Particle dynamics
• Particle overview

• Particle system

• Forces

• Constraints

• Second order motion analysis
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.
Particle animation

- Each particle has a position, mass, and velocity
  - maybe color, age, temperature
- Seeded randomly at start
  - maybe some created each frame
- Move each frame according to physics
- Eventually die when some condition met
Sparks from a campfire

- Add 2-3 particles at each frame
  - initialize position and temperature randomly
- Move in specified turbulent smoke flow and decrease temperature as evolving
- Render as a glowing dot
- Kill when too cold to glow visibly
• Simplest rendering: color dots
• Animated sprites
• Deformable blobs
• Transparent spheres
• Shadows
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

\[
\begin{align*}
\dot{x} &= v \\
\dot{v} &= f/m
\end{align*}
\]
Phase space

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

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

\[
\begin{bmatrix}
\dot{\mathbf{x}} \\
\dot{\mathbf{v}}
\end{bmatrix} = \begin{bmatrix}
\mathbf{v} \\
\frac{\mathbf{f}}{m}
\end{bmatrix}
\]

First order differential equation: *velocity* in the phase space
• Particle overview
  • Particle system
• Forces
• Constraints
• Second order motion analysis
Particle structure

Particle

\[ x \quad \text{position} \]
\[ v \quad \text{velocity} \]
\[ f \quad \text{force accumulator} \]
\[ m \quad \text{mass} \]

\[ \text{a point in the phase space} \]
Solver interface

system

particle

x
v
f
m

solver interface

GetDim
Get/Set State
Deriv Eval

solver

6
x
v
f
m

Deriv Eval
Particle system structure

System

Particles

n  time

\[ \begin{align*}
  x_1 & \quad x_2 & \quad \ldots & \quad x_n \\
  v_1 & \quad v_2 & \quad \ldots & \quad v_n \\
  f_1 & \quad f_2 & \quad \ldots & \quad f_n \\
  m_1 & \quad m_2 & \quad \ldots & \quad m_n
\end{align*} \]
Particle system structure

- **System**
  - Particles
  - Time
  - n

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

- **Solver**
  - 6n
  - \( \frac{x_1}{f_1} \)
  - \( \frac{v_1}{m_1} \)
  - \( \frac{x_2}{f_2} \)
  - \( \frac{v_2}{m_2} \)
  - \( \cdots \)
  - \( \frac{x_n}{f_n} \)
  - \( \frac{v_n}{m_n} \)
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
- Particle overview
- Particle system
- Forces
- Constraints
- Second order motion analysis
Forces

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

- System
  - Particles
    - $n$ particles
    - $t$: time
  - Forces
    - $F_1$, $F_2$, ..., $F_m$
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

system

particles

n time

forces

\begin{align*}
  \mathbf{x}_1 & \quad \mathbf{v}_1 & \quad \mathbf{f}_1 \\
  m_1 & & \\
  \vdots & & \vdots \\
  \mathbf{x}_n & \quad \mathbf{v}_n & \quad \mathbf{f}_n \\
  m_n & & \\
\end{align*}

\begin{align*}
  \mathbf{F}_1 & & \\
  \mathbf{F}_2 & & \\
  \vdots & & \\
  \mathbf{F}_m & & \\
\end{align*}
Gravity

Unary force: \( \mathbf{f} = m \mathbf{G} \)

Exerting a constant force on each particle

\[ p \rightarrow \mathbf{f} += p \rightarrow m \times F \rightarrow \mathbf{G} \]
At very low speeds for small particles, air resistance is approximately:

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

Act on any or all pairs of particles, depending on their positions

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

\[ f_q = -f_p \]

\[ l = x_p - x_q \]
Attraction

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

\[ f_p = - \left[ k_s (|l| - r) + k_d \frac{\hat{l} \cdot l}{|l|} \right] \frac{l}{|l|} \]

\[ f_q = -f_p \]

\[ l = x_p - x_q \]
The forces acting on the particle system are given by:

\[
f_p = -\left[ k_s (|l| - r) + k_d \frac{\dot{l} \cdot l}{|l|} \right] \frac{l}{|l|}
\]
1. Clear force accumulators

2. Invoke apply_force functions

3. Return derivatives to solver

\[
\begin{bmatrix}
\dot{x} \\
\dot{v}
\end{bmatrix} =
\begin{bmatrix}
v \\
\frac{f}{m}
\end{bmatrix}
\]
ODE solver

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. Advance time
4. GetDim
5. Apply forces

system

particles

time

solver interface

solver

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

$v_{t+1} = v_t + h\dot{v}_t$

$x_1 \quad x_2 \quad \ldots \quad x_n$

$v_1 \quad v_2 \quad \ldots \quad v_n$

$\frac{V_1}{m_1} \quad \frac{V_2}{m_2} \quad \ldots \quad \frac{V_n}{m_n}$
• Particle overview
• Particle system
• Forces
• Constraints
• Second order motion analysis
Particle Interaction

• We will revisit collision when we talk about rigid body simulation
• For now, just simple point-plane collisions
Collision detection

Normal and tangential components

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

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

Before collision

\[ \mathbf{v} \]

\[ \mathbf{v}_T \]

\[ \mathbf{v}_N \]

After collision

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

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 $\mathbf{f}_c$ is exerted to cancel the normal component of $\mathbf{f}$
• Particle overview
• Particle system
• Forces
• Constraints

• Second order motion analysis
Linear analysis

• Linearly approximate acceleration

\[ a(x, v) \approx a_0 - Kx - Dv \]

• Split up analysis into different cases
  • constant acceleration
  • linear acceleration
Constant acceleration

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

$$\mathbf{a}(x, v) = -Kx - Dv$$

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

• Need to compute the eigenvalues of $A$
Linear acceleration

Assume $\alpha$ is an eigenvalue of $A$, \[
\begin{bmatrix}
0 & I \\
-K & -D
\end{bmatrix}
\begin{bmatrix}
u_1 \\
u_2
\end{bmatrix}
= \alpha
\begin{bmatrix}
u_1 \\
u_2
\end{bmatrix}
\]

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

Often, $D$ 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 $A$, 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 $D$

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

\[
\alpha = -\frac{1}{2} \lambda_d \pm \sqrt{(\frac{1}{2} \lambda_d)^2 - \lambda_k}
\]
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 but low or zero damping
  - 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)
What’s next?
• How do we enforce constraints on the particles?

• Read (optional): Particle animation and rendering using data parallel computation, SIG90, Karl Sims