Particle Systems

- phenomena:
  - fireworks
  - dust, water spray, mud, smoke, fire
  - hair, fur, grass, cloth
  - crowds, flocks

- need to consider:
  - (A) when and where particles start
  - (B) rules that govern motion
  - (C) how to render

- classic papers:
  - W. Reeves, "Particle Systems"
  - K. Sims, "Particle Animation and Rendering"

- attributes:
  - position
  - velocity
  - orientation
  - colour
  - radius
  - mass
  - age
  - temperature
  - fuel
  - ...

(A) Seeding Particles

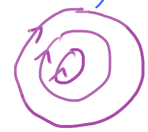
- creation and deletion of particles
- where?
  - randomly within volume or on surface
  - at a point where an event occurs, e.g., cresting wave
- when?
  - at start of simulation
  - at each frame
  - at the occurance of an event

(B) Particle Motion

First order motion

- particles move according to specified velocity field:

$$\frac{dx}{dt} = v(x,t)$$



e.g., tornado

- break into time steps and integrate:

$$\vec{v}_i = v(x_i, t)$$

$$\vec{x}_i^{new} = \vec{x}_i + \Delta t\, \vec{v}_i$$

- velocity field can come from:
  - predesigned elements:
  - "noise", i.e., curl noise
  - from a simulation of fluid or air

Second order motion

- particles move according to specified accelerations, moves according to underlying forces

$$\frac{d^2\vec{x}}{dt^2} = \frac{\sum F}{m} \qquad \sum F = m \cdot a$$

- integration:

$$\vec{a}_i = \frac{1}{m_i} \sum F_i$$

$$\vec{V}_i^{new} = \vec{V}_i^{old} + \Delta t \, \vec{a}_i$$

$$\vec{x}_i^{new} = \vec{x}_i^{old} + \Delta t \, \vec{V}_i^{new}$$

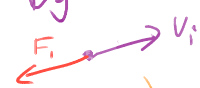
] explicit Euler integration

"Euler integration" simple, easy to implement but needs small time steps due to stability problems.

alternative: implicit integration methods
→ offer more stability for more complexity

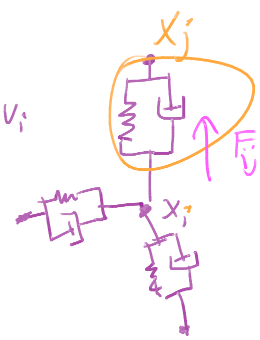
- particle forces:
  - gravity: $\vec{F}_i = m\vec{g}$

  - drag: $\vec{F}_i = -k\vec{V}_i$



  - spring and damper:

$$\vec{F}_{ij} = -k_s(\vec{x}_i - \vec{x}_j) - k_d(\vec{V}_i - \vec{V}_j)$$



  - spring with non-zero rest length:

$$F_i = -k_s\left[\frac{\|x_i - x_j\|}{L_{ij}} - 1\right]\frac{(x_i - x_j)}{\|x_i - x_j\|}$$

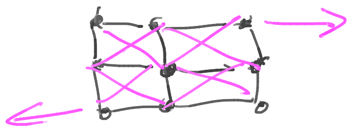
magnitude
m: % stretch
m = 0 rest length
m = +1 stretch of 100% ie, length = 2 x rest length
m = -1 compressed.

direction of force

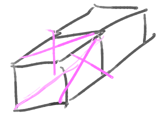
need to get:
- right layout
- right stiffnesses
- right integration

- animate objects using particles connected with springs
  - hair, flexible rod

  - cloth: 2D mesh of springs & dampers

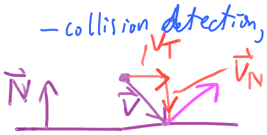


  - Jello



- liquids using particles:
  - smoothed particle hydrodynamics (SPH)

- collision detection, collision responce



$\vec{V} = \vec{V_N} + \vec{V_T}$

$V_N = (\vec{V} \cdot \vec{N})\vec{N}$

$V_T = V - V_N$

$\vec{V}_{new} = \vec{V_T} - \epsilon \vec{V_N}$

$\epsilon = 0$ inelastic

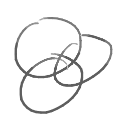
$\epsilon = 1$ perfectly elastic

(c) Particle Rendering

- ~~dot or circle~~ for each particle
- kernel function or "splat"

render using variable opacity (α in OpenGL)

- multiple overlapping particles at a pixel:
  - add
  - composite together using depth order.



- motion blur

Frame n n+1 n+2

draw line or polygon from old position to new position.
- "simulates" shutter being open.

- implicit surfaces for water, mud
  - explicitly builds a surface surrounding the particles

.. "level set method"

$\sum_i f(x - x_i)$

surface to render $\Sigma f(\ ) =$ contour value $\rightarrow \alpha$

$\Sigma f < \alpha$

$\Sigma f > \alpha$

use "marching cubes" to build surface

Particles – Equations of Motion (EOM)

state $\vec{X} = \begin{bmatrix} p_x \\ p_y \\ p_z \\ v_x \\ v_y \\ v_z \end{bmatrix}$

EOM: $\frac{d\vec{X}}{dt} = \begin{bmatrix} \dot{p}_x \\ \dot{p}_y \\ \dot{p}_z \\ \dot{v}_x \\ \dot{v}_y \\ \dot{v}_z \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \\ v_z \\ \sum F_x / m \\ \sum F_y / m \\ \sum F_z / m \end{bmatrix}$

Integrate:

$p = p + \dot{p}\Delta t$

$v = v + \dot{v}\Delta t$