CS354 Computer Graphics
Particle Systems

Qixing Huang
April 18th 2018
Reading

• Required:
  – Witkin and Baraff, *Differential Equation Basics*, SIGGRAPH ’01 course notes on Physically Based Modeling.

• Optional
What are particle systems?

• A particle system is a collection of point masses that obeys some physical laws (e.g., gravity, heat convection, spring behaviors, ...).

• Particle systems can be used to simulate all sorts of physical phenomena:
Particle in a flow field

• We begin with a single particle with:

\[
\vec{x} = \begin{bmatrix} x \\ y \end{bmatrix}
\]

\[
\text{Position, } \quad \vec{x} = \begin{bmatrix} x \\ y \end{bmatrix}
\]

\[
\text{Velocity, } \quad \vec{v} = \dot{\vec{x}} = \frac{d\vec{x}}{dt} = \begin{bmatrix} dx/dt \\ dy/dt \end{bmatrix}
\]

• Suppose the velocity is actually dictated by some driving function \(g\): \(\mathbf{x} = g(\mathbf{x}, t)\)
Vector fields

• At any moment in time, the function $g$ defines a vector field over $\mathbf{x}$:

• How does our particle move through the vector field?
Diff eqs and integral curves

- The equation \( \dot{x} = g(\vec{x}, t) \)
  is actually a **first order differential equation**.
- We can solve for \( x \) through time by starting at an initial
  point and stepping along the vector field:

- This is called an **initial value problem** and the solution is
  called an **integral curve**.
Euler’s method

- One simple approach is to choose a time step, $\Delta t$, and take linear steps along the flow:
  $$\vec{x}(t + \Delta t) = \vec{x}(t) + \Delta t \cdot \vec{v}(t) = \vec{x}(t) + \Delta t \cdot g(\vec{x},t)$$

- Writing as a time iteration:
  $$\vec{x}^{i+1} = \vec{x}^i + \Delta t \cdot \vec{v}^i$$

- This approach is called Euler’s method and looks like:

- Properties:
  - Simplest numerical method
  - Bigger steps, bigger errors. Error $\sim O(\Delta t^2)$. 
  - Need to take pretty small steps, so not very efficient. Better (more complicated) methods exist, e.g., “Runge-Kutta” and “implicit integration.”
Particle in a force field

- Now consider a particle in a force field $\mathbf{f}$.
- In this case, the particle has:
  - Mass, $m$
  - Acceleration, $\ddot{\mathbf{a}} \equiv \ddot{\mathbf{x}} = \frac{d\mathbf{v}}{dt} = \frac{d^2\mathbf{x}}{dt^2}$

- The particle obeys Newton’s law: $\mathbf{f} = m\ddot{\mathbf{a}} = m\ddot{\mathbf{x}}$

- The force field $\mathbf{f}$ can in general depend on the position and velocity of the particle as well as time.
- Thus, with some rearrangement, we end up with:

$$\ddot{\mathbf{x}} = \frac{\mathbf{f}(\mathbf{x}, \dot{\mathbf{x}}, t)}{m}$$
Second order equations

This equation:

\[ \ddot{x} = \frac{\vec{f}(\vec{x}, \dot{\vec{x}}, t)}{m} \]

is a second order differential equation.

Our solution method, though, worked on first order differential equations.

We can rewrite this as:

\[
\begin{bmatrix}
\dot{\vec{x}} = \vec{v} \\
\dot{\vec{v}} = \frac{\vec{f}(\vec{x}, \vec{v}, t)}{m}
\end{bmatrix}
\]

where we have added a new variable \( \vec{v} \) to get a pair of coupled first order equations.
**Phase space**

$$\begin{bmatrix} \ddot{x} \\ \ddot{v} \end{bmatrix}$$

- Concatenate $x$ and $v$ to make a 6-vector: position in **phase space**.

$$\begin{bmatrix} \dot{x} \\ \dot{v} \end{bmatrix}$$

- Taking the time derivative: another 6-vector.

$$\begin{bmatrix} \dot{x} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} \ddot{v} \\ \ddot{f}/m \end{bmatrix}$$

- A vanilla $1^{\text{st}}$-order differential equation.
Differential equation solver

Starting with:
\[
\begin{bmatrix}
\dot{x} \\
\dot{v}
\end{bmatrix} = \begin{bmatrix}
v \\
f/m
\end{bmatrix}
\]

Applying Euler’s method:
\[
\ddot{x}(t + \Delta t) = \ddot{x}(t) + \Delta t \cdot \dot{x}(t)
\]
\[
\dot{x}(t + \Delta t) = \dot{x}(t) + \Delta t \cdot \ddot{x}(t)
\]

And making substitutions:
\[
\ddot{x}(t + \Delta t) = \ddot{x}(t) + \Delta t \cdot \ddot{v}(t)
\]
\[
\dot{x}(t + \Delta t) = \dot{x}(t) + \Delta t \cdot \frac{\ddot{f}(\ddot{x}, \dot{x}, t)}{m}
\]

Writing this as an iteration, we have:
\[
\ddot{x}^{i+1} = \ddot{x}^i + \Delta t \cdot \ddot{v}^i
\]
\[
\ddot{v}^{i+1} = \ddot{v}^i + \Delta t \cdot \frac{\ddot{f}^i}{m}
\]

Again, performs poorly for large \(\Delta t\).
Verlet Integration

• Also called Størmer’s Method
  – Invented by Delambre (1791), Størmer (1907), Cowell and Crommelin (1909), Verlet (1960) and probably others

• More stable than Euler’s method (time reversible as well)
Forces

• Each particle can experience a force which sends it on its merry way.

• Where do these forces come from? Some examples:
  – Constant (gravity)
  – Position/time dependent (force fields)
  – Velocity-dependent (drag)
  – Combinations (Damped springs)

• How do we compute the net force on a particle?
Gravity and viscous drag

The force due to gravity is simply:

$$\vec{f}_{\text{grav}} = m\vec{G}$$

$$p\rightarrow f = p\rightarrow m \times F\rightarrow G$$

Often, we want to slow things down with viscous drag:

$$\vec{f}_{\text{drag}} = -k\vec{v}$$

$$p\rightarrow f = F\rightarrow k \times p\rightarrow v$$
Damped spring

Recall the equation for the force due to a spring: \[ f = -k_{spring} (|\Delta \vec{x}| - r) \]

We can augment this with damping: \[ f = - \left[ k_{spring} (|\Delta \vec{x}| - r) + k_{damp} |\vec{v}| \right] \]

The resulting force equations for a spring between two particles become:

\[
\begin{align*}
\vec{f}_1 &= - \left[ k_{spring} (|\Delta \vec{x}| - r) + k_{damp} \left( \frac{\Delta \vec{v} \cdot \Delta \vec{x}}{|\Delta \vec{x}|} \right) \right] \frac{\Delta \vec{x}}{|\Delta \vec{x}|} \\
\vec{f}_2 &= - \vec{f}_1 \\
\end{align*}
\]

\[ r = \text{rest length} \]

\[ \Delta \vec{x} = \vec{x}_1 - \vec{x}_2 \]

\[ p_1 = \begin{bmatrix} x_1 \\ v_1 \end{bmatrix} \]

\[ p_2 = \begin{bmatrix} x_2 \\ v_2 \end{bmatrix} \]
derivEval

Clear forces
  Loop over particles, zero force accumulators
Calculate forces
  Sum all forces into accumulators
Return derivatives
  Loop over particles, return $v$ and $f/m$

1. Clear force accumulators

2. Apply forces to particles

3. Return derivatives to solver

\[
\begin{bmatrix}
  \vec{v}_1 \\
  f_1/m_1 \\
\end{bmatrix}
\begin{bmatrix}
  \vec{v}_2 \\
  f_2/m_2 \\
\end{bmatrix}
\ldots
\begin{bmatrix}
  \vec{v}_n \\
  f_n/m_n \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
  \vec{x}_1 \\
  \vec{v}_1 \\
  f_1 \\
  m_1 \\
\end{bmatrix}
\begin{bmatrix}
  \vec{x}_2 \\
  \vec{v}_2 \\
  f_2 \\
  m_2 \\
\end{bmatrix}
\ldots
\begin{bmatrix}
  \vec{x}_n \\
  \vec{v}_n \\
  f_n \\
  m_n \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
  F_1 \\
  F_2 \\
  F_3 \\
  F_{nf} \\
\end{bmatrix}
\]
Bouncing off the walls

- Add-on for a particle simulator
- For now, just simple point-plane collisions

A plane is fully specified by any point $P$ on the plane and its normal $N$. 

Collision Detection

How do you decide when you’ve crossed a plane?

[Diagram showing a point P, a vector V, and a plane with a normal N and a point X]
Normal and tangential velocity

To compute the collision response, we need to consider the normal and tangential components of a particle’s velocity.

\[ \vec{v}_N = (\vec{N} \cdot \vec{v}) \vec{N} \]

\[ \vec{v}_T = \vec{v} - \vec{v}_N \]
Collison Response

\[ \vec{V}' = \vec{V}_T - k_{\text{restitution}} \vec{V}_N \]

Without backtracking, the response may not be enough to bring a particle to the other side of a wall. In that case, detection should include a velocity check:
Discussion