Game Physics

Game and Media Technology
Master Program - Utrecht University

Dr. Nicolas Pronost

Rigid body physics

Particle system

- Most simple instance of a physics system
  - Each object (body) is a particle
  - Each particle have forces acting upon it
    - Constant, *e.g.* gravity
    - Position dependent, *e.g.* force fields
    - Velocity dependent, *e.g.* drag forces
    - Event based, *e.g.* collision forces
    - Restrictive, *e.g.* joint constraint
  - So net force is a function $F(p_o, v, a, m, t, \dots)$

Particle system

- Use the equations of motion to find the position of each particle at each frame
- At the start of each frame
  - Sum up all of the forces for each particle
  - From these forces compute the acceleration
  - Integrate into velocity and position

Mass

- The mass is the measure of the amount of matter in the volume of an object

$$m = \int_V \rho \, dV$$

where ρ is the density at each location in the object volume V

- We can also state that the mass is a measure of an object's resistance to motion or a change of motion
  - the larger mass, the more difficult it is to set in or change the motion of an object

Mass

- For a 3D object, the mass is therefore the integral over its volume along the three dimensions

$$m = \int\int\int \rho(x, y, z)\, dx\, dy\, dz$$

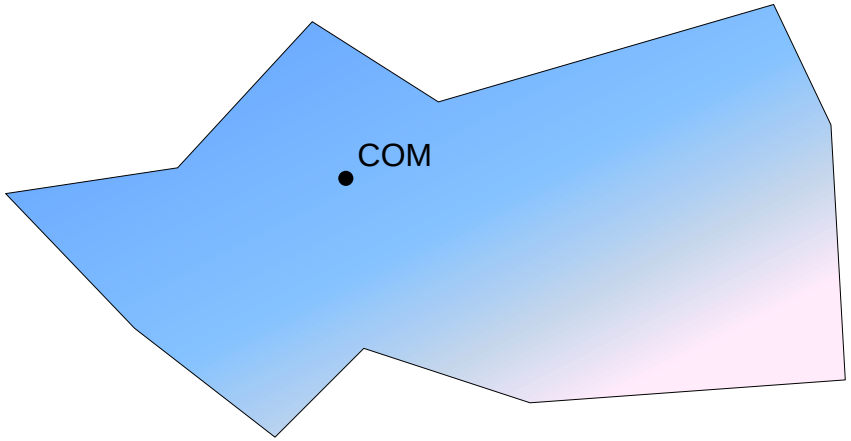
- For uniform density objects (rigid bodies usually are), the mass is then

$$m = \rho * V$$

where ρ is the density of the object and V its volume

Center of mass

- The center of mass (COM) is the point at which all the mass can be considered to be ‘concentrated’
  - obtained from the first moment, *i.e.* mass times distance
  - point of ‘balance’ of the object
  - if uniform density, COM is also the centroid



Center of mass

- Coordinate of the COM

$$COM = \frac{1}{m}\int_V \rho(p) * p \, dV$$

where p is the position at each location in V

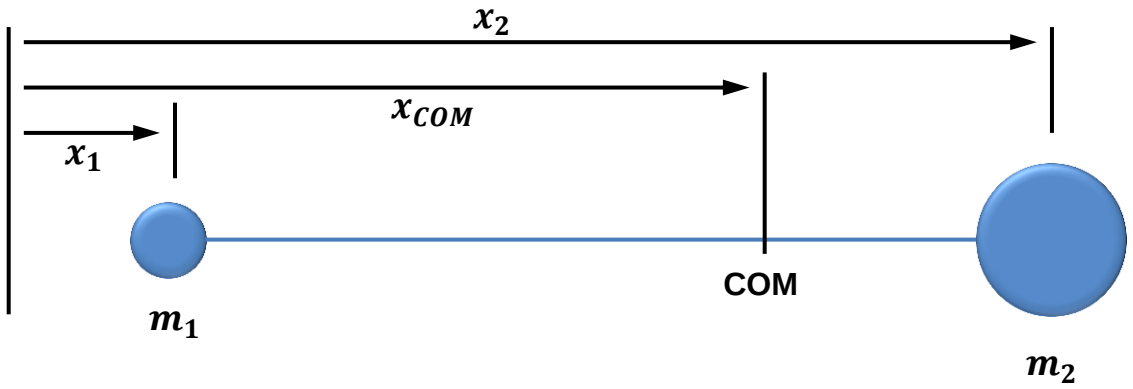
- For a body made of particles

$$COM = \frac{1}{m}\sum_{i=1}^{n} m_i p_i$$

where m_i is the mass of each particle p_i in the body

Center of mass

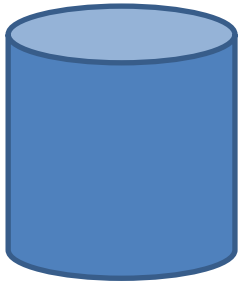
- Example for a body made of two particles in 1D



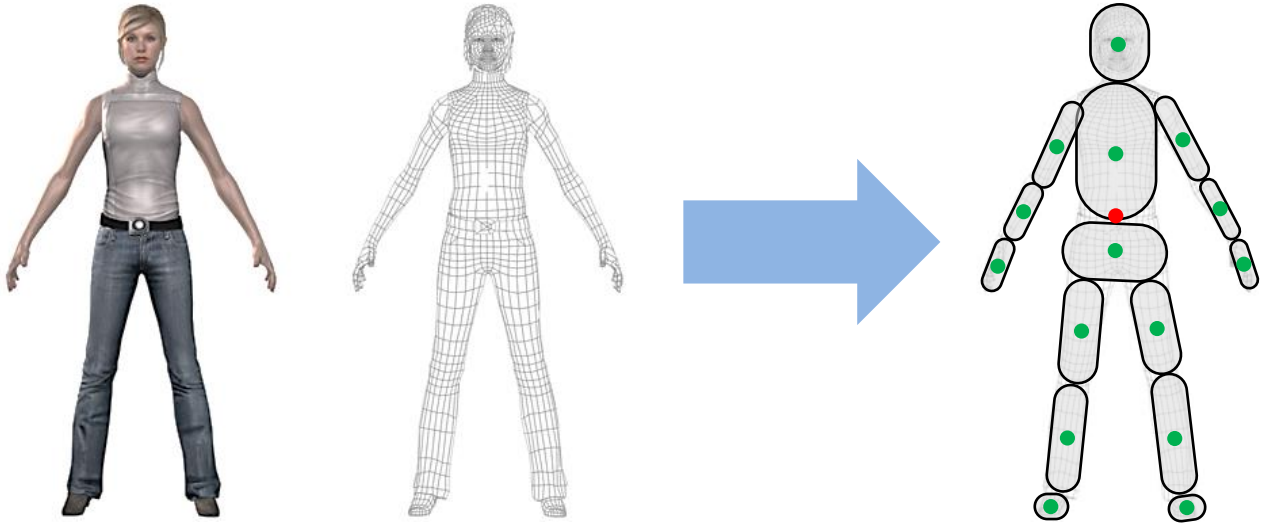
$$x_{COM} = \frac{m_1 x_1 + m_2 x_2}{m_1 + m_2}$$

Center of mass

- Quite easy to determine for primitive shapes



- But what about complex surface based models?

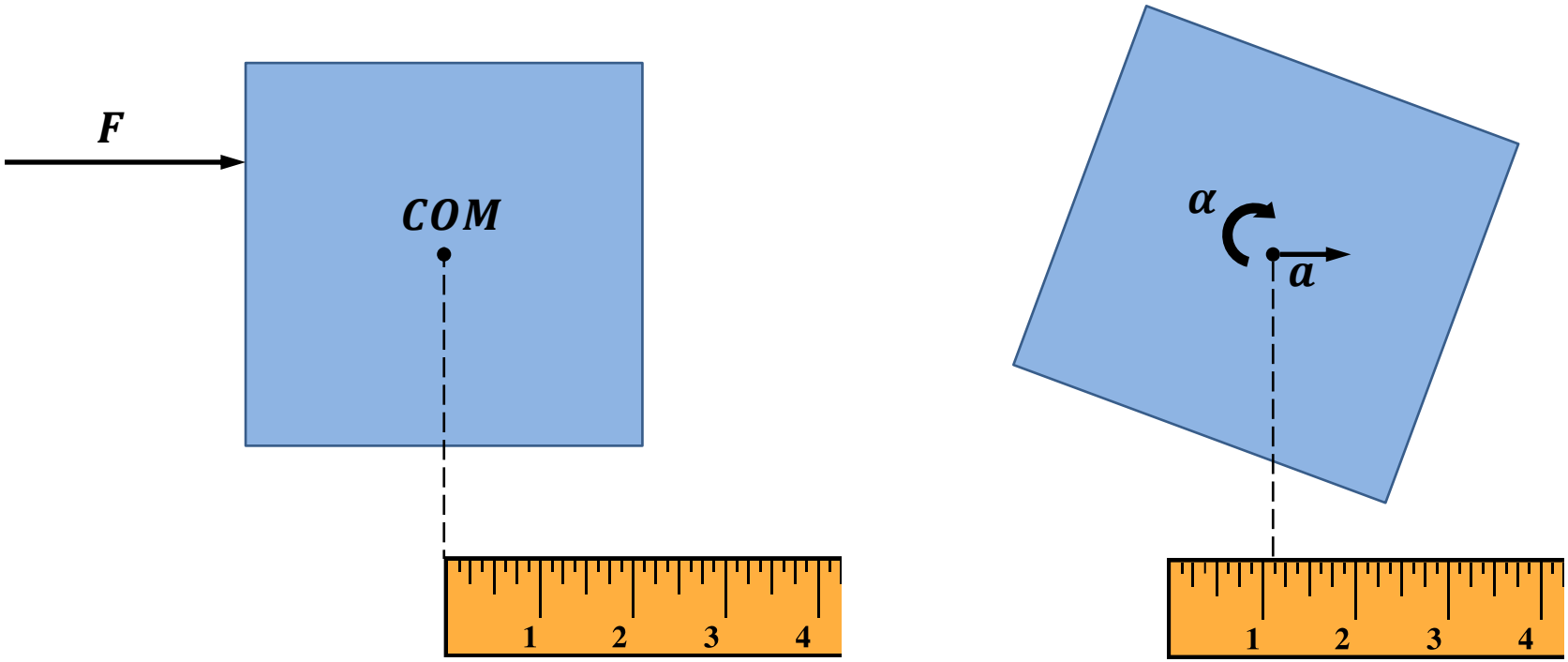


Rigid body

- In rigid body physics, the motion of an object is not summarized to the translational motion of its center of mass
- A rigid body is considered as a system of particles remaining at fixed distances from each other with no relative translation or rotation among them

Rigid body

- A force can be applied anywhere on the object, producing also a rotational motion



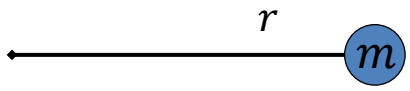
Moment of inertia

- The moment of inertia of a rigid body is a measure of how much the mass of the body is spread out
- It is a measure of the rigid body's ability to resist change in rotational motion
- It is defined with respect to a specific rotation axis
- We define r as the distance between any point in the object and the axis of rotation

Moment of inertia

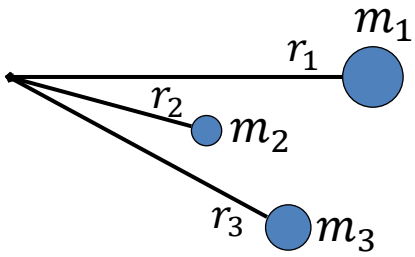
- For a mass point:

$$I = m * r^2$$



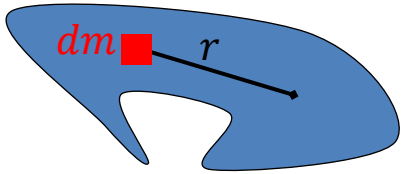
- For a collection of mass points:

$$I = \sum_i m_i {r_i}^2$$



- For a continuous mass distribution:

$$I = \int_M r^2 \, dm$$



Momentum and inertia

- Remember that the angular momentum is

$$L = I * \omega = r \times p = r \times mv = m(r \times (\omega \times r))$$

- If we look at the set of all elementary mass elements in the body, we have

$$L = \int_M r \times (\omega \times r)\, dm$$

Momentum and inertia

- Let's define $r = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$ and $\omega = \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix}$
- So we have

$$L = \int \begin{pmatrix} (y^2 + z^2)\omega_x - xy\omega_y - xz\omega_z \\ -yx\omega_x + (z^2 + x^2)\omega_y - yz\omega_z \\ -zx\omega_x - zy\omega_y + (x^2 + y^2)\omega_z \end{pmatrix} dm$$

Momentum and inertia

- Let's define

$$I_{xx} = \int (y^2 + z^2) dm \qquad I_{xy} = I_{yx} = \int (xy) dm$$

$$I_{yy} = \int (z^2 + x^2) dm \qquad I_{xz} = I_{zx} = \int (xz) dm$$

$$I_{zz} = \int (x^2 + y^2) dm \qquad I_{yz} = I_{zy} = \int (yz) dm$$

Momentum and inertia

- We then have the angular momentum

$$L = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix} \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} = I * \omega$$

Inertia

- Finally the inertia can be expressed as the matrix

$$I = \begin{bmatrix} \int (y^2 + z^2) dm & -\int (xy) dm & -\int (xz) dm \\ -\int (xy) dm & \int (z^2 + x^2) dm & -\int (yz) dm \\ -\int (xz) dm & -\int (yz) dm & \int (x^2 + y^2) dm \end{bmatrix}$$

- The diagonal elements are called the (principal) moment of inertia
- The off-diagonal elements are called products of inertia

Inertia

- Equivalently, we can define the inertia as

$$I = \int_M r^2 \, dm = \int_V \rho * r^2 \, dV$$

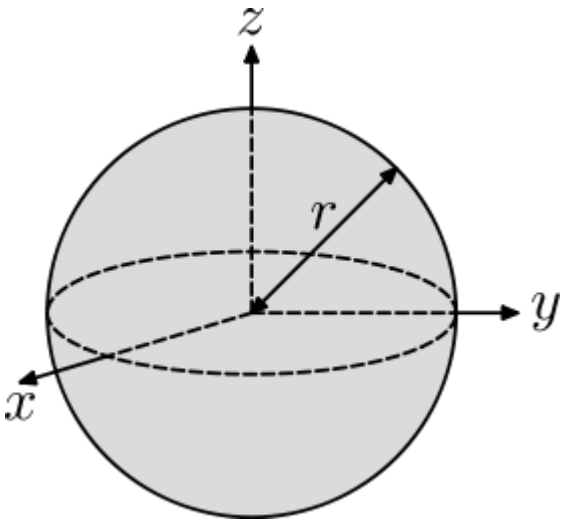
$$I = \int_V \rho(x,y,z) \begin{bmatrix} y^2+z^2 & -xy & -xz \\ -xy & z^2+x^2 & -yz \\ -xz & -yz & x^2+y^2 \end{bmatrix} dx \, dy \, dz$$

  - you can notice that the diagonal elements are the distances to the respective principal axis
  - and the non-diagonal elements the products of the perpendicular distances to the respective planes

Inertia of primitive shapes

- For primitive shapes, the inertia can be expressed with the parameters of the shape

- Illustration on a solid sphere
  - we can calculate the inertia of the sphere by integration of the moment of inertia of thin discs along one axis (*e.g.* z)
  - the surface of the sphere is defined by $x^2 + y^2 + z^2 = R^2$



Inertia of primitive shapes

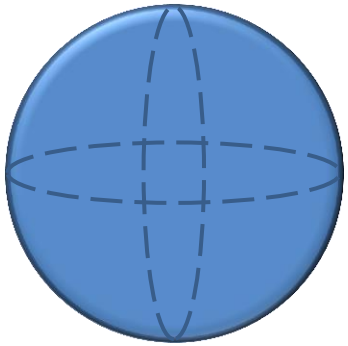
- Illustration on a solid sphere
  - the distance to the axis of rotation is the radius of the disc at the cross section along z: $r^2 = x^2 + y^2 = R^2 - z^2$
  - the inertia is given by the sum of moments of inertia of small cylinders of inertia $I = \frac{r^2 m}{2}$ along the z-axis:

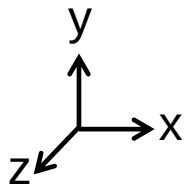
$$dI = \frac{1}{2} r^2 dm = \frac{1}{2} r^2 \rho dV = \frac{1}{2} r^2 \rho \pi r^2 dz$$

  - so we have $I = \frac{1}{2}\rho\pi \int_{-R}^{R} r^4 dz = \frac{1}{2}\rho\pi \int_{-R}^{R} (R^2 - z^2)^2 dz = \frac{1}{2}\rho\pi [R^4 z - 2R^2 z^3/3 + z^5/5]_{-R}^{R} = \rho\pi(1 - 2/3 + 1/5)R^5$
  - as $m = \rho(4/3)\pi R^3$, we finally have $I = \frac{2}{5} m R^2$

Inertia of primitive shapes

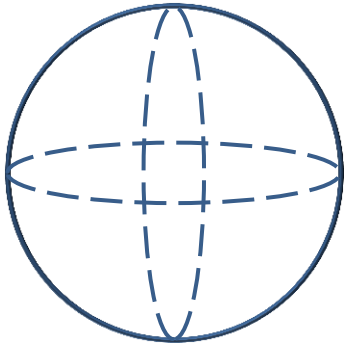
- Solid sphere, radius r and mass m





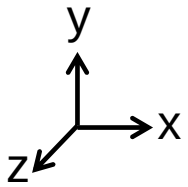
$$I = \begin{bmatrix} \frac{2}{5}mr^2 & 0 & 0 \\ 0 & \frac{2}{5}mr^2 & 0 \\ 0 & 0 & \frac{2}{5}mr^2 \end{bmatrix}$$

- Hollow sphere, radius r and mass m

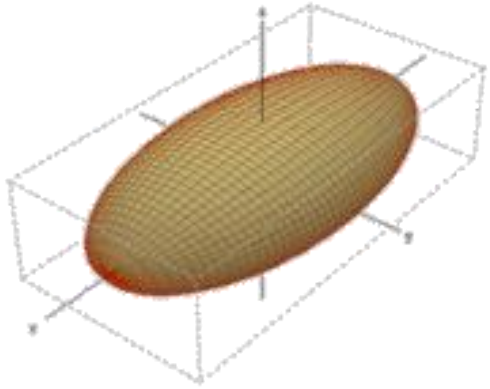


$$I = \begin{bmatrix} \frac{2}{3}mr^2 & 0 & 0 \\ 0 & \frac{2}{3}mr^2 & 0 \\ 0 & 0 & \frac{2}{3}mr^2 \end{bmatrix}$$

Inertia of primitive shapes

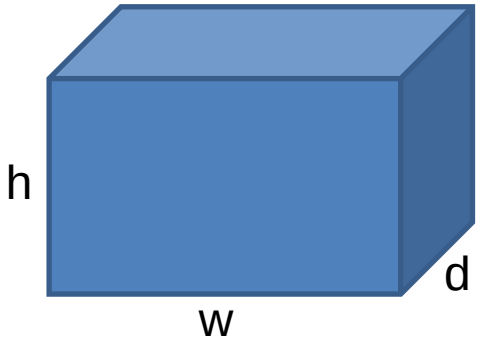


- Solid ellipsoid, semi-axes a, b, c and mass m



$$I = \begin{bmatrix} \frac{1}{5}m(b^2+c^2) & 0 & 0 \\ 0 & \frac{1}{5}m(a^2+c^2) & 0 \\ 0 & 0 & \frac{1}{5}m(a^2+b^2) \end{bmatrix}$$

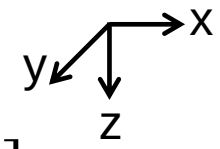
- Solid box, width w, height h, depth d and mass m

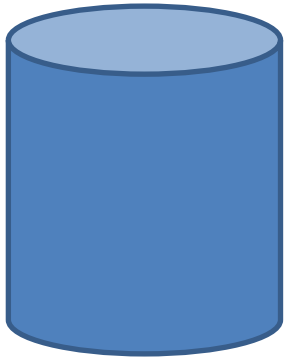


$$I = \begin{bmatrix} \frac{1}{12}m(h^2+d^2) & 0 & 0 \\ 0 & \frac{1}{12}m(w^2+d^2) & 0 \\ 0 & 0 & \frac{1}{12}m(w^2+h^2) \end{bmatrix}$$

Inertia of primitive shapes

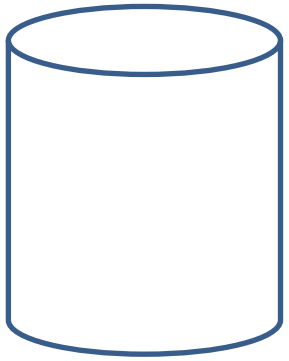
- Solid cylinder, radius r, height h and mass m





$$I = \begin{bmatrix} \frac{1}{12}m(3r^2+h^2) & 0 & 0 \\ 0 & \frac{1}{12}m(3r^2+h^2) & 0 \\ 0 & 0 & \frac{1}{2}mr^2 \end{bmatrix}$$

- Hollow cylinder, radius r, height h and mass m



$$I = \begin{bmatrix} \frac{1}{12}m(6r^2+h^2) & 0 & 0 \\ 0 & \frac{1}{12}m(6r^2+h^2) & 0 \\ 0 & 0 & mr^2 \end{bmatrix}$$

Parallel axis theorem

- But the object does not necessarily rotate around the center of mass
- To account for it, we need to modify the inertia matrix by applying the parallel axis theorem

$$I_v = I_{COM} + mr^2$$

where I_v is the inertia of the object about any axis v, I_{COM} the inertia about an axis through the COM, m is the mass of the object and r is the distance between the axes

Parallel axis theorem

- So the elements of our inertia matrix become

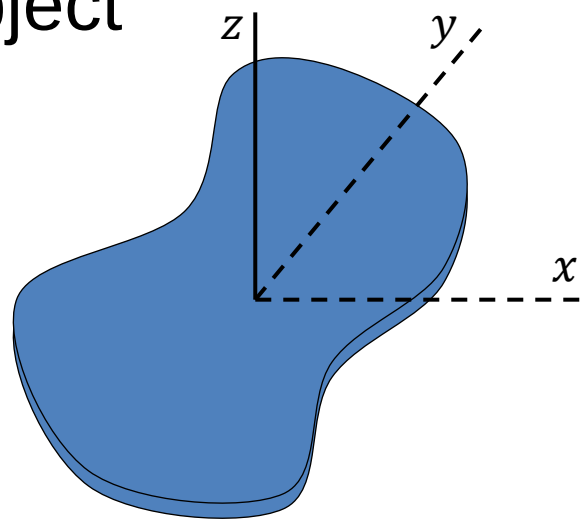
$$I_{xx} = \int (y^2 + z^2) dm + m{d_x}^2 \qquad I_{xy} = \int (xy) dm + m d_x d_y$$

$$I_{yy} = \int (z^2 + x^2) dm + m{d_y}^2 \qquad I_{xz} = \int (xz) dm + m d_x d_z$$

$$I_{zz} = \int (x^2 + y^2) dm + m{d_z}^2 \qquad I_{yz} = \int (yz) dm + m d_y d_z$$

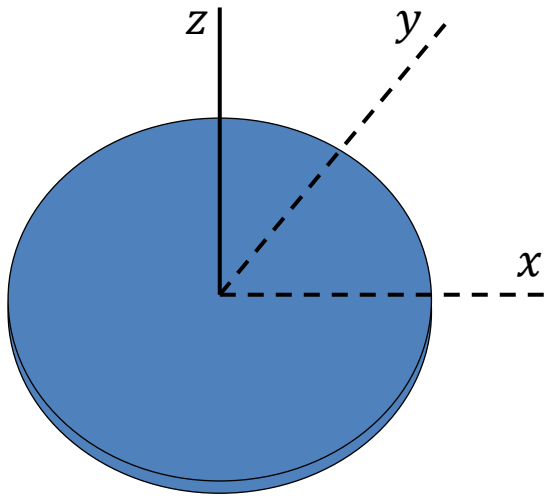
Perpendicular axis theorem

- For a planar 2D object, the moment of inertia about an axis perpendicular to the plane is the sum of the moments of inertia of two perpendicular axes through the same point in the plane of the object



$$I_z = I_x + I_y$$

for any planar object



$$I_z = 2I_x = 2I_y$$

for symmetrical objects

Reference frame

- Do not forget that the inertia tensor is constant in body space but varies in world space
- So at each simulation frame, the inertia tensor in world space is calculated by

$$I_{world}(t) = R(t) * I_{body}(t) * R(t)^T$$

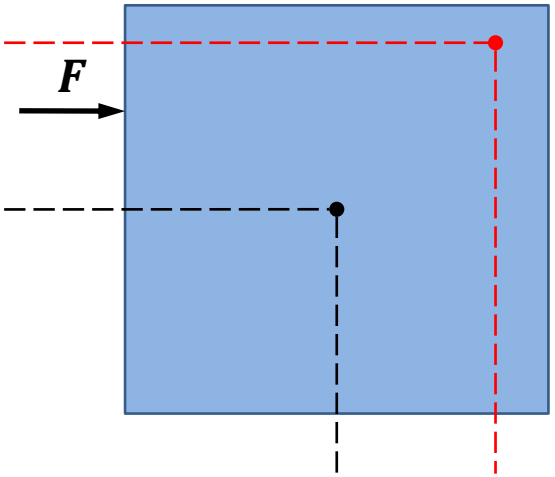
where R is the rotation matrix describing the orientation of the body in the world space

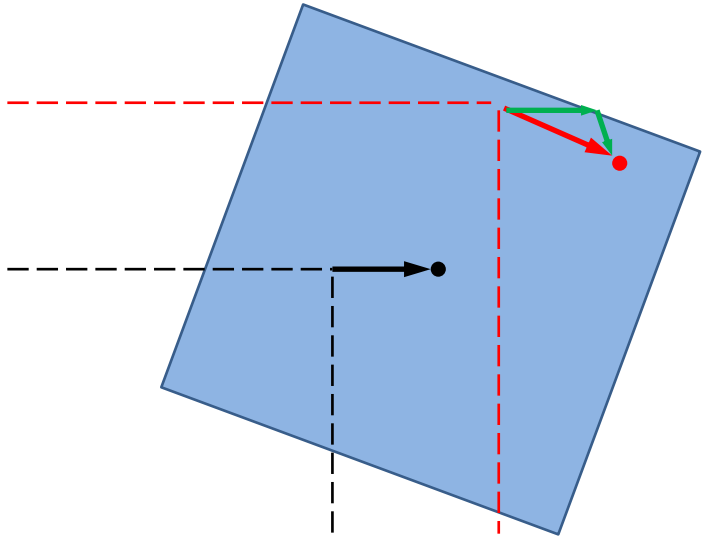
Complex objects

- When an object consists of multiple primitive shapes
  1. Calculate the individual inertia of each shape
  2. Use parallel axis theorem to transform to inertia about an axis through the COM of the object
  3. Add the inertia matrices together

Position on an object

- Remember that the object moves linearly as the COM moves
- Rotation add to the movement for points on the object
- Total motion of a point on the object is the sum of the two motions



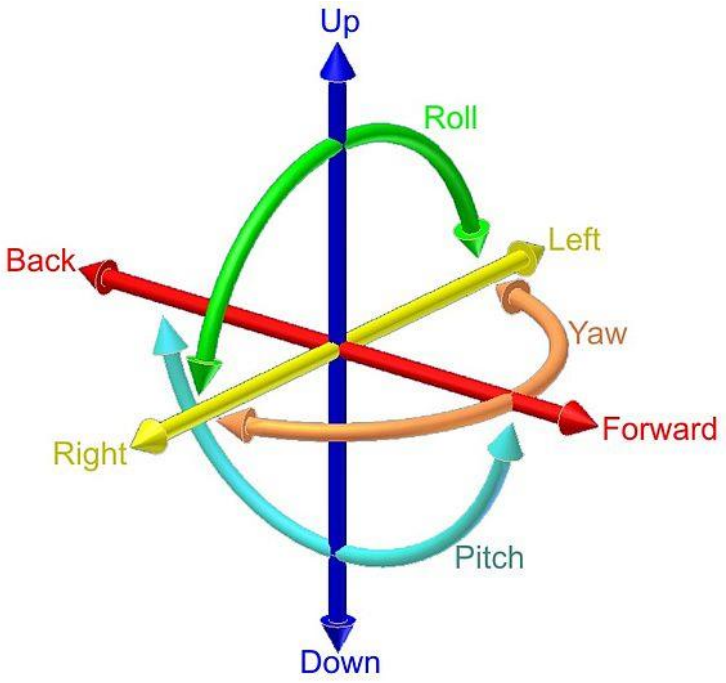


Motion constraint

- Sometimes a rigid body is not free to move around ‘on its own’, we want to constrain its movement
  - wheels on a chair
  - human body parts
  - trigger of a gun
  - opening door
  - actually almost anything you can think of in a game...

Degree of freedom

- To describe how a body can move in space you have to specify its degrees of freedom (DOF)
  - Translational
  - Rotational



Kinematic pair

- A kinematic pair is a connection between two bodies that imposes constraints on their relative movement
  - Lower pair, constraint on a point, line or plane
    - Revolute pair, or hinged joint: 1 rotational DOF
    - Prismatic joint, or slider: 1 translational DOF
    - Screw pair: 1 coordinated rotation/translation DOF
    - Cylindrical pair: 1 translational + 1 rotational DOF
    - Spherical pair, or ball-and-socket joint: 3 rotational DOF
    - Planar pair: 3 translational DOF
  - Higher pair, constraint on a curve or surface

Integrate constraints

- A good enough approximation for game applications is to assume that the energy of the components of the net force acting on constrained degrees of freedom is converted into heat and sound (E_o)
- Therefore you can project the net force on the unconstrained degrees of freedom and forget about the other ones
- Not true for soft bodies

Integrate constraints

- As solving a constraint for a body may influence the solving for another one, we need an iterative process
  - ```
    while (!done) {
          for all constraints c do solve c;
    }
    ```
  - Convergence is usually ensured by reaching either a maximal amount of iterations or a minimal change in every constraints
    ```

Sequential impulses

- Sequential impulses (SI) is a popular example of such iterative solver
  - It applies impulses at each constraint to correct velocity
  - SI is quite stable and converges to a global solution
- Why impulses?
  - Easier to deal with friction and collision
  - Work with velocity rather than acceleration
  - Given the time step, impulse and force are interchangeable
  - But velocity constraints not precise, might produce position drift, breaking the constraint

Sequential impulses

Step 1

Integrate applied forces, yielding to tentative velocities

Step 2

Apply impulses sequentially for all constraints to correct the velocity errors

Step 3

Use the new velocities to update the positions

Coordinate simulation

- Instead of adding impulses to enforce constraints, you can just not produce any force / torque along a specific DOF
- This is referred to as reduced coordinate simulation while impulse-based is a full coordinate simulation
- It allows for faster simulation (as less calculations are performed) and less tuning (notably of the magnitude of the impulses)
- But more vulnerable to numerical instability and might be difficult to parameterize the DOF system

Featherstone's Algorithm

- The Featherstone's Articulated Body Method (FABM) is an example of reduced coordinate constraint representation
- It computes the DOF accelerations given the articulated body current state and any external forces and torques
  - made of three loops over the articulated body, so $O(n)$
  - relies on the linear relation between the acceleration and the force

$$f + F_{ext} = I * a + p$$

where f is the test force, F_{ext} are the external forces (*e.g.* gravity), I the inertia of the articulated body, a the DOF acceleration (unknown) and p the bias force

Featherstone's Algorithm

Algorithm FABM $(q,\ \dot{q},\ F_{ext},\ \tau)$ // q is the state of the body (DOF values)

$v_0 \leftarrow 0$

for $i \leftarrow 1$ to n // compute velocities for all links

$j \leftarrow i - 1$ // j is the parent of link i

$v_i \leftarrow R_i^j v_j + s_i \dot{q}_i$ // R_i^j is the transformation from local system j to i, s_i is the joint axis

$p_i \leftarrow v_i \times I_i v_i - F_{ext,i}$

$c_i \leftarrow v_i \times s_i \dot{q}_i$

for $i \leftarrow n$ to 1 // compute inertia and bias force

$I_i^{accu} \leftarrow I_i$ // start accumulation of inertia

$p_i^{accu} \leftarrow p_i$ // start accumulation of bias force

for $j \leftarrow 1$ to numChildren(i) // for each child of i of index j

$$I_i^{accu} \leftarrow I_i^{accu} + R_i^j \left(I_j^{accu} - \frac{h_j h_j^T}{d_j} \right) R_j^i$$

$$p_i^{accu} \leftarrow I_i^{accu} + R_i^j \left(p_j^{accu} + I_j^{accu} c_j + \frac{u_j}{d_j} h_j \right)$$

$h_i \leftarrow I_i^{accu} s_i$

$d_i \leftarrow s_i^T h_i$

$u_i \leftarrow \tau_i - h_i^T c_i - s_i^T p_i^{accu}$

$a_0 \leftarrow 0$

for $i \leftarrow 1$ to n // compute acceleration for all joints

$j \leftarrow i - 1$ // j is the parent of link i

$\ddot{q}_i \leftarrow \frac{u_i - h_i^T R_i^j a_j}{d_i}$

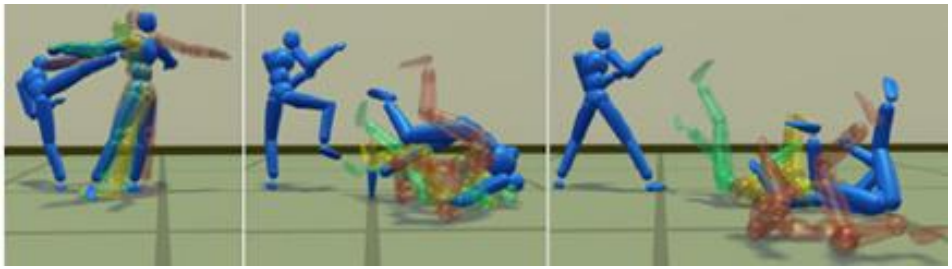
$a_i \leftarrow R_i^j a_j + c_i + s_i \ddot{q}_i$

Motion control

- Imagine you finally have rigid bodies moving and constrained correctly according to the forces you apply
- To let them ‘live’ on their own will be fine for ‘passive’ objects
  - projectile, furniture, environmental objects *etc*.
- But not for living beings as they will just fall onto the ground at the beginning of the game

Motion control

- In current games, animated bodies are controlled using a mix of kinematics and dynamics
- Kinematics to replay and slightly adapt pre-recorded motions
- Dynamics to passively animate objects reacting to external forces (*e.g.* human ragdoll)
- A real-time controller switches from one to the other according to events, forces, poses *etc.*



Motion control

- But imagine that you want to actively actuate rigid bodies using forces and torques
  - not yet in games but will probably in a near future
- You need actuators (to generate motion from within the bodies)
  - Joint torques
  - External forces
  - Virtual forces
  - Muscle forces
  - *contractile elements activated by the brain ... up to the source of motion! Where do we stop for a game?*

Joint torques

- Most straightforward actuation model
- Joint torques directly generate torques for each actuated DOF
- We assume that there is a ‘fake’ motor at the location of the joint that can produce torque
- For example if you want to increase a joint angle you will generate a positive torque and add it to the solving of the laws of motion
- Very useful to track pre-recorded motions or any other error-based poses (*e.g.* balance pose)
  - Amount of applied torque depends directly on the error

External forces

- We can try to control the motion by applying external forces on the right object at the right local position for the right amount of time
  - Difficult to produce realistic motions
  - Do not really fit the real world neither as motion originates from inside
- Similar to a puppetry technique
  - but need a master knowing how to control the system
- But very useful for the control of the global orientation and position of a complex system (root node)

Virtual forces

- Not really an actuation method
- Emulate the effect of applying an external force by computing the equivalent joint torque
- Use the relation between joint rotation and position where a force is applied (Jacobian of the system)

Muscle forces

- Motion actually comes from the contraction of muscles
  - they also produce torques at joints, but not in a ‘fake motor’ way
- By adding their position and physical behavior to the model, we can actuate rigid bodies with muscle activation
  - commonly used in biomechanics / motion analysis
  - the muscle model is usually a combination of two non-linear springs and one damper

Controller design

- Read the survey paper ‘Interactive Character Animation Using Simulated Physics: A State-of-the-Art Review’
- Joint-space motion control
  - defines and tracks kinematics target
- Stimulus-response network control
  - genetically evolves controllers according to objectives
- Constrained dynamics optimization control
  - finds optimal torques through online optimization

PD controller

- If your goal is to actuate a joint by feeding targets to track over time, then updating the necessary torque is quite easy
- You can use a Proportional Derivative (PD) controller
  - it can be used to compute joint torques linearly proportional to the difference between the current state and the target state
  - it will be based on joint orientation and angular velocity

PD controller

$$\tau = k_p(\theta_d - \theta) + k_v\left(\dot{\theta}_d - \dot{\theta}\right)$$

where τ is the generated joint torque, θ_d and θ the desired and current joint angle, $\dot{\theta}_d$ and $\dot{\theta}$ the desired and current joint angular velocity

- k_p and k_v are the controller gains, they regulate how responsive the controller is to deviations in orientation and angular velocity, respectively

PD controller

- A difficult and time-consuming task for the motion controller designer is to define the gain values
  - too high k_p will produce stiff unresponsive motions
  - too low k_p will not track correctly the target
  - too high k_v will converge too slowly to the target
  - too low k_v will produce oscillations
- Expect to spend time fine tuning gains

End of
Rigid body physics

Next

Numerical Integration