Course Plan

lec. 1: Introduction
lec. 2: Mathematics for 3D Games
lec. 3: Scene Graph
lec. 4: Game 3D Physics +
lec. 5: Game Particle Systems
lec. 6: Game 3D Models
lec. 7: Game Textures
lec. 9: Game Materials
lec. 8: Game 3D Animations
lec. 10: Networking for 3D Games
lec. 11: 3D Audio for 3D Games
lec. 12: Rendering Techniques for 3D Games
lec. 13: Artificial Intelligence for 3D Games

174

Enforcing a positional constraint: the general case with masses.

- Check: does the equality/inequality hold?
- If so, nothing to do!
- Else:
  - All positions must be displaced a bit, so that it does
  - Infinite ways to achieve this. Which one to pick?
  - Answer:
    minimize the sum of *squared* displacements
    (with respect to current position)
    weighted by particle masses
  - Find the minimizer by analytically
    solving simple math problems
    (“analytically” = in closed form = “with formulas”)

175

Enforcing positional constraints in the general case: formal problem definition

- 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, …
- $\mathcal{C}$ defined as an equality/inequality of $\mathbf{p}_a$, $\mathbf{p}_b$, $\mathbf{p}_c$, … :

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

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

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

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

among all the choices that satisfy this,

we want the one which minimizes this

176

Example: the equidistance constraint

- To enforce the constraint "particles a and b must stay at distance L"
  - input: current positions $\mathbf{p}_a$, $\mathbf{p}_b$
  - input: masses m_a , m_b
- We need to the displacements $\vec{d_a}$, $\vec{d_b}$ found by minimizing:

$$\underset{\vec{d_a}, \vec{d_b}}{\operatorname{argmin}} \left(m_a \|\vec{d_a}\|^2 + m_b \|\vec{d_b}\|^2 \right)$$

$$\text{such that } \left\| (\mathbf{p}_a + \vec{d_a}) - (\mathbf{p}_b + \vec{d_b}) \right\| = L$$

- And the solution (in closed form) is…

177

Equidistance constraints: solution for non-equal 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;
```

178

Rigid-bodies as compounds of particles + constraints

- Interesting/rich/useful set of "emerging behaviors" (they just automatically happen) :
  - rigid, deformable, jointed objects
    - made of particles + hard constraints
  - their angular velocities
    - rotation around proper axis
  - their barycenter
  - their momentum of inertia
    - angular velocity is maintained
  - somewhat believable bounces on "impacts"
    - for more control: impact impulses can be added (see collisions)

you don't need to compute or store these

consequence of constraints disallowing compenetration

179

Rigid-body as (particles + constraints) Challenges

- Approximations are introduced
  - e.g.: mass is concentrated in a few locations
- Scalability issues
  - many constraints to enforce, many particles to track
- Some of the info which is kept *implicit* is needed by the rest of the game engine
  - and must therefore be extracted ☹
  - example: the transform (position + orientation) of the "rigid body" is needed to render the associated mesh
  - similarly: angular speed, barycenter pos, velocity…

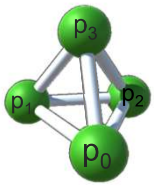
180

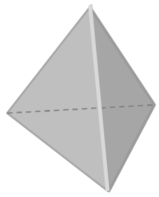
Particles + constraint, or rigid bodies?

- Rigid-body based systems:
  - explicitly compute dynamics for rigid bodies
  - also store their current orientation + angular velocity
  - update them (just like position + velocity)
- Particles-based systems with PBD:
  - only compute dynamics for particles
  - rigid (or deformable, or jointed) bodies as an emerging behavior
- Mixed systems:
  - dynamically swap between the two representations for rigid bodies
  - how to pass from one to the other?

181

How to extract…





Particle Compound

STATIC (on initializ.)

masses $m_0 \dots m_3$
initial positions $r_0 \dots r_3$

DYNAMIC (during simulation)

positions $p_0 \dots p_3$
velocities $v_0 \dots v_3$

? →
? →
← ?

Rigid Body

STATIC

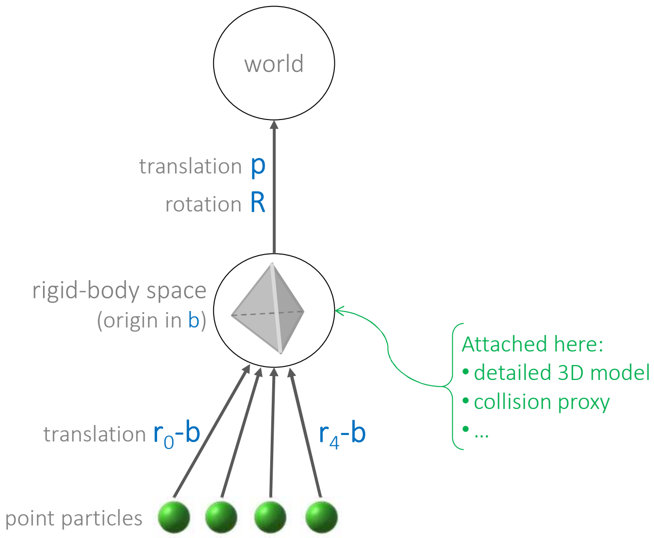
mass m
barycenter b (in loc. space)
moment of inertia I (matrix)

DYNAMIC

position p (of barycenter)
velocity v
rotation (i.e. orientation) R
angular velocity a

182

Scene graph interpretation



183