Course Plan

135

Changing the value of dt in Verlet

(whenever it's not constant)

Problem:

if dt now changes to a new dt'

then, all $\mathbf{p}_{old}$ must be updated to some $\mathbf{p}'_{old}$

Find $\mathbf{p}'_{old}$:

$$\vec{v} = (\mathbf{p}_{now} - \mathbf{p}_{old})/dt$$
$$\vec{v} = (\mathbf{p}_{now} - \mathbf{p}'_{old})/dt'$$

current velocity $\vec{v}$ and position $\mathbf{p}_{now}$ must not change

$$\Longrightarrow$$

$$\mathbf{p}'_{old} = \mathbf{p}_{now} \cdot (dt - dt')/dt + \mathbf{p}_{old} \cdot dt'/dt$$

136

Velocity damping in Verlet

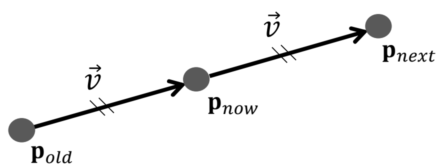
- Velocity at next frame: $\vec{v} = (\mathbf{p}_{next} - \mathbf{p}_{now})/dt$ (implicit)
- We want to multiply $\vec{v}$ a factor c_{DAMP} (e.g. 0.98 obtained as $(1\text{-}dt\cdot c_{\text{DRAG}})$)
  - before applying accelerations
- We can do that using a more general formula for $\mathbf{p}_{next}$

$$\mathbf{p}_{next} = 2 \cdot \mathbf{p}_{now} - 1 \cdot \mathbf{p}_{old} + dt^2 \cdot \vec{a}$$

$$\mathbf{p}_{next} = (1 + c_{\text{DAMP}}) \cdot \mathbf{p}_{now} - c_{\text{damp}} \cdot \mathbf{p}_{old} + dt^2 \cdot \vec{a}$$

138

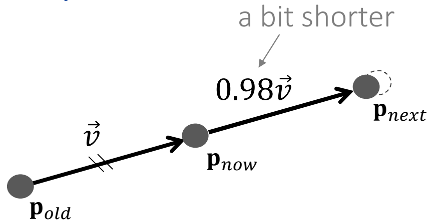
Velocity damping in Verlet (geometric interpretation)



$$\mathbf{p}_{next} = 2 \cdot \mathbf{p}_{now} - 1 \cdot \mathbf{p}_{old}$$

Equivalently,
$\mathbf{p}_{next}$ is an extrapolation of $\mathbf{p}_{now}$, $\mathbf{p}_{old}$:

$$\mathbf{p}_{next} = mix(\mathbf{p}_{old}, \mathbf{p}_{now}, 2)$$



$$\mathbf{p}_{next} = 1.98 \cdot \mathbf{p}_{now} - 0.98 \cdot \mathbf{p}_{old}$$

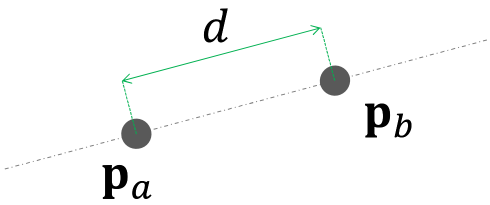
Equivalently,
$\mathbf{p}_{next}$ is a different extrapolation of $\mathbf{p}_{now}$, $\mathbf{p}_{old}$:

$$\mathbf{p}_{next} = mix(\mathbf{p}_{old}, \mathbf{p}_{now}, 1.98)$$

139

Example of positional constraint: equidistance constraint

*«Particles **a** and **b** must stay at a fixed distance **d** »*



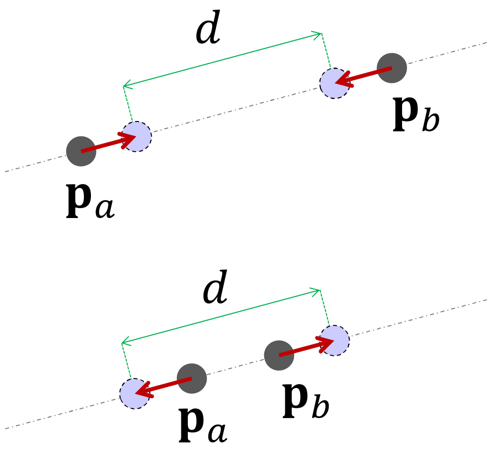
I want that… $\|\mathbf{p}_a - \mathbf{p}_b\| = d$

140

Enforce equidistance constraints
(assuming equal masses for now)

if $\|\mathbf{p}_a - \mathbf{p}_b\| > d$

if $\|\mathbf{p}_a - \mathbf{p}_b\| < d$



141

Enforce equidistance constraints: pseudo code

```
Vector3 pa, pb; // curr positions of a,b
float d;        // distance (to enforce)

Vector3 v = pa - pb;
float currDist = v.length;

v /= currDist;  // normalization of v

float delta = currDist - d ;

pa += ( 0.5 * delta) * v;
pb -= ( 0.5 * delta) * v;
```

assuming equal mass, we move each particle *half the way* (see later for the more general case)

142

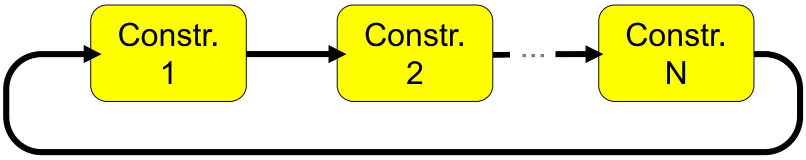
Compare: equidistance constraints vs. springs

- Similar
  - they both mean: these 2 particles "want to be" at *this* distance (not more, not less)
    (some constant scalar parameter *D*)
- but different
  - equidistance constraint:
    - applied during constraint enforcement
    - directly affects positions
    - models a rigid rod between the two particles
      - of a given length
    - sometimes called a "HARD" constraint
  - spring:
    - applied during force evaluation step
    - affects forces, therefore accelerations
    - models a deformable spring between the two particles
      - of a given length
    - sometimes called a "SOFT" constraint
- A physic engine can combine them in one object!

143

Enforcing sets of constraints

- There are many constraints to impose:
  when you solve one → maybe you break another!
- Simultaneous enforcement: computationally expensive
- Practical & easy solution: just enforce them in cascade
  (similar in concept to Gauss-Seidel solvers):



Repeat until convergence (= max error below threshold)
...but at most for *N* times! (reminder: our simulation is *soft* real-time)

144

Enforcing a set of constraints one after the other (in cascade)

- The whole loop for imposing the constraints happen in the constraint enforcement phase on one physics step
- Notes about convergence:
  - needed iterations (typically) few: e.g. 1 ~ 10 (efficient!).
  - if convergence not reached within a given number of steps: never mind, next frames will fix it (it's fairly robust)
  - (it is never reached, if constraints are contradictory)
  - Optimization (to reduce the number of needed iterations): solve the most unsatisfied constraints first
- ⚠ Problem: it's a sequential approach! ☹
  - parallelized versions (similar to Jacobi solvers) are possible
  - they have a worse convergence in practice
    (they require more iterations), but each iteration is faster

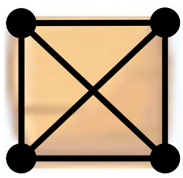
145

Compounds of particles disguised as rigid bodies

146

We can combine equidistance constraints to obtain rigid objects!

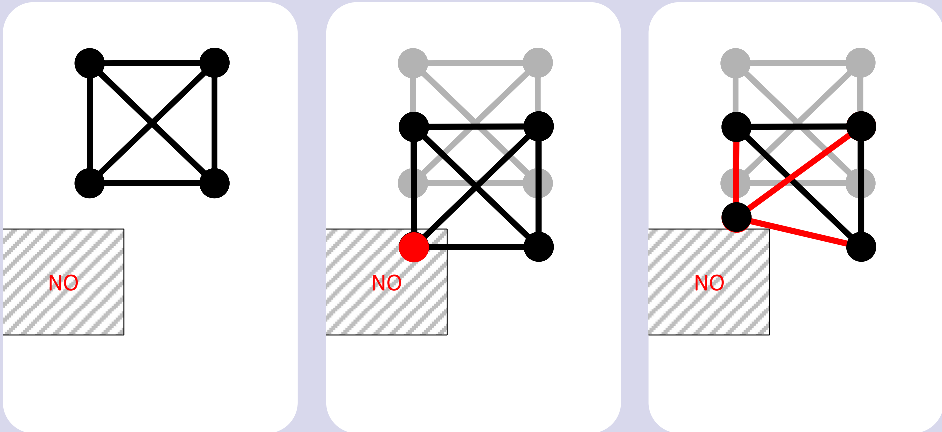
- Rigid body dynamics as emerging behavior
  - without explicitly keeping track their orientation, angular vel, angular acc., etc.



A box in 2D?
(rigid object)
A configuration of:
- 4 particles
- 6 equidistance constraints

147

Example



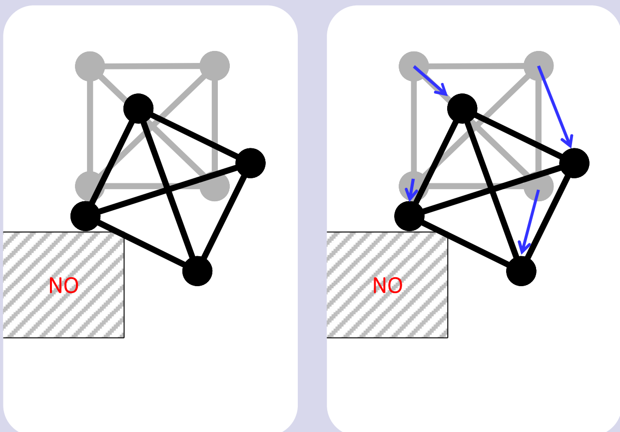
FRAME 0

FRAME 1
before constraints

FRAME 1
after 1st constraint

148

Example



FRAME 1
after all constraints
multiple times

FRAME 1
resulting
(implicit) velocities

In total: the “box”,
under gravity + collision

- had rotated
- gained angular velocity (will keep rotating by inertia)

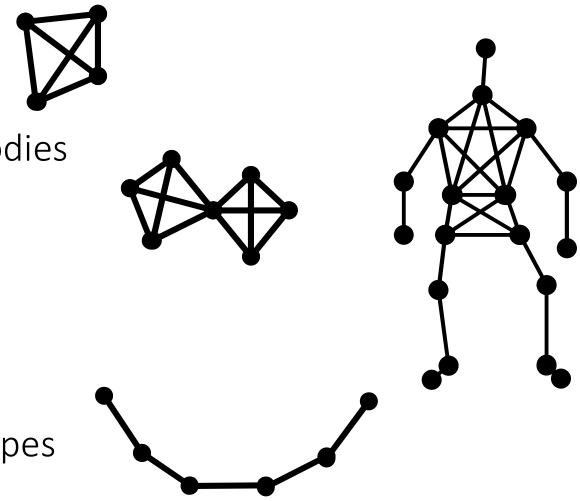
even the system does not (explicitly) handle rotations or
angular velocities

(works in 3D as well!)

149

We can combine equidistance constraints to obtain…

- Rigid bodies
- Articulated bodies
- Ragdolls
- Cloth
- Non-elastic ropes
- …and more



151

Positional constraint (in general terms)

- A predicate defined on the position(s) of a number of particles (usually, a small number: 1 - 4)

$$\mathcal{C}: (\mathbf{p}_a, \mathbf{p}_b, \mathbf{p}_c, \ldots) \rightarrow \{ \, true \, , false \, \}$$

- For example, the equidistance constraints is

$$\mathcal{C}(\mathbf{p}_a, \mathbf{p}_b) \Leftrightarrow \|\mathbf{p}_a - \mathbf{p}_b\| = k_{CONST}$$

- They can be an equality ($=$) or an inequality ($\leq$ or $\geq$)

152

Equality positional constraint: examples

- Equidistance constraint (the one we have seen):
  «these N particles must stay at distance k»
  - E.g: because they are linked my a metal rod of length k
- Fixed positions:
  «this particle must stay in position $\mathbf{p}_\mathrm{a}$ *»*
  - the particle is "pinned" in position
  - trivial to impose, but still useful!
- Coplanarity / collinearity:
  «these N particles must stay on a line / on a plane»

153

Equality positional constraint: other examples

- Volume preservation:
  "The volume delimited by the squishy ballon defined by these particles is a constant k_{CONST} *" (e.g. because it's filled with water)*
- How to impose it:
  1. Estimate current total volume v
  2. uniform scale the entire object by factor $\sqrt[3]{f_{\mathrm{CONST}} / v}$ around its barycenter

154

Inequality positional constraints: example

- "please don't sink below the ground"
  assuming the ground is the plane Y = 0

$$C(\mathbf{p}_a) \Leftrightarrow \mathbf{p}_a.y \geq 0$$

- Trivial to impose:
  just set the y to 0, if it is < 0

155

Inequality positional constraints: example

- "this particle must stay above this fixed (and arbitrary) plane"
  - For example, because the plane is a solid unmovable slab
  - The plane is given by a point on it $\mathbf{p}_q$ and its normal $\hat{n}_q$

$$C(\mathbf{p}_a) \Leftrightarrow (\mathbf{p}_a - \mathbf{p}_q) \cdot \hat{n}_q \geq 0$$

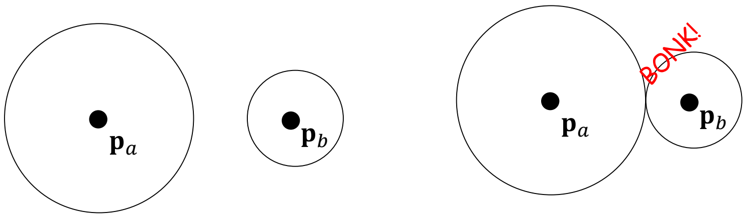
156

Inequality positional constraints: example

- These two particles must be at least k_{CONST} apart

  $$\mathcal{C}(\mathbf{p}_a, \mathbf{p}_b) \Leftrightarrow \|\mathbf{p}_a - \mathbf{p}_b\| \geq k_{CONST}$$

  - For example, because they are the centers of two rigid spheres and k_{CONST} is the sum of their radii
  - part of collision handling (see next lecture)



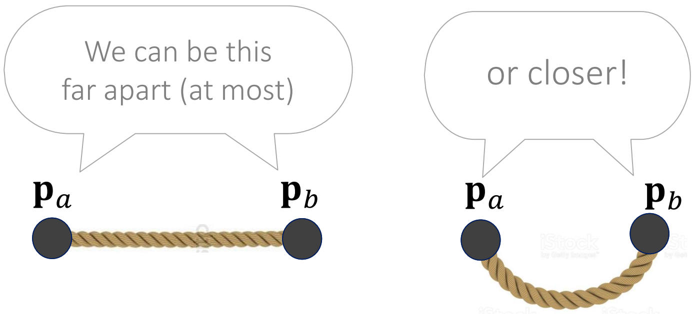
157

Inequality positional constraints: example

- These two particles must be at most k_{CONST} apart

  $$\mathcal{C}(\mathbf{p}_a, \mathbf{p}_b) \Leftrightarrow \|\mathbf{p}_a - \mathbf{p}_b\| \leq k_{CONST}$$

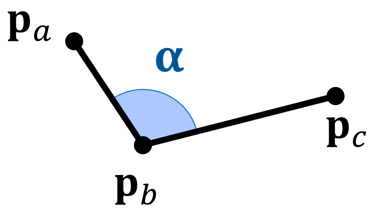
  - For example, because they are tied by an inextensible rope that has length k_{CONST} (but can fold)



158

Inequality positional constraints: example

- Angle constraints, e.g. $\boldsymbol{\alpha} < \boldsymbol{\alpha}_{\max}$
  with $\boldsymbol{\alpha}$ the angle between $\mathbf{p}_a$, $\mathbf{p}_b$ and $\mathbf{p}_b$, $\mathbf{p}_c$
  - e.g., on joints: *«elbows cannot bend backward»*
  - (a constraint between three particles!)



159

Enforcing one positional constraint (in general terms)

- Inequality constraint:
  1. *Test:* does the inequality already hold?
  2. If so: do nothing
  3. If not: enforce it as an equality (=) instead (see below!)
- Equality constraint:
  - All involved particles must be displaced
    from that current position, so that it now holds
  - There can be infinite ways to achieve this!
    Question: Which one to pick?

160

Enforcing one equality constraint:
(assuming for now all particles have same mass)

- Answer:
  minimize the sum of *squared* displacements
  (with respect to current position)
- For each kind of constraint, we need to find the minimizer analytically
  ("analytically" = in closed form = "with formulas" = "solving a simple math problem on paper")
  - That's what we did for the equality constraint

161

Enforcing one equality constraint
(assuming for now all particles have same mass)

- We want to enforce a constraint $\mathcal{C}$ on particles $\mathrm{a}, \mathrm{b}, \mathrm{c}, \ldots$ currently in positions $\mathbf{p}_\mathrm{a}, \mathbf{p}_\mathrm{b}, \mathbf{p}_\mathrm{c}, \ldots$

$$\mathcal{C}: (\mathbf{p}_\mathrm{a}, \mathbf{p}_\mathrm{b}, \mathbf{p}_\mathrm{c}, \ldots) \rightarrow \{\, true\,, false \,\}$$

- We must apply the displacements $\vec{d_\mathrm{a}}, \vec{d_\mathrm{b}}, \vec{d_\mathrm{c}}$ that are the

$$\underset{\vec{d_\mathrm{a}}, \vec{d_\mathrm{b}}, \vec{d_\mathrm{c}}, \ldots}{\operatorname{argmin}} \left(\|\vec{d_\mathrm{a}}\|^2 + \|\vec{d_\mathrm{b}}\|^2 + \|\vec{d_\mathrm{c}}\|^2 + \cdots \right)$$

$$\text{such that } \mathcal{C}\left(\mathbf{p}_\mathrm{a} + \vec{d_\mathrm{a}}, \mathbf{p}_\mathrm{b} + \vec{d_\mathrm{b}}, \mathbf{p}_\mathrm{c} + \vec{d_\mathrm{c}}, \ldots\right)$$

among all the choices that satisfy this,
we want the one which minimizes this

162

Enforcing one equality constraint
(in general)

- We want to enforce a constraint $\mathcal{C}$ on particles a , b , c, ... in positions $\mathbf{p}_a$, $\mathbf{p}_b$, $\mathbf{p}_c$ and with masses m_a, m_b, m_c, ...

$$\mathcal{C}: (\mathbf{p}_a, \mathbf{p}_b, \mathbf{p}_c, \dots) \rightarrow \{ true , false \}$$

- We must apply the displacements $\overrightarrow{d_a}$, $\overrightarrow{d_b}$, $\overrightarrow{d_c}$ found by:

$$\underset{\overrightarrow{d_a}, \overrightarrow{d_b}, \overrightarrow{d_c}, \dots}{\text{argmin}} \left(m_a \|\overrightarrow{d_a}\|^2 + m_b \|\overrightarrow{d_b}\|^2 + m_c \|\overrightarrow{d_c}\|^2 + \cdots \right)$$

$$\text{such that } \mathcal{C}\left(\mathbf{p}_a + \overrightarrow{d_a}, \mathbf{p}_b + \overrightarrow{d_b}, \mathbf{p}_c + \overrightarrow{d_c}, \dots\right)$$

among all the choices that satisfy this,
we want the one which minimizes this

163

Example: solve the "please don't sink under this plane"

$$\text{C}(\mathbf{p}_a) \Leftrightarrow (\mathbf{p}_a - \mathbf{p}_q) \cdot \hat{n}_q \geq 0$$

a point on plane (const)
plane normal (const)

- We need to find displacement $\overrightarrow{d_a}$ as:

$$\underset{\overrightarrow{d_a}}{\text{argmin}} \left(m_a \|\overrightarrow{d_a}\|^2 \right)$$

$$\text{such that } \left(\mathbf{p}_a + \overrightarrow{d_a} - \mathbf{p}_q\right) \cdot \hat{n}_q \geq 0$$

- And the solution (in closed form) is...

164

In pseudocode

```
Vector3 pa; // curr positions of a
float ma;   // mass (no effect here)
Vector3 pq; // point on the plane
Vector3 nq; // normal of the plane (unit)

Vector3 v = pa - pq;
float currDist = Vector3.dot( v , n );

if (currDist < 0.0)
   pa -= currDist * n; // just project!
else {} // constrain ok, nothing to do
```

165

Example: the equidistance constraint (for unequal masses)

$$\mathcal{C}(\mathbf{p}_a, \mathbf{p}_b) \Leftrightarrow \|\mathbf{p}_a - \mathbf{p}_b\| = k_{CONST}$$

- With partcle masses m_a , m_b
- We need to the displacements $\overrightarrow{d_a}$, $\overrightarrow{d_b}$ found by minimizing:

$$\underset{\overrightarrow{d_a}, \overrightarrow{d_b}}{\operatorname{argmin}} \left(m_a \left\|\overrightarrow{d_a}\right\|^2 + m_b \left\|\overrightarrow{d_b}\right\|^2 \right)$$
$$\text{such that } \left\|(\mathbf{p}_a + \overrightarrow{d_a}) - (\mathbf{p}_b + \overrightarrow{d_b})\right\| = k_{CONST}$$

- And the solution (in closed form) is…

166

Example: the equidistance constraint (for unequal masses)

```
Vector3 pa, pb; // curr positions of a,b
float ma, mb;   // masses of a,b
float d;        // distance (to enforce)

Vector3 v = pa - pb;
float currDist = v.length;

v /= currDist;  // normalization of v

float delta = currDist - d ;

/* solutions of the minimization: */
pa += ( mb/(ma+mb) * delta) * v;
pb -= ( ma/(ma+mb) * delta) * v;
```

167

Observe and verify

- The way we have seen to impose…
  - The “fixed position” constraint
  - The “equidistance” constraint
  - The “stay above ground” constraint
  - Etc.

  are the ones that minimizes the mass-weighted squared displacements of the particles
  - (the mass is not always relevant)

168

Position Based Dynamics (PBD) summary

- A general approach for computing dynamics
- Ingredients:
  1. Use Verlet integration **on particles**
     - their velocities are implicit
     - changes in positions induce changes in velocities
  2. Implement positional constraints **on particles** (e.g., equidistance constraint) to model things like:
     - Rigid bodies (their rotational speed is an emerging feature!)
     - Articulated / non rigid bodies
     - Basic collision detection

169

Not forces: summary

$$\ldots \quad \vec{f} = \text{function}(\mathbf{p}, \ldots) \quad \ldots$$

not in here

- We have seen many kinds of real-world forces that are modelled by things that aren't "forces" in our simulation:
  - Frictions
    - *In reality:* a ("dissipative") force contrasting motion
    - Can be simulated by: drag / velocity damp
  - Violent sudden events, such as impacts
    - *In reality:* a very strong force that is sustained for a very short time << *dt*
    - E.g.: hitting a ball with a mace
    - Must be simulated by: impulses
  - Resistance forces
    - *In reality:* a force that contrast and nullifies an external force (e.g. gravity)
    - E.g.: what prevents your computer from falling through the table RN
    - Can be simulated by: positional constraints

170