Particle Systems and ODEs
Types of Animation

• Keyframing
• Procedural
• Physically-based
  – Particle Systems: TODAY
    • Smoke, water, fire, sparks, etc.
    • Usually heuristic as opposed to simulation, but not always
    • Mass-Spring Models (Cloth) NEXT CLASS
  – Continuum Mechanics (fluids, etc.), finite elements
    • Not in this class
  – Rigid body simulation
    • Not in this class
Types of Animation: Physically-Based

- Assign physical properties to objects
  - Masses, forces, etc.
- Also procedural forces (like wind)
- Simulate physics by solving equations of motion
  - Rigid bodies, fluids, plastic deformation, etc.
- Realistic but difficult to control

\[ m \quad v_0 \quad g \]
Types of Dynamics

• Point
Types of Dynamics

- Point
- Rigid body
Types of Dynamics

- Point
- Rigid body
- Deformable body
  (include clothes, fluids, smoke, etc.)
Today We Focus on Point Dynamics

- Lots of points!
- Particles systems
  - Borderline between procedural and physically-based
Particle Systems Overview

• **Emitters** generate tons of “particles”
  – Sprinkler, waterfall, chimney,
    gun muzzle, exhaust pipe, etc.
Particle Systems Overview

- **Emitters** generate tons of “particles”
- Describe the external **forces** with a force field
  - E.g., gravity, wind
Particle Systems Overview

• **Emitters** generate tons of “particles”
• Describe the external **forces** with a force field
• **Integrate** the laws of mechanics (ODEs)
  – Makes the particles move

Images of particle systems removed due to copyright restrictions.

http://www.particlesystems.org/
Particle Systems Overview

- **Emitters** generate tons of “particles”
- Describe the external **forces** with a force field
- **Integrate** the laws of mechanics (ODEs)
- In the simplest case, each particle is **independent**

Images of particle systems removed due to copyright restrictions.

http://www.particlesystems.org/
Particle Systems Overview

- **Emitters** generate tons of “particles”
- Describe the external **forces** with a force field
- **Integrate** the laws of mechanics (ODEs)
- In the simplest case, each particle is **independent**
- If there is enough **randomness** (in particular at the emitter) you get nice effects
  - sand, dust, smoke, sparks, flame, water, …

Images of particle systems removed due to copyright restrictions.

http://www.particlesystems.org/
Sprinkler

• http://www.youtube.com/watch?v=rhvH12nC6_Q
Fire

- [http://www.youtube.com/watch?v=6hG00etwRBU](http://www.youtube.com/watch?v=6hG00etwRBU)
Generalizations

• More advanced versions of behavior
  – flocks, crowds

• Forces between particles
  – Not independent any more


We’ll come back to this a little later.

Generalizations – Next Class

- Mass-spring and deformable surface dynamics
  - surface represented as a set of points
  - forces between neighbors keep the surface coherent
• It’s not all hacks: **Smoothed Particle Hydrodynamics (SPH)**
  – A family of “real” particle-based fluid simulation techniques.

  – Fluid flow is described by the **Navier-Stokes Equations**, a nonlinear partial differential equation (PDE)
    • SPH discretizes the fluid as small packets (particles!), and evaluates pressures and forces based on them.
These Stanford folks use SPH for resolving the small-scale spray and mist that would otherwise be too much for the grid solver to handle.

Real-time particles in games

- http://www.youtube.com/watch?v=6DicVajK2xQ
EA Fight Night 4 Physics Trailer

MAY CONTAIN CONTENT INAPPROPRIATE FOR CHILDREN

Visit www.esrb.org for rating information

This video is intended for promotional purposes only, and may not be sold, rented nor reproduced by any party. Any unauthorized use of this video is prohibited by applicable law.
Take-Home Message

• Particle-based methods can range from pure heuristics (hacks that happen to look good) to "real" simulation

• Basics are the same: Things always boil down to integrating ODEs!
  – Also in the case of grids/computational meshes
What is a Particle System?

- Collection of many small simple pointlike things
  - Described by their current state: position, velocity, age, color, etc.

- Particle motion influenced by external force fields and internal forces between particles

- Particles created by *generators* or *emitters*
  - With some randomness

- Particles often have lifetimes

- Particles are often independent

- Treat as points for dynamics, but rendered as anything you want

Image courtesy of Halixi72 on Wikimedia Commons. License: CC-BY-SA. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.
Simple Particle System: Sprinkler

PL: linked list of particle = empty;

Image by Jeff Lander removed due to copyright restrictions.
Simple Particle System: Sprinkler

PL: linked list of particle = empty;
spread=0.1; //how random the initial velocity is
colorSpread=0.1; //how random the colors are
Simple Particle System: Sprinkler

PL: linked list of particle = empty;
spread=0.1;  //how random the initial velocity is
colorSpread=0.1;  //how random the colors are

For each time step

Image by Jeff Lander removed due to copyright restrictions.
Simple Particle System: Sprinkler

PL: linked list of particle = empty;
spread=0.1; //how random the initial velocity is
colorSpread=0.1; //how random the colors are
For each time step
  Generate k particles
    p=new particle();
    p->position=(0,0,0);
    p->velocity=(0,0,1)+spread*(rnd(), rnd(), rnd());
    p.color=(0,0,1)+colorSpread*(rnd(), rnd(),rnd());
    PL->add(p);
Simple Particle System: Sprinkler

PL: linked list of particle = empty;
spread=0.1; //how random the initial velocity is
colorSpread=0.1; //how random the colors are

For each time step
  Generate k particles
    p=new particle();
    p->position=(0,0,0);
    p->velocity=(0,0,1)+spread*(rnd(), rnd(), rnd());
    p.color=(0,0,1)+colorSpread*(rnd(), rnd(), rnd());
    PL->add(p);

  For each particle p in PL
    p->position+=p->velocity*dt; //dt: time step
    p->velocity-=g*dt; //g: gravitation constant
    glColor(p.color);
    glVertex(p.position);
Simple Particle System: Sprinkler

PL: linked list of particle = empty;
spread=0.1; //how random the initial velocity is
colorSpread=0.1; //how random the colors are

For each time step
  Generate k particles
    p=new particle();
    p->position=(0,0,0);
    p->velocity=(0,0,1)+spread*(rnd(), rnd(), rnd());
    p.color=(0,0,1)+colorSpread*(rnd(), rnd(),rnd());
    PL->add(p);
  For each particle p in PL
    p->position+=p->velocity*dt; //dt: time step
    p->velocity-=g*dt; //g: gravitation constant
    glColor(p.color);
    glVertex(p.position);
Demo with Processing

- http://processing.org/learning/topics/simpleparticlesystem.html
Questions?

Image removed due to copyright restrictions.
Path forward

- Basic particle systems are simple hacks
- Extend to physical simulations, e.g., clothes
- For this, we need to understand numerical integration
- This lecture: point particles
- Next lecture: mass-spring and clothes
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
Newtonian Mechanics

• Point mass: 2nd order ODE

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

• Position \( \vec{x} \) and force \( \vec{F} \) are vector quantities
  – We know \( \vec{F} \) and \( m \), want to solve for \( \vec{x} \)

• You have all seen this a million times before
Reduction to 1\textsuperscript{st} Order

- Point mass: 2\textsuperscript{nd} order ODE

$$\vec{F} = m \vec{a} \quad \text{or} \quad \vec{F} = m \frac{d^2 \vec{x}}{dt^2}$$

- 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 ($\vec{x}$, $\vec{v}$) instead of just $\vec{x}$
Reduction to 1\textsuperscript{st} Order

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

- Why reduce?

2 variables (x, v) instead of just one
Reduction to 1\textsuperscript{st} Order

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

2 variables (\(x, v\)) instead of just one

- Why reduce?
  - Numerical solvers grow more complicated with increasing order, can just write one 1st order solver and use it
  - Note that this doesn’t mean it would always be easy :-)

• Why reduce?
Notation

• Let’s stack the pair \((x, v)\) into a bigger state vector \(X\)

\[
X = \begin{pmatrix} x \vspace{2pt} \& \vspace{2pt} v \end{pmatrix}
\]

For a particle in 3D, state vector \(X\) has 6 numbers

\[
\frac{d}{dt} X = f(X, t) = \begin{pmatrix} \vec{v} \vspace{2pt} \& \vspace{2pt} \vec{F}(x, v)/m \end{pmatrix}
\]
Now, Many Particles

- We have N point masses
  - Let’s just stack all xs and vs in a big vector of length 6N

\[
X = \begin{pmatrix}
  x_1 \\
  v_1 \\
  \vdots \\
  x_N \\
  v_N
\end{pmatrix}
\]

\[
f(X, t) = \begin{pmatrix}
  v_1 \\
  F^1(X, t) \\
  \vdots \\
  v_N \\
  F^N(X, t)
\end{pmatrix}
\]
Now, Many Particles

- We have N point masses
  - Let’s just stack all $x$s and $v$s in a big vector of length $6N$
  - $F^i$ denotes the force on particle $i$
    - When particles don’t interact, $F^i$ only depends on $x_i$ and $v_i$.

\[
X = \begin{pmatrix}
  x_1 \\
  v_1 \\
  \vdots \\
  x_N \\
  v_N
\end{pmatrix}
\]

\[
f(X, t) = \begin{pmatrix}
  v_1 \\
  F^1(X, t) \\
  \vdots \\
  v_N \\
  F^N(X, t)
\end{pmatrix}
\]

$f$ gives $d/dt X$, remember!
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 $dt$?”
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.
Questions?

http://vimeo.com/14597952
Numerics of ODEs

- Numerical solution is called “integration of the ODE”
- Many techniques
  - Today, the simplest one
  - Thursday and next week we’ll look at some more advanced techniques
Intuitive Solution: Take Steps

- Current state \( X \)
- Examine \( f(X,t) \) at (or near) current state
- Take a step to new value of \( X \)

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

\[
\Rightarrow \textit{d}X = dt \cdot f(X, t)
\]

\( f = d/dt \, X \) is a vector that sits at each point in phase space, pointing the direction.
Euler’s Method

- Simplest and most intuitive
- Pick a step size $h$
- Given $X_0 = X(t_0)$, take step:

  $$t_1 = t_0 + h$$
  $$X_1 = X_0 + h f(X_0, t_0)$$

- Piecewise-linear approximation to the path
- Basically, just replace $dt$ by a small but finite number
Euler, Visually

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

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

Image by MIT OpenCourseWare.
Euler, Visually

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

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

Image removed due to copyright restrictions.
Effect of Step Size

- Step size controls accuracy
- Smaller steps more closely follow curve
  - May need to take many small steps per frame
  - Properties of $f(X, t)$ determine this (more later)
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’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

Image by MIT OpenCourseWare.
More Accurate Alternatives

• Midpoint, Trapezoid, Runge-Kutta
  – Also, “implicit methods” (next week)

More on this during next class

• Extremely valuable resource: SIGGRAPH 2001 course notes on physically based modeling
What is a Force?

- A force changes the motion of the system
  - Newton says: When there are no forces, motion continues uniformly in a straight line (good enough for us)

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

- For point masses, forces are vectors
  - I.e., to get total force, take vector sum of everything
Forces: Gravity on Earth

- Depends only on particle mass
- \( f(X, t) = \text{constant} \)
- Hack for smoke, etc: make gravity point up!
  - Well, you can call this buoyancy, too.
Forces: Gravity (N-body problem)

- Gravity depends on all other particles
- Opposite for pairs of particles
- Force in the direction of $\mathbf{p}_i - \mathbf{p}_j$ with magnitude inversely proportional to square distance

$$\| \mathbf{F}_{ij} \| = \frac{G m_i m_j}{r^2}$$

where $G = 6.67 \times 10^{-11}$ Nm$^2$/kg$^2$

- Testing all pairs is $O(n^2)$!

Particles are not independent!
Real-Time Gravity Demo

http://www.youtube.com/watch?v=uhTuJZiAG64

NVIDIA
An Aside on Gravity

• That was Brute Force
  – Meaning all $O(n^2)$ pairs of particles were considered when computing forces
  – Yes, computers are fast these days, but this gets prohibitively expensive soon. (The square in $n^2$ wins.)

• Hierarchical techniques approximate forces caused by many distant attractors by one force, yields $O(n)$!
  – This inspired very cool hierarchical illumination rendering algorithms in graphics (hierarchical radiosity, etc.)
Forces: Viscous Damping

\[ f^{(i)} = -dv^{(i)} \]

- Damping force on particle i determined its velocity
  - Opposes motion
  - E.g. wind resistance
- Removes energy, so system can settle
- Small amount of damping can stabilize solver
- Too much damping makes motion like in glue
Forces: Spatial Fields

- Externally specified force (or velocity) fields in space
- Force on particle $i$ depends only on its position
- Arbitrary functions
  - wind
  - attractors, repulsors
  - vortices
- Can depend on time
- Note: these add energy, may need damping
Processing demo

- [http://processing.org/learning/topics/smokeparticlesystem.html](http://processing.org/learning/topics/smokeparticlesystem.html)
Example: Procedural Spatial Field


Plausible, controllable force fields – just advecting particles along the flow gives cool results!

And it’s simple, too!
Curl-Noise for Procedural Fluid Flow

Robert Bridson
Jim Hourihan
Markus Nordenstam
Forces: Other Spatial Interaction

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!
http://www.youtube.com/watch?v=nl7makIgYnl&feature=related
Lennard-Jones forces

http://www.youtube.com/watch?v=XfjYIKxKIWQ&feature=autoplay&list=PL0605C44C6E8D5EDB&If=autoplay&playnext=2
Questions?

http://www.youtube.com/watch?v=dHWCT7RPjPo
Collisions

- Detection
- Response
- Covered later
More Eyecandy from NVIDIA

- Fluid flow solved using a regular grid solver
  - This gives a velocity field
- 0.5M smoke particles advected using the field
  - That means particle velocity is given by field
- Particles are for rendering, motion solved using other methods

[Link to video](#)
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

© ACM. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

Courtesy of Craig W. Reynolds. Used with permission.
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
Predator-Prey

- http://www.youtube.com/watch?v=rN8DzIgMt3M
Massive software

• http://www.massivesoftware.com/
• Used for battle scenes in the Lord of The Rings
Processing demo

• http://processing.org/learning/topics/flocking.html
Battle of the Helm’s deep, LOTR

Image removed due to copyright restrictions.
Questions?

Image removed due to copyright restrictions.
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
  - create $n$ particles.
  - update $t_{last}$ if $n > 0$
- Create with (random) distribution of initial $\mathbf{x}$ and $\mathbf{v}$
  - if creating $n > 1$ particles at once, spread out on path

\[
    n = \left\lfloor (t - t_{last}) \times \text{rate} \right\rfloor
\]

[Image removed due to copyright restrictions.](http://www.particlesystems.org/)
Particle Controls

• 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!
UNREAL DEVELOPMENT KIT
Tutorial: Materials and Particles for Fire

by Lee A. (Gizmosan)
• 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 \cdot dt)
  – Or an elongated quad with texture
Rendering and Motion Blur

© Konami Digital Entertainment, Inc.. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.
• Often use texture maps (fire, clouds, smoke puffs)
  – Called “billboards” or “sprites”
  – Always parallel to image plane
Star Trek 2 – The Wrath of Khan

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

• Described in [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!

Particle Modeling [Reeves 1983]
Questions?

Early particle fun by Karl Sims

Courtesy of Karl Sims. Used with permission.
That’s All for Today!

• Further reading
    • Extremely good, easy-to-read resource. Highly recommended!
  – William Reeves: Particle systems—a technique for modeling a class of fuzzy objects, Proc. SIGGRAPH 1983
    • The original paper on particle systems
  – particlesystems.org