Particle dynamics
Unified Particle Physics for Real-Time Applications

Miles Macklin  Matthias Müller  Nuttapong Chentanez  Tae-Yong Kim

NVIDIA
• Second-order motion

• Particle system

• Forces

• Constraints

• Second order motion analysis (advanced)
Particle system

- Particles are objects that have mass, position, and velocity, but without spatial extent.
- Particles are the easiest objects to simulate, but they can be made to exhibit a wide range of objects.
A Newtonian particle

- First order motion is sufficient, if
  - a particle state only contains position
  - no inertia
  - particles are extremely light
- Most likely particles have inertia and are affected by gravity and other forces
- This puts us in the realm of second order motion
What is the differential equation that describes the behavior of a mass point?

\[ f = ma \]

What does \( f \) depend on?

\[ \ddot{x}(t) = \frac{f(x(t), \dot{x}(t))}{m} \]
Second-order ODE

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

This is not a first order ODE because it has second derivatives.

Add a new variable, \( v(t) \), to get a pair of coupled first order equations:

\[ \begin{cases} 
\dot{x} = v \\
\dot{v} = \frac{f}{m} 
\end{cases} \]
Phase space

\[
\begin{bmatrix}
  x \\
  v
\end{bmatrix} =
\begin{bmatrix}
  x_1 \\
  x_2 \\
  x_3 \\
  v_1 \\
  v_2 \\
  v_3
\end{bmatrix}
\]

Concatenate position and velocity to form a 6-vector: 
*position* in phase space

\[
\begin{bmatrix}
  \dot{x} \\
  \dot{v}
\end{bmatrix} = f(\begin{bmatrix}
  x \\
  v
\end{bmatrix}) = \begin{bmatrix} v \\
  f/m \end{bmatrix}
\]

First order differential equation: *velocity* in the phase space
A mass point attached to a spring obeys Hooke’s Law:

\[ f = -K(x - \bar{x}) \]

What is the ODE that describes this motion?
Integrate second-order ODE

Express a second-order motion in two first-order ODEs,
\[
\begin{pmatrix}
\dot{x} \\
\dot{v}
\end{pmatrix} = \begin{pmatrix}
0 & 1 \\
-K/m & 0
\end{pmatrix} \begin{pmatrix}
x \\
v
\end{pmatrix} + \begin{pmatrix}
0 \\
(K/m)\bar{x}
\end{pmatrix}
\]

Integrate both position and velocity via explicit Euler

\[
\begin{pmatrix}
x_1 \\
v_1
\end{pmatrix} = \begin{pmatrix}
x_0 \\
v_0
\end{pmatrix} + h \begin{pmatrix}
\dot{x}_0 \\
\dot{v}_0
\end{pmatrix}
\]

\[
= \begin{pmatrix}
x_0 \\
v_0
\end{pmatrix} + h \begin{pmatrix}
v_0 \\
K/m(\bar{x} - x_0)
\end{pmatrix}
\]
Quiz

• Integrate the same ODE using midpoint method.

\[
\begin{pmatrix}
\dot{x} \\
\dot{v}
\end{pmatrix} = 
\begin{pmatrix}
0 & 1 \\
-K/m & 0
\end{pmatrix}
\begin{pmatrix}
x \\
v
\end{pmatrix} + 
\begin{pmatrix}
0 \\
(K/m)\ddot{x}
\end{pmatrix}
\]
• Second-order motion

• Particle system

• Forces

• Constraints

• Second order motion analysis (advanced)
Particle structure

Particle

\( x \)  position
\( v \)  velocity
\( f \)  force accumulator
\( m \)  mass

\( a \) point in the phase space
Solver interface

System

Particle

x
v
f
m

Solver interface

GetDim
Get/Set State
Deriv Eval

Solver

6
x
v
f
m
Particle system structure

System

Particles

n  time

\[ x_1 \quad v_1 \quad f_1 \quad m_1 \]

\[ x_2 \quad v_2 \quad f_2 \quad m_2 \]

... 

\[ x_n \quad v_n \quad f_n \quad m_n \]
Particle system structure

System:
- particles
- \( n \)
- time

Solver Interface:
- GetDim
- Get/Set State
- Deriv Eval

Solver:
- \( 6n \)
- \( \frac{V_1}{f_1 m_1} \)
- \( \frac{V_2}{f_2 m_2} \)
- \( \ldots \)
- \( \frac{V_n}{f_n m_n} \)
Clear forces: loop over particles, zero force accumulator

Calculate forces: sum all forces into accumulator

Gather: loop over particles, copy v and f/m into destination array
• Second-order motion
• Particle system
• Forces
• Constraints
• Second order motion analysis (advanced)
Forces

- Constant gravity
- Position dependent force fields, springs
- Velocity dependent drag
Particle systems with forces

- System
  - Particles: $x_1, v_1, f_1, m_1; x_2, v_2, f_2, m_2; \ldots; x_n, v_n, f_n, m_n$
  - Forces: $F_1, F_2, \ldots, F_m$
  - Time: $t$
Force structure

- Unlike particles, forces are heterogeneous (type-dependent)
- Each force object “knows”
  - which particles it influences
  - how much contribution it adds to the force accumulator
Particle systems with forces

$\text{system}$

$\begin{align*}
\text{particles} & \quad \text{n} & \quad \text{time} \\
f_1 & \quad v_1 & \quad m_1 \quad x_1 \\
f_2 & \quad v_2 & \quad m_2 \quad x_2 \\
& \cdots & \cdots \cdots \\
f_n & \quad v_n & \quad m_n \quad x_n
\end{align*}$

$\begin{align*}
F_1 & \\
F_2 & \\
& \cdots \\
F_m
\end{align*}$
Gravity

Unary force: $\mathbf{f} = mG$

Exerting a constant force on each particle

```
p->f += p->m*F->G
```
At very low speeds for small particles, air resistance is approximately:

\[ f_{\text{drag}} = -k_{\text{drag}} v \]
Act on any or all pairs of particles, depending on their positions

\[ f_p = -k \frac{m_p m_q}{|l|^2} \frac{1}{|l|} \]

\[ f_q = -f_p \]

\[ l = x_p - x_q \]
Attraction

\[ f_p = -k \frac{m_p m_q}{|l|^2} \frac{1}{|l|} \]
Damped spring

\[ \mathbf{f}_p = - \left[ k_s(|\mathbf{l}| - r) + k_d \frac{\mathbf{i} \cdot \mathbf{l}}{|\mathbf{l}|} \right] \frac{1}{|\mathbf{l}|} \]

\[ \mathbf{f}_q = - \mathbf{f}_p \]

\[ \mathbf{l} = \mathbf{x}_p - \mathbf{x}_q \]
Damped spring

\[ f_p = - \left[ k_s (|l| - r) + k_d \frac{\vec{l} \cdot \vec{l}}{|l|} \right] \frac{1}{|l|} \]
For an ideal spring, what is the force it applies to two particles, \( p \) and \( q \), attached to it. Write down the pseudo code for its “apply_fun”.

\[
\begin{align*}
F_x &= f_p + f_q \\
F_v &= v_p + v_q \\
F_f &= f_p + f_q \\
F_m &= m_p + m_q \\
\end{align*}
\]
1. Clear force accumulators

\[
\begin{bmatrix}
x_1 \\
v_1 \\
f_1 \\
m_1 \\
x_2 \\
v_2 \\
f_2 \\
m_2 \\
\vdots \\
x_n \\
v_n \\
f_n \\
m_n \\
\end{bmatrix}
\]

2. Invoke apply\_force functions

\[
F_1 \quad F_2 \quad \ldots \quad F_m
\]

3. Return derivatives to solver

\[
\begin{bmatrix}
\dot{x} \\
\dot{v}
\end{bmatrix} = \begin{bmatrix}
v \\
f/m
\end{bmatrix}
\]
Euler’s method:

\[ x(t_0 + h) = x(t_0) + hf(x, t) \]

\[ x_{t+1} = x_t + h\dot{x}_t \]

\[ v_{t+1} = v_t + h\dot{v}_t \]
Euler step

1. Deriv Eval
2. Get/Set State
3. 
   \[ \mathbf{x}_{t+1} = \mathbf{x}_t + h \mathbf{x}_t \]
   \[ \mathbf{v}_{t+1} = \mathbf{v}_t + h \mathbf{v}_t \]
4. GetDim
5. Advance time
How to modify the algorithm to use midpoint method?
Example: freefall motion

- Solution is

\[ v(t) = v_0 + a_0 t \]

\[ x(t) = x_0 + v_0 t + \frac{1}{2} a_0 t^2 \]

- \( v(t) \) only needs 1st order accuracy, but \( x(t) \) demands 2nd order accuracy
Let particle $p$ start at position $x_0$ with velocity $v_0$, what is the state of $p$ after two time steps ($h$) using the midpoint method? Assume that gravity is the only force present in the scene.
• Second-order motion
• Particle system
• Forces
• Constraints
• Second order motion analysis (advanced)
• We will revisit collision when we talk about rigid body simulation
• For now, just simple point-plane collisions
Collision detection

Particle is on the legal side if

\[(x - p) \cdot N \geq 0\]

Particle is within \(\epsilon\) of the wall if

\[(x - p) \cdot N < \epsilon\]

Particle is heading in if

\[v \cdot N < 0\]
Collision response

Normal and tangential components

\[ v_N = (N \cdot v)N \]
\[ v_T = v - v_N \]
Collision response

Before collision

After collision

$v' = v_T - k_r v_N$

coefficient of restitution: \[0 \leq k_r < 1\]
Contact

Conditions for resting contact:
1. particle is on the collision surface
2. zero normal velocity

If a particle is pushed into the contact plane a contact force $f_c$ is exerted to cancel the normal component of $f$
• Second-order motion
• Particle system
• Forces
• Constraints

• Second order motion analysis (advanced)
Linear analysis

- Linearly approximate acceleration

\[
\frac{d}{dt} \begin{bmatrix} x \\ v \end{bmatrix} = f(\begin{bmatrix} x \\ v \end{bmatrix}, t) = A \begin{bmatrix} x \\ v \end{bmatrix} + a_0
\]

- Split up analysis into different cases
  - constant acceleration
  - linear acceleration

\[
\frac{d}{dt} \begin{bmatrix} x \\ v \end{bmatrix} = \begin{bmatrix} 0 & I \\ -K & -D \end{bmatrix} \begin{bmatrix} x \\ v \end{bmatrix} + a_0
\]
• Solution is

\[ v(t) = v_0 + a_0 t \]

\[ x(t) = x_0 + v_0 t + \frac{1}{2} a_0 t^2 \]

• \( v(t) \) only needs 1st order accuracy, but \( x(t) \) demands 2nd order accuracy
Linear acceleration

- When K (or D) dominates ODE, what type of motion does it correspond to?

\[
\frac{d}{dt} \begin{bmatrix} x \\ v \end{bmatrix} = \begin{bmatrix} 0 & I \\ -K & -D \end{bmatrix} \begin{bmatrix} x \\ v \end{bmatrix} = A \begin{bmatrix} x \\ v \end{bmatrix}
\]

- Need to compute the eigenvalues of A
Assume $\alpha$ is an eigenvalue of $A$, $\begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$ is the corresponding eigenvector

$$\begin{bmatrix} 0 & I \\ -K & -D \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \alpha \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

The eigenvector of $A$ has the form $\begin{bmatrix} u \\ \alpha u \end{bmatrix}$

Assuming $D$ is linear combination of $K$ and $I$ (Rayleigh damping)
That means $K$ and $D$ have the same eigenvectors
Linear acceleration

For any \( u \), if \[ \begin{bmatrix} u \\ \alpha u \end{bmatrix} \] is an eigenvector of \( A \), the following must be true
\[
\begin{bmatrix}
0 & I \\
-K & -D
\end{bmatrix}
\begin{bmatrix}
u \\
\alpha u
\end{bmatrix}
= \alpha
\begin{bmatrix}
u \\
\alpha u
\end{bmatrix}
\]

Now assume \( u \) is an eigenvector for both \( K \) and \( D \)

\[-\lambda_k u - \alpha \lambda_d u = \alpha^2 u\]

\[\alpha = -\frac{1}{2}\lambda_d \pm \sqrt{\left(\frac{1}{2}\lambda_d\right)^2 - \lambda_k}\]
Eigenvalue approximation

- If $D$ dominates
  \[ \alpha \approx -\lambda_d, 0 \]
  - exponential decay
- If $K$ dominates
  \[ \alpha \approx \pm \sqrt{-1} \sqrt{\lambda_k} \]
  - oscillation
Analysis

• Constant acceleration (e.g. gravity)
  • demands 2nd order accuracy for position
• Position dependence (e.g. spring force)
  • demands stability, oscillatory motion
  • looks at imaginary axis
• Velocity dependence (e.g. damping)
  • demands stability, exponential decay
  • looks at negative real axis
Explicit methods

- First-order explicit Euler method
  - constant acceleration: bad (1st order)
  - position dependence: very bad (unstable)
  - velocity dependence: ok (conditionally stable)
- RK3 and RK4
  - constant acceleration: great (high order)
  - position dependence: ok (conditionally stable)
  - velocity dependence: ok (conditionally stable)
Implicit methods

- Implicit Euler method
  - constant acceleration: bad (1st order)
  - position dependence: ok (stable but damped)
  - velocity dependence: great (monotone)

- Trapezoidal rule
  - constant acceleration: great (2nd order)
  - position dependence: great (stable and no damp)
  - velocity dependence: good (stable, not monotone)