Physically Based Simulation II:
Particle Systems

CS 248: Interactive Computer Graphics
Prof. Vladlen Koltun
Stanford University

(Based on “Physically Based Modeling” by Witkin and Baraff)

Winter 2010
Images from Sims, “Parallel animation and rendering using data parallel computation”, 1990
Images of “Star Trek II: The Wrath of Khan” from Reeves, “Particle systems - A technique for modeling a class of fuzzy objects”, 1983
Particles

Point masses that respond to forces.

Easy: No spatial extent, no angular mass, no moment of inertia tensor, no surface-surface contact...

Versatile
Particle rendering

Textured point primitives (GL_POINT_SPRITE_ARB). Given a single point, draws a square textured billboard of specified size.

Texture is a circular “splat” with color attenuation

Ideally attenuate opacity as well, but transparency is tricky with the graphics pipeline

Can be made elliptical with fragment shading
Particle lifecycle

*Emitters* stochastically emit particles from surfaces with initial velocity and other properties.

Particle dynamics is simulated over time. Properties like color, size, and shape can be updated as well.

Particles are stochastically decimated, or terminated at the end of a set lifespan.
Particles

\[ f = ma \]

\[ \ddot{x} = \frac{f}{m} \]

Second order equation, doesn’t fit into simulation framework.
Phase space

\[
\begin{align*}
\dot{x} &= v \\
\dot{v} &= f/m \\
\begin{bmatrix}
\dot{x}_1 \\
\dot{x}_2 \\
\dot{x}_3 \\
\dot{v}_1 \\
\dot{v}_2 \\
\dot{v}_3
\end{bmatrix} &= 
\begin{bmatrix}
v_1 \\
v_2 \\
v_3 \\
f_1/m \\
f_2/m \\
f_3/m
\end{bmatrix}
\end{align*}
\]

\[(f = f(x, \dot{x}, t))\]
Solver interface

Particle System

Solver Interface

Dim(State)

Get/Set State

Deriv Eval

\[
\begin{align*}
\text{Dim(State)} & : \text{particles} \quad n \quad \text{time} \\
\text{Get/Set State} & : x_1 \quad v_1 \quad x_2 \quad v_2 \cdots x_n \quad v_n \\
\text{Deriv Eval} & : v_1 \quad \frac{f_1}{m_1} \quad v_2 \quad \frac{f_2}{m_2} \cdots v_n \quad \frac{f_n}{m_n}
\end{align*}
\]
Forces

Gravity

Drag

Attraction and repulsion ("flocking")

Springs (next lecture)
Application of forces

Maintain a list of force objects that can apply themselves to any set of particles
Derivative evaluation with forces

Iterate over force objects, apply each

Newton’s first law: The forces update $\dot{v}$, not $v$
Unary forces

Gravity: \( \mathbf{f} = m \mathbf{g} \), where \( \mathbf{g} \) is the gravity vector

Viscous drag: \( \mathbf{f} = -k_d \mathbf{v} \), where \( k_d \) is the coefficient of drag
Attraction and repulsion

Can set up spatial interaction forces that apply whenever a pair (or n-tuple) of particles are within a certain range of distances from each other.
Collision detection

If $\mathbf{v} \cdot \mathbf{n} < 0$ and $(\mathbf{x} - \mathbf{p}) \cdot \mathbf{n} < \varepsilon$, collision
Collision response

Before update: \( v_n = (v \cdot n)n, \ v_t = v - v_n \)

After update: \( v^\text{new} = v_t - r v_n, \)
where \( r \) is the coefficient of restitution.
Contact force $-f_n$ cancels out the normal component of $f$. Frictional force $-k_f f_t$ simulates friction.
Images from Sims, “Parallel animation and rendering using data parallel computation”, 1990