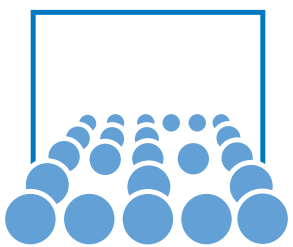
PSE Game Physics

Session (6)
Angular momentum, microcollisions, damping

Atanas Atanasov, Philipp Neumann, Martin Schreiber

27.05.2011



Outline

Angular velocity - starting point

- With our quaternions, we are able to setup the rotation of an object.
- Question: **How to express the angular velocity?**
- Possible answer: Specify **angle(s) of rotation** and **specify velocity** at which the point is rotating around the axis
- There's a **better solution** - describe the angular velocity with a simple vector:

$$\vec{v}_r = s \cdot \vec{r}$$

with $\vec{v}_r$ being the angular velocity, s the angular speed and $\vec{r}$ the normalized axis of rotation.

- Remark: s corresponds to the angle of rotation α per second.

Angular velocity - multiple velocities

- Looking at the translational velocity, we are allowed to simply sum different velocity vectors:

$$\vec{v} = \vec{v_1} + \vec{v_2}$$

- Using angular velocity vectors, we are also allowed to simply add both vectors:

$$\vec{v_r} = \vec{v}_{r1} + \vec{v}_{r2}$$

Angular velocity - from acceleration to velocity

- For the translational acceleration, we used the **explicit Euler** to compute the updated velocity:

$$\vec{v}(t + \Delta t) = \vec{v}(t) + \Delta t \cdot \vec{a}$$

and to update the position of the object (using the *translate()* method in the engine):

$$\vec{x}(t + \Delta t) = \vec{x}(t) + \Delta t \cdot \vec{v}(t + \Delta t)$$

- For a given angular acceleration $\vec{\alpha}$, we can update the angular velocity with a similar method:

$$\vec{v}_r(t + \Delta t) = \vec{v}_r(t) + \Delta t \cdot \vec{\alpha}$$

From angular velocity to quaternions

- Remember Session (3):
- Given rotation axis (x, y, z) and angle $\theta = |\vec{v}_r|\Delta t$

$$\rightarrow q_r := \begin{pmatrix} w \\ i \\ j \\ k \end{pmatrix} = \begin{pmatrix} \cos\frac{\theta}{2} \\ x\sin\frac{\theta}{2} \\ y\sin\frac{\theta}{2} \\ z\sin\frac{\theta}{2} \end{pmatrix} \in \mathbb{R} \times \mathbb{R}^3$$

- Computing a single timestep given an angular velocity vector $\vec{v}_r$, a quaternion can be set up representing the rotation for a single timestep.
- Thus, you are able to compute the new quaternion of the next timestep:

$$q(t + \Delta t) = q(t) \cdot q_r(t)$$

How to move on...

- So far we (should) know:
  - What an angular velocity vector is.
  - How to rotate an object during each timestep.
    - Update angular velocity of objects using angular acceleration.
    - Update the orientation by converting the angular rotation to a quaternion and applying the rotational quaternion to the one describing the orientation of the object.
- Next: How to modify the angular velocity of an object OR how to obtain $\vec{\alpha}$?

Torque (Drehmoment)

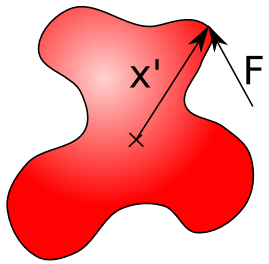


Figure: Force acting on object

- Let's assume to have a directed force $\vec{F}$ (e. g. created by a spring) exceeded on the point $\vec{x'}$ relative to the center of mass of an object.
- Then the torque is computed by

$$\vec{\tau} = \vec{x'} \times \vec{F}$$

Multiple Torques

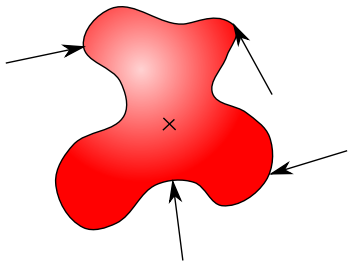
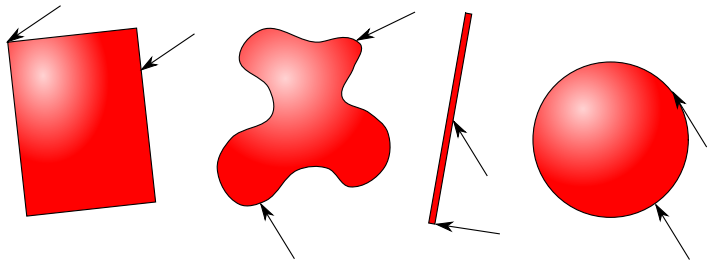


Figure: Multiple forces acting on an object

- We are not only interested in a single torque acting on an object
- When multiple forces are exceeded on an object, they have to be combined somehow
- According to the summation of angular velocities, we can just add them up:

$$\vec{\tau} = \sum_i \vec{x}_i' \times \vec{F}_i$$

Inertia (Massentraegheit)



- Change of angular velocity due to torque clearly depends on the **shape of the object**!
- This information can be expressed in form of the **inertia tensor**.
- For spheres and boxes, these tensors are

$$I_{\text{Sphere}} = \begin{pmatrix} \frac{2}{3}mr^2 & . & . \\ . & \frac{2}{3}mr^2 & . \\ . & . & \frac{2}{3}mr^2 \end{pmatrix} \qquad I_{\text{Box}} = \begin{pmatrix} \frac{1}{12}m(w^2+d^2) & . & . \\ . & \frac{1}{12}m(h^2+d^2) & . \\ . & . & \frac{1}{12}m(h^2+w^2) \end{pmatrix}$$

See `http://en.wikipedia.org/wiki/List_of_moments_of_inertia` for more inertia tensors

Inertia (1/2)

- Similar to the translational force $\vec{F} = m \cdot \vec{a}$, we get a formula for the torque depending on the angular acceleration α and the inertia tensor:

$$\vec{\tau} = I\vec{\alpha}$$

- Since we are **interested in computing the change in the angular acceleration**, we are allowed to rewrite the equation:

$$\vec{\alpha} = I^{-1}\vec{\tau}$$

- Warning: This formula is valid only in object space!!!
  However, the forces exerted on the object might only be given in world space.

  $\Rightarrow$ convert to world space

Inertia (2/2)

- The torque τ is given as a **vector**
- Object $\rightarrow$ World space: inverse transposed matrix (see session (2))
- World $\rightarrow$ Object space: transposed matrix
- Putting everything together, the angular acceleration vector is computed by

$$\vec{\alpha} = M^{-T} I^{-1} M^{T} \vec{\tau}$$

with model matrix M

Angular momentum

- We start by looking at our “simple“ **translational case**. There we computed the momentum by

$$\vec{p}_l = m \cdot \vec{v}$$

- For angular rotations, we express the **angular momentum** in a similar way:

$$\vec{p}_r = I \cdot \vec{v}_r$$

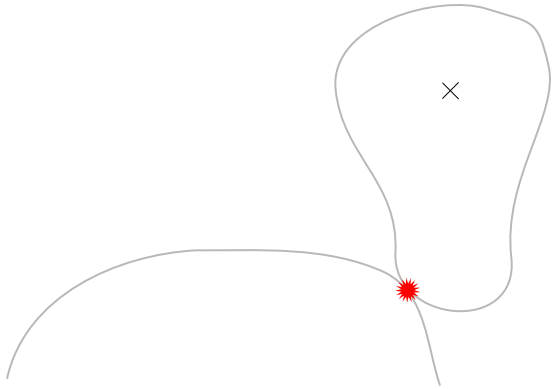
- The angular momentum created by a **translational momentum** acting at point $\vec{x}'$ on the object is given by

$$\vec{p}_r = \vec{x}' \times \vec{p}_l$$

- Finally we can express the change of angular velocity due to momentum by

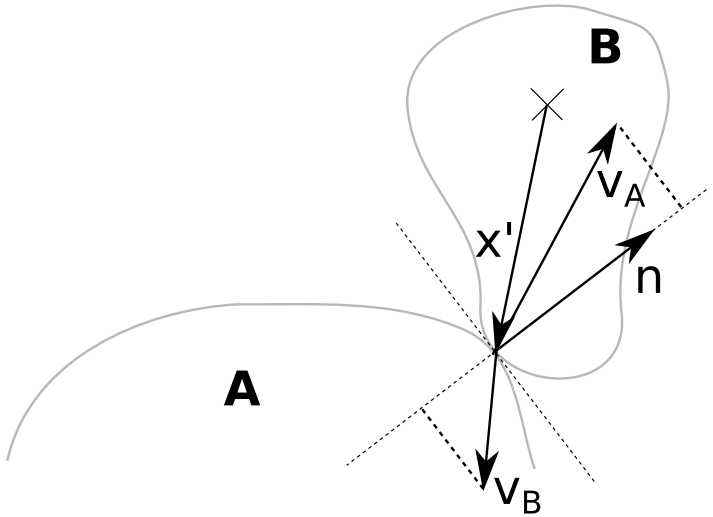
$$\vec{v}_r = I^{-1}\vec{p}_r = I^{-1}(\vec{x}' \times \vec{p}_l)$$

Angular collision impulse



- From the translational case, we know that due to the collision, the “velocity has to change“ to match $v_s = C_r \cdot v_c$ and other physical properties.
- In particular, the **velocity of both collision points** has to match this formula.
- Our goal is to **compute the impulse which has to be applied to the collision points** to match v_s.

Step 1) Compute the closing velocity

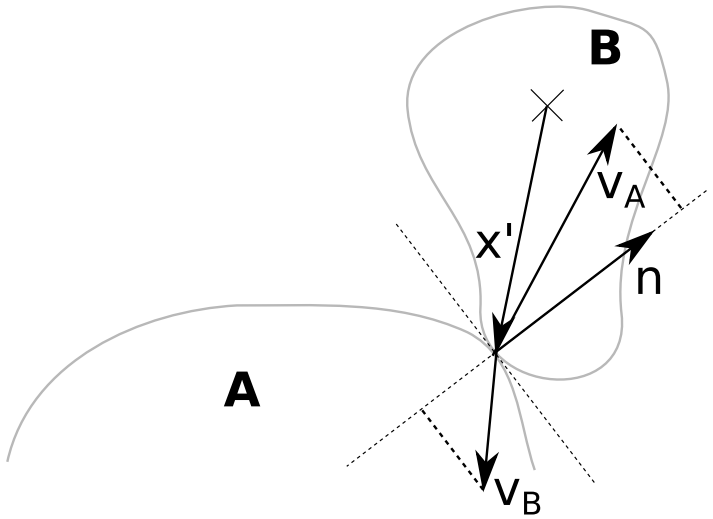


- The velocity of a point on the surface of the object is given by

$$\vec{v} = \vec{v}_r \times \vec{x}' + \vec{v}_l$$

with *angular velocity* $\vec{v}_r$, $\vec{x}'$ as the vector from the center of mass to the collision point and $\vec{v}_l$ the *(linear) translational velocity*.

Step 1) Compute the closing velocity (con'd)



- Since **only the velocity towards the collision normal** may be modified (without friction!), this fraction is computed by using the dot product:

$$v_c = \vec{n} \cdot \vec{v}_A - \vec{n} \cdot \vec{v}_B$$

- Now we are able to compute the separating velocity with the same formulas as for the purely translational case.

Step 2) Compute the change in velocity for unit impulse

- It's problematic to rearrange the formulas to directly compute the amount of impulse which has to be applied to gain a specific separating velocity.
- Therefore we utilize an important property of the impulse (similar to Newtons 3rd law):
  When an impulse is exceeded to an object during a collision, an impulse of same magnitude but opposite direction is exceeded to the other object.
- Important: The direction of the unit impulse of one object aims to the opposite direction of the unit impulse applied to the other object.
- The change in velocity is computed based on applying a **unit impulse** to both objects which is explained sucessively in the next slides.

Step 2) Compute the change in velocity for unit impulse (con'd)

- The **unit impulse is parallel to the collision normal**.
- Computing the change in translational velocity for the unit impulse is(/should be) known from previous sessions:

$$\Delta \vec{v}_l = \vec{n} \cdot m^{-1}$$

- The change in angular velocity is computed using

$$\Delta \vec{v}_r = \underbrace{M^{-T} I^{-1} M^T}_{\text{Inertia to world space}} \underbrace{\vec{x}'}_{\text{lever arm}} \times \underbrace{\vec{n}}_{\text{unit impulse}}$$

- Finally, we compute the change of velocity in the specific point on the object:

$$\Delta \vec{v} = \Delta \vec{v}_l + \Delta \vec{v}_r \times \vec{x}'$$

- This has to be computed for both objects!

Step 3) Computing the impulse to apply (1/2)

- So far **we know the change of velocity** for both collision points.
- Since the separating velocity only accounts towards the normal, we may **consider only the speed in direction of the normal**:

$$\Delta s = \Delta \vec{v} \cdot \vec{n}$$

- The **change in separating velocity** of both objects when **exceeding a unit impulse to both objects** is given by simply adding Δs of both objects (Depending on your implementation, maybe you have to change the sign of one component!)
- Knowing the **closing velocity** and the coefficient of restitution, we can compute the **separating velocity** using the formula

$$C_r = \frac{v_s}{v_c}$$

- By **knowing the separating velocity**, we can now compute the **fraction** f **of the unit impulse** (i.e. the fraction of the separating velocity) to apply.

Step 3) Computing the impulse to apply (2/2)

- Current state: We know the fraction of the unit impulse which has to be applied to both objects to gain the separating velocity!
- There are 2 ways how to apply this impulse:
  - Either we apply the impulse which is now well knowned by multiplying the fraction f with the unit impulse which has to be applied to gain the separating velocity or
  - we use directly the fraction f to compute the change of translational and angular velocity for each object by using our **previously computed variables** $\Delta\vec{v}_l$ and $\Delta\vec{v}_r$.

$$\vec{v}_l = \vec{v}_l + f \cdot \Delta\vec{v}_l$$

$$\vec{v}_r = \vec{v}_r + f \cdot \Delta\vec{v}_r$$

Some more hints & comments:

- Always consider the **different spaces** (model or world space) in which you're doing the computations. Sometimes you'll have to do some transformations.
- To gain better results, also the **interpenetration** has to account for the **rotations** of objects.