14. Particle Systems

Reading

Required:
- Witkin, *Particle System Dynamics*, SIGGRAPH ’97 course notes on Physically Based Modeling.

Optional
- Witkin and Baraff, *Differential Equation Basics*, SIGGRAPH ’97 course notes on Physically Based Modeling (online).

What are particle systems?

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

Particle systems can be used to simulate all sorts of physical phenomena:

Overview

- A single particle
- Particle systems
- Forces: gravity, springs
- Collision detection
Particle in a flow field

We begin with a single particle with:

- Position, \( \mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix} \)
- Velocity, \( \mathbf{v} = \frac{d\mathbf{x}}{dt} = \begin{bmatrix} \frac{dx}{dt} \\ \frac{dy}{dt} \end{bmatrix} \)

Suppose the velocity is actually dictated by some driving function \( g \):

\[
\dot{\mathbf{x}} = g(\mathbf{x}, t)
\]

Vector fields

At any moment in time, the function \( g \) defines a vector field over \( x \):

How does our particle move through the vector field?

Diff eqs and integral curves

The equation

\[
\dot{\mathbf{x}} = g(\mathbf{x}, t)
\]

is actually a first order differential equation.

We can solve for \( \mathbf{x} \) through time by starting at an initial point and stepping along the vector field:

\[
\begin{align*}
\mathbf{x}(t + \Delta t) &= \mathbf{x}(t) + \Delta t \cdot \dot{\mathbf{x}}(t) \\
&= \mathbf{x}(t) + \Delta t \cdot g(\mathbf{x}, t)
\end{align*}
\]

This approach is called Euler's method and looks like:

Properties:

- Simplest numerical method
- Bigger steps, bigger errors

Need to take pretty small steps, so not very efficient. Better (more complicated) methods exist, e.g., “Runge-Kutta.”
Particle in a force field

Now consider a particle in a force field $f$. In this case, the particle has:

- Mass, $m$
- Acceleration, $a = \frac{dv}{dt} = \frac{d^2x}{dt^2}$

The particle obeys Newton’s law:

$$f = ma = m\ddot{x}$$

The force field $f$ can in general depend on the position and velocity of the particle as well as time.

Thus, with some rearrangement, we end up with:

$$\ddot{x} = \frac{f(x,\dot{x},t)}{m}$$

Second order equations

This equation:

$$\dot{x} = \frac{f(x,v,t)}{m}$$

is a second order differential equation.

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

We can rewrite this as:

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

where we have added a new variable $v$ to get a pair of coupled first order equations.

Phase space

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

Taking the time derivative: another 6-vector.

A vanilla 1st-order differential equation.

Differential equation solver

Starting with:

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

Applying Euler method:

$$\begin{align*}
    x(t + \Delta t) &= x(t) + \Delta t \cdot \dot{x}(t) \\
    \dot{x}(t + \Delta t) &= \dot{x}(t) + \Delta t \cdot \ddot{x}(t)
\end{align*}$$

And making substitutions:

$$\begin{align*}
    x(t + \Delta t) &= x(t) + \Delta t \cdot \frac{f(x,\dot{x},t)}{m} \\
    \dot{x}(t + \Delta t) &= \dot{x}(t) + \Delta t \cdot \frac{f(x,\dot{x},t)}{m}
\end{align*}$$

Writing this as an iteration, we have:

$$\begin{align*}
    x_{i+1} &= x_i + \Delta t \cdot v_i \\
    v_{i+1} &= v_i + \Delta t \cdot \frac{f(x_i,v_i,t)}{m}
\end{align*}$$

Again, performs poorly for large $\Delta t$.  

Particle structure

How do we represent a particle?

<table>
<thead>
<tr>
<th>x</th>
<th>position</th>
</tr>
</thead>
<tbody>
<tr>
<td>v</td>
<td>velocity</td>
</tr>
<tr>
<td>f</td>
<td>force accumulator</td>
</tr>
<tr>
<td>m</td>
<td>mass</td>
</tr>
</tbody>
</table>

Position in phase space

Single particle solver interface

```
[ x ]
[ v ]
[ f ]
[ m ]

[6]
```

getDim

getState

setState

derivEval

Particle systems

In general, we have a particle system consisting of \( n \) particles to be managed over time:

```
<table>
<thead>
<tr>
<th>x</th>
<th>x</th>
<th>x</th>
<th>x</th>
<th>x</th>
</tr>
</thead>
<tbody>
<tr>
<td>v</td>
<td>v</td>
<td>v</td>
<td>v</td>
<td>v</td>
</tr>
<tr>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
</tr>
<tr>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
</tr>
</tbody>
</table>
```

Particle system solver interface

For \( n \) particles, the solver interface now looks like:

```
<table>
<thead>
<tr>
<th>particles</th>
<th>n</th>
<th>time</th>
</tr>
</thead>
</table>
```

```
<table>
<thead>
<tr>
<th>x_1</th>
<th>v_1</th>
<th>x_2</th>
<th>v_2</th>
<th>\ldots</th>
<th>x_n</th>
<th>v_n</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td></td>
<td>f</td>
<td></td>
<td>\ldots</td>
<td>f</td>
<td></td>
</tr>
<tr>
<td>m_1</td>
<td></td>
<td>m_2</td>
<td></td>
<td>\ldots</td>
<td>m_n</td>
<td></td>
</tr>
</tbody>
</table>
```
Particle system diff. eq. solver

Thus, we start with:

\[
\begin{bmatrix}
  x_1 \\
  v_1 \\
  \vdots \\
  x_n \\
  v_n 
\end{bmatrix} =
\begin{bmatrix}
  v_1 \\
  f_i / m_i \\
  \vdots \\
  v_n \\
  f_n / m_n 
\end{bmatrix}
\]

And can solve, using the Euler method:

\[
\begin{bmatrix}
  x_i^{t+1} \\
  v_i^{t+1} \\
  \vdots \\
  x_n^{t+1} \\
  v_n^{t+1} 
\end{bmatrix} =
\begin{bmatrix}
  x_i^t \\
  v_i^t \\
  \vdots \\
  x_n^t \\
  v_n^t 
\end{bmatrix} + \Delta t
\begin{bmatrix}
  v_i^t \\
  f_i / m_i \\
  \vdots \\
  v_n^t \\
  f_n / m_n 
\end{bmatrix}
\]

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)
- N-ary (springs)

How do we compute the net force on a particle?

Particle systems with forces

Force objects are black boxes that point to the particles they influence and add in their contributions.

We can now visualize the particle system with force objects:

Gravity and viscous drag

The force due to gravity is simply:

\[ f_{grav} = mG \]

\[ p->f += p->m * F->G \]

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

\[ f_{drag} = -k_{drag} v \]

\[ p->f -= F->k * p->v \]
Damped spring

A spring is a simple example of an "N-ary" force.

\[ f_1 = -k_{spring} (|\Delta x| - r) + k_{damp} \left( \frac{\Delta v \cdot \Delta x}{|\Delta x|} \right) \Delta x \]

\[ f_2 = -f_1 \]

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

\[ \Delta x = x_1 - x_2 \]

\[ \Delta v = v_1 - v_2 \]

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

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

derivEval Loop

1. Clear forces
   - Loop over particles, zero force accumulators
2. Calculate forces
   - Sum all forces into accumulators
3. Gather
   - Loop over particles, copying \( v \) and \( f/m \) into destination array

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 \( N \).

Normal and tangential components

First we need to consider the normal and tangential components of a particle’s velocity.

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

\[ \mathbf{v}_T = \mathbf{v} - \mathbf{v}_N \]
Collision Detection

To detect collisions, we need the equation for a plane:

\[(x - P) \cdot N < \epsilon \quad \text{within } \epsilon \text{ of the wall} \]
\[N \cdot v < 0 \quad \text{heading in} \]

Collision Response

\[v' = v_T - k_{\text{rest}} v_N \]

Summary

What you should take away from this lecture:

- The meanings of all the **boldfaced** terms
- Euler method for solving differential equations
- Combining particles into a particle system
- Physics of a particle system
- Various forces acting on a particle
- Simple collision detection