

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: **3D Audio** for 3D Games ●
- lec. 11: **Networking** for 3D Games ●
- lec. 12: **Artificial Intelligence** for 3D Games ●
- lec. 13: **Rendering Techniques** for 3D Games ●

164

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** ← you don't need to compute or store these
 - rotation around proper **axis** ←
 - their **barycenter** ←
 - their **momentum of inertia** ← consequence of constraints disallowing compensation
 - angular velocity is maintained
 - somewhat believable **bounces on “impacts”** with ground
 - for more control: **impact impulses** can be added (see collisions)

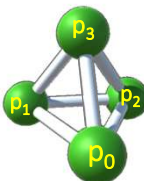
165

Rigid-body as particles + constraints: issues

- **Approximations** are introduced
 - e.g.: mass is concentrated in a few locations only
- **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 ☹
 - mainly: the **transform** (position + orientation) of the virtual “rigid body” is needed to render the associated meshes
 - or: its **velocity**, **angular velocity**, total **mass** may be needed for... gameplay reasons (e.g. damage), graphics (motion blur), etc

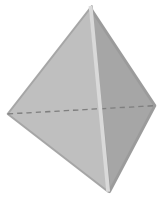
166

How to extract...



Particle Compound

STATIC (initialization)	masses $m_0 \dots m_N$ initial positions $\mathbf{r}_0 \dots \mathbf{r}_N$?	STATIC (initialization)
DYNAMIC (during simulation)	positions $\mathbf{p}_0 \dots \mathbf{p}_N$ velocities $\mathbf{v}_0 \dots \mathbf{v}_N$?	DYNAMIC (during simulation)



(Virtual) Rigid Body

STATIC (initialization)	mass m barycenter $\bar{\mathbf{r}}$ (local sp. origin?) moment of inertia \mathbf{M} (matrix)		STATIC (initialization)
DYNAMIC (during simulation)	position \mathbf{p} (of barycenter) velocity \mathbf{v} (linear) rotation \mathbf{R} (i.e. orientation) angular velocity \mathbf{a}		DYNAMIC (during simulation)

168

Marco Tarini

Università degli studi di Milano

2

For example:

Particle Compound

STATIC

masses $m_0 \dots m_N$

DYNAMIC

positions $\mathbf{p}_0 \dots \mathbf{p}_N$

velocities $\vec{v}_0 \dots \vec{v}_N$

Rigid Body

(total) mass m

(current) position $\bar{\mathbf{p}}$ (of barycenter)

(linear) velocity \vec{v}

A

B

C

A:

$$m = \sum m_i$$

B:

$$\bar{\mathbf{p}} = \frac{1}{m} \sum m_i \mathbf{p}_i$$

C:

$$\vec{v} = \frac{1}{m} \sum m_i \vec{v}_i$$

Questions:

How to find / update the...

- rotation
- angular velocity
- moment of inertia matrix (const)

...of the rigid body?

171

Local Space

where things are fixed, because body is rigid!

Transform

?

rotate

?

translate

1

scale

World Space

172

Extract the (global) transform of the “virtual” rigid-body 1/2



- Input:
 - particle “rest” positions $\mathbf{r}_0 \dots \mathbf{r}_n$ in local space (fixed)
 - their masses $m_0 \dots m_n$ (fixed)
 - their current positions $\mathbf{p}_0 \dots \mathbf{p}_n$ in global space (this frame)
- Output: current transform \leftarrow a roto-translation, by definition
 - Scaling: 1 \leftarrow it’s rigid, duh
 - Translation: difference between the barycenters: $\bar{\mathbf{p}} - \bar{\mathbf{r}}$
 - Rotation: see the next slide
(just for completeness)
(the algorithm is not part of the exam)

173

Extract the (global) transform of the “virtual” rigid-body 2/2

this part is just for completeness,
not part of the exam.
But maybe useful in life...

- Find the rotation (as a 3x3 matrix):
 - Step 1 : remove the respective barycenters from pts
 $\vec{\mathbf{r}}_i \leftarrow \mathbf{r}_i - \bar{\mathbf{r}}$ and $\vec{\mathbf{p}}_i \leftarrow \mathbf{p}_i - \bar{\mathbf{p}}$
 - Step 2: find 3x3 rot matrix \mathbf{R} such that $\forall i. \mathbf{R}(\vec{\mathbf{r}}_i) \cong \vec{\mathbf{p}}_i$
as closely as possible (weighted by the mass);
that is, such that $\sum m_i \|\mathbf{R}(\vec{\mathbf{r}}_i) - \vec{\mathbf{p}}_i\|^2 \rightarrow \min$

as objects rotate
around their
barycenter!

The theory gives us this:

first, compute $\mathbf{M} = \sum m_i (\vec{\mathbf{r}}_i \otimes \vec{\mathbf{p}}_i)$

then, find \mathbf{R} as the closest *rotation* matrix to \mathbf{M}

using Principal Values Decomposition (PVD) – look it up

External product: $\vec{\mathbf{v}} \otimes \vec{\mathbf{w}} = \vec{\mathbf{v}} \vec{\mathbf{w}}^T$

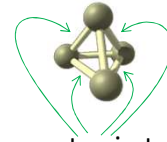
174

Rigid-bodies as particle compounds:



a shortcut

Only for rigid-bodies.
Not: articulated models, ropes,
or other things obtainable
by combining constraints



In the constraint enforcement step....

- instead of enforcing many equidistance constraints,
- enforce a single **combined** positional constraint:
«*please keep these n particles in a rigid configuration*»
 - the one defined by fixed local space positions $\mathbf{r}_0 \dots \mathbf{r}_n$
- How to impose it:
 - Step 1: compute rigid roto-translation $(\mathbf{R}, \vec{\mathbf{t}})$
from current positions $\mathbf{p}_0 \dots \mathbf{p}_n$ (see above)
 - Step 2: reposition each particle:
$$\mathbf{p}_i \leftarrow \mathbf{R}(\mathbf{r}_i) + \vec{\mathbf{t}}$$

175

Summary:

two ways to handle rigid-bodies

- As a compound of particles, with PBD
 - Bonus: we can also handle...
deformable bodies, articulated bodies
 - Bonus: mixes well with many other useful constraints
 - Cost: need to extract status of the «virtual» rigid body
 - Cost: mass concentrated at particles (approximation)
- With Rigid Bodies dynamics
 - With explicit rotation / angular velocity
 - The integration methods (Euler, Leapfrog, Verlet) can be adapted
- Or, mixed systems:
 - Requires to convert (dynamically) between the two

176