Particle Systems
and ODE Solvers II,
Mass-Spring Modeling

Lots of slides from Frédo Durand
Administrivia

- No class next Tuesday (Monday schedule!)

- Quiz will be on Tue Oct 20 in class
  - Everything that we’ve covered up until then
  - You can bring one letter-size sheet of notes, 2-sided
    - Can use font size as small as you want
  - Previous years’ quizzes will be available on Stellar soon
    - Don’t panic, the material might’ve been slightly different

- Monday is a holiday
  - Poll: Next office hours on Monday or Tuesday?
Plan

- Finish treatment of forces from Tuesday
- Why Euler’s method is not great
- More accurate ODE solvers
- Mass-spring modeling, cloth
- Particle system implementation notes
What is a Force?

• A force changes the motion of the system

• Forces can depend on location, time, velocity
  – Gravity, spring, viscosity, wind, etc.

• For point masses, forces are vectors
Types of Forces

• Gravity
  – On Earth, just constant acceleration ($g$)
  – In the N-body problem, every particle affects every other

• Viscous damping
  – $F = -kv$
  – Force opposes motion
  – Useful for removing energy from the system

• Spatial fields
  – Either heuristic/design (procedural wind, etc.)...
  – ...Or from a simulation
Spatial interaction: \( f^{(i)} = \sum_{j} f(x^{(i)}, x^{(j)}) \)

- E.g., approximate fluid using Lennard-Jones force:
  \[
  f(x^{(i)}, x^{(j)}) = \frac{k_1}{|x^{(i)} - x^{(j)}|^m} - \frac{k_2}{|x^{(i)} - x^{(j)}|^n}
  \]
- Repulsive + attractive force
- Again, \( O(N^2) \) to test all pairs
  - usually only local
  - Use buckets to optimize. Cf. 6.839

Particles are not independent!
Demo: Lennard-Jones

• Real-time particle system written in CUDA

• (May not run on external display)
More Advanced “Forces”

- Flocking birds, fish shoals
  - http://www.red3d.com/cwr/boids/
- Crowds (www.massivesoftware.com)
Flocks ("Boids")

- From Craig Reynolds
- Each bird modeled as a complex particle ("boid")
- A set of forces control its behavior
- Based on location of other birds and control forces
Flocks ("Boids")

- ("Boid" was an abbreviation of "birdoid". His rules applied equally to simulated flocking birds, and shoaling fish.)
Flocks ("Boids")

COURSE: 07
COURSE ORGANIZER: DEMETRI TERZOPoulos

"BOIDS DEMOS"
CRAIG REYNOLDS
SILICON STUDIOS, MS 3L-980
2011 NORTH SHORELINE BLVD.
MOUNTAIN VIEW, CA 94039-7311
Questions?
Where do particles come from?

- Often created by generators or emitters
  - Can be attached to objects in the model
- Given rate of creation: particles/second
  - record $t_{last}$ of last particle created
  - $n = \left\lfloor (t - t_{last}) \times \text{rate} \right\rfloor$
    - create $n$ particles.
    - update $t_{last}$ if $n > 0$
- Create with (random) distribution of initial $x$ and $v$
  - if creating $n > 1$ particles at once, spread out on path

http://www.particlesystems.org/
In production tools, all these variables are time-varying and controllable by the user (artist)

- Emission rate, color, velocity distribution, direction spread, textures, etc. etc.
  - All as a function of time!
- Example: ParticleFX (Max Payne Particle Editor)
  - Custom editor software
  - You can download it (for Windows) and easily create your own particle systems. Comes with examples!
  - This is what we used for all the particles in the game!
Emitter Controls

- Again, reuse splines!
Emitter Controls

- Again, reuse splines!

![Graphs showing Emission Rate, Lifetime, and Size controls for Large Flaming with control points and labels.](image-url)
Rendering and Motion Blur

- Often not shaded (just emission, think sparks)
  - But realistic non-emissive particles needs shadows, etc.
- Most often, particles don’t contribute to the z-buffer, i.e., they do not fully occlude stuff that’s behind
  - Rendered with z testing on
    (particles get occluded by solid stuff)
- Draw a line for motion blur
  - (x, x+v dt)
  - Or an elongated quad with texture
Rendering and Motion Blur

Metal Gear Solid by Konami
• Often use texture maps (fire, clouds, smoke puffs)
  – Called “billboards” or “sprites”
  – Always parallel to image plane

Futuremark Corp., used with permission
Image from Sameboat
Star Trek 2 – The Wrath of Khan

- One of the earliest particle systems (from 1982)
- Also, fractal landscapes

- Described in [Reeves, 1983]

Paramount Pictures
Particle Modeling [Reeves 1983]

- The grass is made of particles
  - The entire lifetime of the particle is drawn at once.
  - This can be done procedurally on the GPU these days!
Questions?

Early particle fun by Karl Sims
ODEs and numerical integration

\[ \frac{dX(t)}{dt} = f(X(t), t) \]

- Given a function \( f(X, t) \) compute \( X(t) \)
- Typically, *initial value problems*:
  - Given values \( X(t_0) = X_0 \)
  - Find values \( X(t) \) for \( t > t_0 \)

- We can use lots of standard tools
Reduction to 1st Order

• Point mass: 2nd order ODE

\[ \vec{F} = m \ddot{x} \quad \text{or} \quad \vec{F} = m \frac{d^2 \vec{x}}{dt^2} \]

• Corresponds to system of first order ODEs

\[
\begin{align*}
\frac{d}{dt} \vec{x} &= \vec{v} \\
\frac{d}{dt} \vec{v} &= \vec{F} / m
\end{align*}
\]

2 unknowns (x, v) instead of just x
ODE: Path Through a Vector Field

• $X(t)$: path in multidimensional phase space

\[
\frac{d}{dt} X = f(X, t)
\]

“When we are at state $X$ at time $t$, where will $X$ be after an infinitely small time interval $dt$?”

• $f = \frac{d}{dt} X$ is a vector that sits at each point in phase space, pointing the direction.
Euler, Visually

\[ \frac{d}{dt} X = f(X, t) \]
Euler’s method: Inaccurate

- Moves along tangent; can leave solution curve, e.g.:
  \[ f(X, t) = \begin{pmatrix} -y \\ x \end{pmatrix} \]

- Exact solution is circle:
  \[ X(t) = \begin{pmatrix} r \cos(t+k) \\ r \sin(t+k) \end{pmatrix} \]

- Euler spirals outward no matter how small \( h \) is
  - will just diverge more slowly
Questions?
Euler’s method: Not Always Stable

• “Test equation” \( f(x, t) = -kx \)
Euler’s method: Not Always Stable

• “Test equation” \( f(x, t) = -k x \)

• Exact solution is a decaying exponential:

\[
x(t) = x_0 e^{-kt}
\]
Euler’s method: Not Always Stable

• “Test equation” \( f(x, t) = -kx \)

• Exact solution is a decaying exponential:
  \[
x(t) = x_0 e^{-kt}
  \]

• Let’s apply Euler’s method:
  \[
x_{t+h} = x_t + h f(x_t, t) \\
  = x_t - hkx_t \\
  = (1 - hk) x_t
  \]
Euler’s method: Not Always Stable

\[ x_{t+h} = (1 - hk) x_t \]

- **Limited step size!**
  - When \( 0 \leq (1 - hk) < 1 \) \( \Leftrightarrow h < 1/k \)
    things are fine, the solution decays
  - When \(-1 \leq (1 - hk) \leq 0 \) \( \Leftrightarrow 1/k \leq h \leq 2/k \)
    we get oscillation
  - When \((1 - hk) < -1 \) \( \Leftrightarrow h > 2/k \)
    things explode!
Euler’s method: Not Always Stable

If $k$ is big, $h$ must be small!

- Limited step size!
  - When $0 \leq (1 - hk) < 1 \iff h < 1/k$
    things are fine, the solution decays
  - When $-1 \leq (1 - hk) \leq 0 \iff 1/k \leq h \leq 2/k$
    we get oscillation
  - When $(1 - hk) < -1 \iff h > 2/k$
    things explode!
Analysis: Taylor series

- Expand exact solution $X(t)$

$$X(t_0 + h) = X(t_0) + h \left( \frac{d}{dt} X(t) \right) \bigg|_{t_0} + \frac{h^2}{2!} \left( \frac{d^2}{dt^2} X(t) \right) \bigg|_{t_0} + \frac{h^3}{3!} (\cdots) + \cdots$$

- Euler’s method approximates:

$$X(t_0 + h) = X_0 + hf(X_0, t_0) \quad \cdots + O(h^2) \text{ error}$$

$$h \to h/2 \implies \text{error} \to \text{error}/4 \text{ per step} \times \text{twice as many steps} \to \text{error}/2$$

- First-order method: Accuracy varies with $h$
- To get 100x better accuracy need 100x more steps
Questions?
Can we do better?

- Problem: $f$ varies along our Euler step
- Idea 1: look at $f$ at the arrival of the step and compensate for variation
2nd Order Methods

• Let

\[
\begin{align*}
f_0 &= f(X_0, t_0) \\
f_1 &= f(X_0 + hf_0, t_0 + h)
\end{align*}
\]

• Then

\[
X(t_0 + h) = X_0 + \frac{h}{2}(f_0 + f_1) + O(h^3)
\]

• This is the trapezoid method
  – Analysis omitted (see 6.839)

• Note! What we mean by “2nd order” is that the error goes down with \( h^2 \), not \( h \) – the equation is still 1st order!
Can we do better?

- Problem: $f$ has varied along our Euler step
- Idea 2: look at $f$ after a smaller step, use that value for a full step from initial position
2nd Order Methods cont’d

• This translates to...

\[ f_0 = f(X_0, t_0) \]
\[ f_m = f(X_0 + \frac{h}{2} f_0, t_0 + \frac{h}{2}) \]

• and we get

\[ X(t_0 + h) = X_0 + h f_m + O(h^3) \]

• This is the *midpoint method*
  – Analysis omitted again,
    but it’s not very complicated, see [here](#).
Comparison

- **Midpoint:**
  - $\frac{1}{2}$ Euler step
  - evaluate $f_m$
  - full step using $f_m$

- **Trapezoid:**
  - Euler step (a)
  - evaluate $f_1$
  - full step using $f_1$ (b)
  - average (a) and (b)

- Not exactly same result, but same order of accuracy
Can we do even better?

• You bet!
• You will implement Runge-Kutta for assignment 3

• Again, see Witkin, Baraff, Kass: Physically-based Modeling Course Notes, SIGGRAPH 2001

• Comparison Demo a little later
Questions?
Mass-Spring Modeling

- Beyond pointlike objects: strings, cloth, hair, etc.
- Interaction between particles
  - Create a network of spring forces that link pairs of particles

- First, slightly hacky version of cloth simulation
- Then, some motivation/intuition for *implicit integration*
How would you simulate a string?

- Each particle is linked to two particles (except ends)
- Come up with forces that try to keep the distance between particles constant
- What forces?
Springs
Spring Force – Hooke’s Law

Rest length $L_0$

$F = L_0 - ||P_j-P_i||$
Spring Force – Hooke’s Law

- Force in the direction of the spring and proportional to difference with rest length $L_0$.

$$F(P_i, P_j) = K(L_0 - ||P_i P_j||) \frac{P_i P_j}{||P_i P_j||}$$

- $K$ is the stiffness of the spring
  - When $K$ gets bigger, the spring really wants to keep its rest length

MIT EECS 6.837 Fall 09 – Lehtinen
Spring Force – Hooke’s Law

• Force in the direction of the spring and proportional to difference with rest length $L_0$.

$$F(P_i, P_j) = K(L_0 - ||P_i\vec{P}_j||)\frac{P_i\vec{P}_j}{||P_i\vec{P}_j||}$$

• $K$ is the stiffness of the spring
  – When $K$ gets bigger, the spring really wants to keep its rest length

This is the force on $P_j$. Remember Newton: $P_i$ experiences force of equal magnitude but opposite direction.
Mass on a Spring, Phase Space

- Click image for link
How would you simulate a string?

- Springs link the particles
- Springs try to keep their rest lengths and preserve the length of the string
- Not exactly preserved though, and we get oscillation
  - Rubber band approximation
Demo – Spring and ODE Solvers

- Forces:
  - Springs
  - structural forces (to resist bending)
  - damping

- Effects of varying spring stiffness and damping

- Euler vs. Midpoint
Questions?
Mass-Spring Cloth
Cloth – Three Types of Forces

• **Structural forces**
  – Try to enforce invariant properties of the system
    • E.g. force the distance between two particles to be constant
  – Ideally, these should be *constraints*, not forces

• **Internal deformation forces**
  – E.g. a string deforms, a spring board tries to remain flat

• **External forces**
  – Gravity, etc.
Hair

- Linear set of particles
- Length-preserving structural springs like before
- Deformation forces proportional to the angle between segments
Springs for Cloth

- Network of masses and springs
- Structural springs:
  - link \((i, j)\) and \((i+1, j)\);
  - and \((i, j)\) and \((i, j+1)\)
- Shear springs
  - \((i, j)\) and \((i+1, j+1)\)
- Flexion springs
  - \((i,j)\) and \((i+2,j)\);
  - \((i,j)\) and \((i,j+2)\)

- See Provot’s Graphics Interface ’95 paper for details
External Forces

- Gravity G
- Viscous damping C
- Wind, etc.
Cloth Simulation

- Then, the all trick is to set the stiffness of all springs to get realistic motion!

- Remember that forces depend on other particles (coupled system)

- But it is *sparse* (only near neighbors)
  - This is in contrast to e.g. the N-body problem.
Contact forces

• Hanging curtain:
• 2 contact points stay fixed
• What does it mean?
  – Sum of the forces is zero
• How so?
  – Because those points undergo an external force that balances the system
• What is the force at the contact?
  – Depends on all other forces in the system
  – Gravity, wind, etc.
Contact forces

- How can we compute the external contact force?
  - Inverse dynamics!
  - Sum all other forces applied to point
  - Take negative

- Do we really need to compute this force?
  - Not really, just ignore the other forces applied to this point!

O:-)
Example

- Excessive rubbery deformation: the strings are not stiff enough
One solution

- Constrain length to increase by less than 10%
  - A little hacky

Simple mass-spring system

Improved solution
(see Provot Graphics Interface 1995)

http://citeseer.ist.psu.edu/provot96deformation.html
The Discretization Problem

• What happens if we discretize our cloth more finely?
• Do we get the same behavior?
• Usually not! It takes a lot of effort to design a scheme that is mostly oblivious to the discretization.
Questions?
The Stiffness Issue

• We use springs while we really mean constraint
  – Spring should be super stiff, which requires tiny $\Delta t$
  – Remember $x' = -kx$ system and Euler speed limit!
    • The story extends to N particles and springs (unfortunately)

• Many numerical solutions
  – Reduce $\Delta t$ (well, not a great solution)
  – Actually use constraints (see 6.839)
  – Implicit integration scheme (more next Thursday)
Euler has a speed limit!

- $h > 1/k$: oscillate. $h > 2/k$: explode!

From the SIGGRAPH PBM notes
Why Stiff Springs are Difficult

• 1D example, with two particles constrained to move along the $x$ axis only, rest length $L_0 = 1$

• Phase space is 4D: $(x_1, v_1, x_2, v_2)$
  – But spring force only depends on $x_1, x_2$ and $L_0$. 

\[ L_0 = 1 \]
Why Stiff Springs are Difficult

\[ K = 1 \]

height = magnitude of spring force
Why Stiff Springs are Difficult

Forces grow really big!
Why Stiff Springs are Difficult

Forces grow really big!

The “admissible region” shrinks towards the line $x_1 - x_2 = 1$ as $K$ grows.
Why Stiff Springs are Difficult

The "admissible region" shrinks towards the line $x_1 - x_2 = 1$ as $K$ grows.

Forces grow really big!
Constrained Dynamics

• In our mass-spring cloth, we have “encouraged” length preservation using springs that want to have a given length (unfortunately, they can refuse offer ;-) )

• Constrained dynamic simulation: force it to be constant!

• How it works – more on 6.839
  – Start with constraint equation
    • E.g., $(x_2-x_1)-1 = 0$ in the previous 1D example
  – Derive extra forces that will exactly enforce constraint
    • This means projecting the external forces (like gravity) onto the “subspace” of phase space where constraints are satisfied
    • Fancy name for this: “Lagrange multipliers”
  – Again, see the SIGGRAPH 2001 Course Notes
Questions?

• Further reading
  – Stiff systems
  – Explicit vs. implicit solvers
  – Again, consult the 2001 course notes!
The collision problem

- A cloth has many points of contact
- Requires
  - Efficient collision detection
  - Efficient numerical treatment (stability)

Image from Bridson et al.
Collisions

- Cloth has many points of contact
- Need efficient collision detection and stable treatment
Cool Cloth/Hair Demos

• Robert Bridson, Ronald Fedkiw & John Anderson: Robust Treatment of Collisions, Contact and Friction for Cloth Animation
  SIGGRAPH 2002


Questions?
Implementation Notes

• It pays off to abstract (as usual)
  – It’s easy to design your particle system and ODE solver to be unaware of each other

• Basic idea
  – Particle system and solver communicate via floating-point vectors and a function that computes \( f(X,t) \)
    • Solver does not need to know anything else!
Implementation Notes

• Basic idea
  – Particle system tells ODE solver how many dimensions (N) the phase space has
  – Particle system has a function to write its state to an N-vector of floating point numbers (and read state from it)
  – Particle system has a function that evaluates $f(X,t)$, given a state vector $X$ and time $t$

  – ODE solver reads state first, computes the derivatives using $f(X,t)$ where needed, outputs a new state vector $X_{n+1}$
Example (Actual Code from String Demo)

```cpp
// read the current positions + velocities (X) from particles
m_psystem->getState( vecState );

switch( m_integrator )
{
    case EULER:
        // derivatives = f(X,t) at current state X
        m_psystem->evaluateDerivatives( vecState, vecDerivatives );
        // state = state + dt*derivatives
        stepSystem( dt, vecState, vecDerivatives, vecState );
        break;

    ....
}

m_psystem->setState( vecState );
```

The `stepSystem(dt, X, f, X_{new})` function just computes

\[ X_{new} = X + dt \cdot f \]
// read the current positions + velocities from particles
m_psystem->getState( vecState );

switch( m_integrator )
{
    case EULER:
        ...
        break;
    case MIDPOINT:
        // evaluate f(X,t) at current state X
        m_psystem->evaluateDerivatives( vecState, vecDerivatives );
        // go half a step forward, intermediate state = state + 0.5*dt*derivatives
        stepSystem( dt/2.0f, vecState, vecDerivatives, vecIntermediateState[0] );
        // evaluate derivatives after half timestep
        m_psystem->evaluateDerivatives( vecIntermediateState[0], vecDerivatives );
        // full timestep using t+0.5dt derivatives FROM ORIGINAL POSITION
        stepSystem( dt, vecState, vecDerivatives, vecIntermediateState[0] );
        vecState = vecIntermediateState[0];
        break;
}

m_psystem->setState( vecState );
Black-box Solver

• Such a design is nice, because you can reuse your solver
  – You could implement a C++ base class “ODE” that would have get/setState functions and evaluateDerivatives as virtual members
  – Then, your solver would just take in a pointer to ODE and compute new states without knowing anything about what is going on in the system!
  • States and derivatives are just numbers!
Questions?
Simulation + Cheat = Profit, Again


- Again, simulate on coarser resolution, add detail with procedural forces!
  - Gives realistic motion on coarse scale, interesting heuristic small-scale detail