Particle Systems

Zoran Popovic
CSE 457
Spring 2019
Reading

• Required:

• Optional
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:

- Smoke
- Snow
- Fireworks
- Hair
- Cloth
- Snakes
- Fish
Overview

1. One lousy particle
2. Particle systems
3. Forces: gravity, springs
4. Implementation
Particle in a flow field

We begin with a single particle with:

- Position, $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

- Velocity, $\mathbf{v} \equiv \frac{d\mathbf{x}}{dt} = \begin{bmatrix} dx/dt \\ dy/dt \end{bmatrix}$

Suppose the velocity is dictated by some driving function $\mathbf{g}$:

Suppose the velocity is dictated by some driving function $\mathbf{g}$:
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 \( g(t, x) \) 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:

\[
x(t + \Delta t) = x(t) + \Delta t \cdot g(x, t)
\]

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 \( \mathbf{f} \).

• In this case, the particle has:
  – Mass, \( m \)
  – Acceleration, \( \mathbf{a} \equiv \frac{d^2 \mathbf{x}}{dt^2} \)

• The particle obeys Newton’s law: \( \mathbf{f} = ma = m \ddot{\mathbf{x}} \)

• 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:

\[
\mathbf{f}(\mathbf{v}) = \frac{\dot{\mathbf{v}}}{m}.
\]
Second order equations

This equation: \[ \frac{f(v, v, t)}{m} \]
is a second order differential equation.

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

We can rewrite this as:

\[
\begin{bmatrix}
\vdots \\
f(v, v, t) \\
\vdots \\
\end{bmatrix}
\]

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

\[
\begin{bmatrix}
\mathbf{x} \\
\mathbf{v}
\end{bmatrix}
\]

Concatenate \( \mathbf{x} \) and \( \mathbf{v} \) to make a 6-vector: position in phase space.

\[
\begin{bmatrix}
\dot{\mathbf{x}} \\
\dot{\mathbf{v}}
\end{bmatrix}
\]

Taking the time derivative: another 6-vector.

\[
\begin{bmatrix}
\mathbf{x} \\
\mathbf{v}
\end{bmatrix}
\frac{d}{dt}
\begin{bmatrix}
\mathbf{x} \\
\mathbf{v}
\end{bmatrix}
\]

A vanilla 1\(^{st}\)-order differential equation.
Particle structure

\[ \begin{bmatrix} x \\ v \\ f \\ m \end{bmatrix} \]

- Position in phase space
- position
- velocity
- force accumulator
- mass
Solver interface

\[
\begin{bmatrix}
x \\ v \\ f \\ m
\end{bmatrix}
\]

\[\text{getDim} \rightarrow [6]\]

\[\text{getState} \rightarrow \begin{bmatrix} x \\ v \end{bmatrix}\]

\[\text{setState} \rightarrow \begin{bmatrix} v \\ f/m \end{bmatrix}\]

\[\text{derivEval} \leftarrow \begin{bmatrix} x \\ v \end{bmatrix}\]
Particle systems

\[
\begin{bmatrix}
x_1 \\
v_1 \\
f_1 \\
m_1 \\
\end{bmatrix}
\begin{bmatrix}
x_2 \\
v_2 \\
f_2 \\
m_2 \\
\end{bmatrix}
\begin{bmatrix}
x_3 \\
v_3 \\
f_3 \\
m_3 \\
\end{bmatrix}
\cdots
\begin{bmatrix}
x_n \\
v \\
f \\
m_n \\
\end{bmatrix}
\]
Solver interface

- **particles**: n
- **time**: n

**get/setState**

**getDim**

**derivEval**

4 columns:
- \(x_1\)
- \(v_1\)
- \(x_2\)
- \(v_2\)

5 rows:
- \(\frac{f_1}{m_1}\)
- \(\frac{f_2}{m_2}\)
- \(\ldots\)
- \(\frac{f_n}{m_n}\)

Total states: \(6n\)
Forces

- Constant (gravity)
- Position/time dependent (force fields)
- Velocity-dependent (drag)
- N-ary (springs)
Gravity

Force law:
\[ f_{\text{grav}} = mG \]

\[ p->f += p->m \times F->G \]
Viscous drag

Force law:

\[ f_{\text{drag}} = -k_{\text{drag}} v \]
Damped spring

Force law:

\[ \mathbf{f}_1 = -k_s \mathbf{x} - \zeta_d \mathbf{r} \]

\[ \mathbf{f}_2 = -\mathbf{f}_1 \]

\[ \mathbf{r} = \text{rest length} \]
Particle systems with forces

\[ \begin{bmatrix}
    x_1 \\
    v_1 \\
    m_1 \\
\end{bmatrix} \quad \begin{bmatrix}
    x_2 \\
    v_2 \\
    m_2 \\
\end{bmatrix} \quad \ldots \quad \begin{bmatrix}
    x_n \\
    v_n \\
    m_n \\
\end{bmatrix} \]
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
derivEval Loop

1. Clear force accumulators

2. Apply forces to particles

3. Return \([v, f/m, \ldots]\) to solver
Solver interface

- particles
- n
- time

- get/setState
- getDim
- derivEval

\[ \begin{align*}
6n \\
\begin{array}{cccccc}
\mathbf{x}_1 & \mathbf{v}_1 & \mathbf{x}_2 & \mathbf{v}_2 & \cdots & \mathbf{v}_n \\
\mathbf{v}_1 & \frac{\mathbf{f}_1}{m_1} & \mathbf{v}_2 & \frac{\mathbf{f}_2}{m_2} & \cdots & \frac{\mathbf{f}_n}{m_n}
\end{array}
\end{align*} \]
Differential equation solver

\[
\begin{bmatrix}
\vdots \\
1 = 1 \\
\vdots
\end{bmatrix}
\]

Euler method:

\[
\begin{bmatrix}
x_{1}^{i+1} \\
v_{1}^{i+1} \\
\vdots \\
x_{n}^{i+1} \\
v_{n}^{i+1}
\end{bmatrix}
= \begin{bmatrix}
x_{1}^{i} \\
v_{1}^{i} \\
\vdots \\
x_{n}^{i} \\
v_{n}^{i}
\end{bmatrix}
+ \begin{bmatrix}
v_{1}^{i} \\
f_{1}^{i} / m_{1} \\
\vdots \\
v_{n}^{i} \\
f_{n}^{i} / m_{n}
\end{bmatrix}
\]
Bouncing off the walls

- Add-on for a particle simulator
- For now, just simple point-plane collisions
Normal and tangential components

\[ V_N = (N \cdot V)N \]

\[ V_T = V - V_N \]
Collision Detection

\[(X - P) \cdot N < \varepsilon \]  
Within \(\varepsilon\) of the wall

\[N \cdot V < 0\]  
Heading in
Collision Response

\[ \mathbf{V}_N \]

\[ \mathbf{V} \]

\[ \mathbf{V}_T \]

\[ \mathbf{V'} = \mathbf{V}_T - k_r \mathbf{V}_N \]

before

after
Artificial Fish
Related Research

- Determining dynamic parameters for cloth simulation
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