mathbf{D}}$. Then,

$$\begin{aligned} \epsilon &= \frac{1}{2}\left((\mathbf{I}+\mathbf{D})^T(\mathbf{I}+\mathbf{D}) - \mathbf{I}\right) \\ &= \frac{1}{2}\left(\mathbf{I}^T\mathbf{I} + \mathbf{D}^T + \mathbf{D} + \mathbf{D}^T\mathbf{D} - \mathbf{I}\right) \\ &= \frac{1}{2}\left(\mathbf{D}^T + \mathbf{D} + \mathbf{D}^T\mathbf{D}\right) \\ &= \frac{1}{2}\left((\mathbf{D}+\mathbf{I})^T + (\mathbf{D}+\mathbf{I}) + \mathbf{D}^T\mathbf{D}\right) - \mathbf{I} \\ &= \frac{1}{2}\left(\mathbf{F}^T + \mathbf{F} + \mathbf{D}^T\mathbf{D}\right) - \mathbf{I} \end{aligned}$$

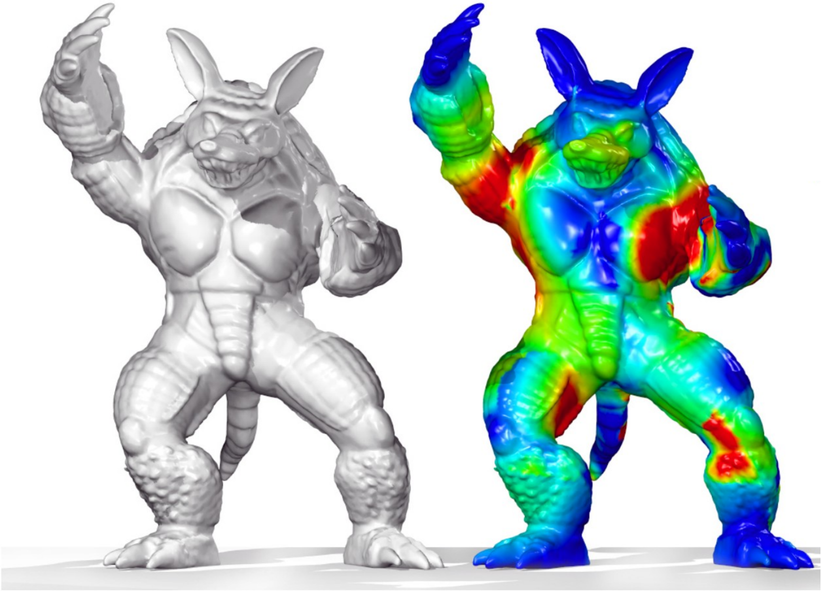
$$\Rightarrow \quad \epsilon = \frac{1}{2}\left(\mathbf{F}^T + \mathbf{F}\right) - \mathbf{I} = \frac{1}{2}\left(\mathbf{F} + \mathbf{F}^T\right) - \mathbf{I}$$

- Advantages
  - It is linear, which leads to faster computation
- Disadvantages
  - Does not correctly account for world-space rotations, leading to a variety of <u>unpleasant artifacts</u> under large deformations

Soft bodies (co-rotated strain)

- The co-rotated strain metric is the most common strain model in computer graphics.
  - Intuitively, this model is Cauchy's linear strain with the rotation explicitly removed through the polar decomposition
- Once the deformation gradient, **F**, is computed, we compute the polar decomposition
  - $\mathbf{F} = \mathbf{Q}\tilde{\mathbf{F}}$
- Then update the Cauchy's infinitesimal strain to
  - $$\epsilon = \frac{1}{2}\left(\tilde{\mathbf{F}} + \tilde{\mathbf{F}}^T\right) - \mathbf{I}$$
- Advantages
  - Accounts for world-space rotations without artefacts
  - More efficient implementation since some pre-computation is still possible
- Disadvantages
  - Requires Polar Decomposition

Soft bodies (stress)



Soft bodies (stress)

- Unlike strain, stress is not a dimension-less quantity.
- Instead of measuring the amount of deformation, it measures the materials *reaction to that deformation*.
- In graphics, we care mostly about linear material models (i.e. when relating to strain to stress)
  - $\boldsymbol{\sigma} = \mathbf{C}\boldsymbol{\epsilon}$
- By making a few assumptions we arrive at
  - $\boldsymbol{\sigma} = \lambda \mathrm{Tr}(\epsilon)\mathbf{I} + 2\mu\epsilon$

Summary

- Brief intro. to physics-based animation
  - Off-line and real-time simulation
- Particles
  - Velocity fields, Newton’s laws of motions, mass-spring systems
- Materials
  - Rigid body dynamics
    - Linear and angular velocity, momentum, inertia.
  - Soft body dynamics
    - Elasticity, deformation gradient, strain, stress.

Readings

- 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
- Erwin Coumans. 2010. Bullet physics engine. Open Source Software: http://bulletphysics. Org 1 (2010)
- Rick Parent. 2012. Computer Animation: Algorithms and Techniques (3rd. ed.). Morgan Kaufmann Publishers Inc.