COL781 guest lecture:

Physics-Based Animation

Prof. Rahul Narain
About me

• Rahul Narain
  http://rahul.narain.name/
  narain@cse.iitd.ac.in

• Assistant professor in CS&E
  Bharti IIA-517

• Research interests:
  computer graphics, physics-based animation, numerical methods
Turbulent fluids and granular materials

Crowd simulation

Cloth modeling

Parallel algorithms for interactive simulation
Basic types of animation

- **Manual**
  - e.g. keyframing
- **Recorded**
  - e.g. motion capture
- **Algorithmic**
  - e.g. physics-based
Basic types of animation

- **Manual**
  - e.g. keyframing
- **Recorded**
  - e.g. motion capture
- **Algorithmic**
  - e.g. physics-based
Physics-based animation
(a.k.a. simulation)
Physics-based animation
(a.k.a. simulation)

Subtopics:
• Rigid bodies
• Collisions and contact
• Deformable objects
• Fluids (smoke, water, fire)
Today’s agenda

• Basic principles of physics-based animation
• Mass-spring systems
• Preview of other techniques
Discretization

How to represent the continuous motion of an object?

Motion of a particle:
\[ \mathbf{x} : \mathbb{R} \to \mathbb{R}^3 \]

Flow field of a fluid:
\[ \mathbf{v} : \mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}^3 \]
Discretization

**Discretization of space**
- particles
- grids
- meshes

**Discretization of time**
- Time stepping
Discretization

Objects represented as collections of particles

• $i$th particle has mass $m_i$, position $\mathbf{x}_i$, velocity $\mathbf{v}_i$

Time divided into discrete time steps

• Usually $t_0, t_1 = t_0 + \Delta t, t_2 = t_0 + 2\Delta t, \ldots$

Given positions, velocities at $t_n$, compute positions, velocities at $t_{n+1}$
Equations of motion

Newton’s second law: $\mathbf{f} = m\mathbf{a}$

Or, for each particle: 
\[
\frac{d^2 \mathbf{x}_i}{dt^2} = \frac{\mathbf{f}_i}{m_i}
\]

$\mathbf{f}_i =$ total force on particle $i$
(including forces from environment $\textit{and}$ from other particles)

Two questions:

• How to define $\mathbf{f}_i$?

• How to solve $d^2 \mathbf{x}_i/dt^2 = \ldots$?
Recap: Single-particle dynamics

Consider a point mass attached to a fixed point via a spring

- Total force = \( mg - k_s(\ell - \ell_0)e - k_d(v \cdot e)e + f_{user} \)
- Call this \( f(x(t), v(t)) \)

\[
dv/dt = f/m \\
x' = v
\]

Simplest time stepping scheme (forward Euler):

\[
\begin{align*}
v^{\text{new}} &= v + f(x, v)/m \Delta t \\
x^{\text{new}} &= x + v \Delta t
\end{align*}
\]
Recap: Single-particle dynamics

What you covered in an earlier class is actually

\[ \mathbf{v}^{\text{new}} = \mathbf{v} + \mathbf{f}(\mathbf{x}, \mathbf{v})/m \Delta t \]

\[ \mathbf{x}^{\text{new}} = \mathbf{x} + \mathbf{v}^{\text{new}} \Delta t . \]

This is sometimes called “symplectic Euler”, actually much better than forward Euler.
Multi-particle dynamics

For each particle,
\[
\frac{dv_i}{dt} = \frac{f_i}{m_i},
\]
\[
\frac{dx_i}{dt} = v_i.
\]

Let's define \( x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \end{bmatrix} \) and similarly \( v \) and \( f \).

Then we can just write
\[
\frac{dv}{dt} = M^{-1}f
\]
\[
\frac{dx}{dt} = v
\]

where \( M^{-1} = \text{diag}(I/m_1, I/m_2, \ldots) \).
Time stepping

Can still apply the same scheme:

\[ \mathbf{v}^{\text{new}} = \mathbf{v} + \mathbf{M}^{-1} \mathbf{f}(\mathbf{x}, \mathbf{v})/m \Delta t \]

\[ \mathbf{x}^{\text{new}} = \mathbf{x} + \mathbf{v}^{\text{new}} \Delta t \]

Algorithm:

- First compute \( \mathbf{f} = \text{total force on each particle} \)
- For each particle, update \( \mathbf{v} \), then update \( \mathbf{x} \)
Mass-spring systems

Collection of particles with springs connecting them
Mass-spring cloth

Simulate as particles but render as smooth surface
Mass-spring cloth

Create grid of particles
Add springs to model internal forces

structural springs
Mass-spring cloth

Create grid of particles
Add springs to model internal forces

shear springs
Mass-spring cloth

Create grid of particles
Add springs to model internal forces

bending springs
Mass-spring cloth

Create grid of particles
Add springs to model internal forces

bending springs
Other applications
Other applications
Advanced topics

- Collision handling
- Time integration schemes
- Realistic materials using the finite element method
- Fluids
Collision handling

**Collision detection:** check if objects intersect, if so return collision point and normal

- Discrete vs. continuous

**Collision response:** apply normal & frictional forces to fix it

- Penalty forces / collision impulses / more complex algorithms
Time integration schemes

Symplectic Euler has a “speed limit”: If $\Delta t$ or spring const too large, solution blows up!

Unconditionally stable method: *backward Euler*:

$$v^{\text{new}} = v + f(x^{\text{new}}, v^{\text{new}})/m \Delta t$$

$$x^{\text{new}} = x + v^{\text{new}} \Delta t$$

Requires solving a system of equations! (Newton’s method)

Analysis of time stepping schemes:

- Accuracy vs. stability vs. conservation vs. compute cost
Finite elements

Instead of springs, divide object into collection of volumetric “elements” (triangles in 2D, tetrahedra in 3D)

• Spring rest length $\rightarrow$ element rest shape

• Spring constant $\rightarrow$ stress-strain relationship
**Fluids**

**Particles**
- Mass, velocity, pressure, etc. stored on particles
- Particles move with fluid

**Grids**
- Velocity, pressure, etc. stored on grid vertices / cells
- Grid doesn’t move

**Forces:** pressure, viscosity, surface tension, …
Online notes for further reading

Foundations:

• Witkin and Baraff, “Physically Based Modeling”
  http://www.pixar.com/companyinfo/research/pbm2001/

Finite elements for animation:

• Sifakis and Barbič, “FEM Simulation of 3D Deformable Solids”
  http://www.femdefo.org/

Fluid simulation:

• Bridson and Müller-Fischer, “Fluid Simulation for Computer Animation”
  https://www.cs.ubc.ca/~rbridson/fluidsimulation/
Course next semester

COL865: Special Topics in Computer Applications
Prof. Rahul Narain / 3 credits (3-0-0)