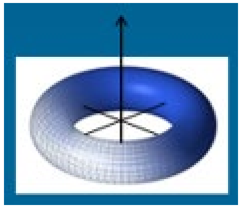
Rigidbody Dynamics

Overview

- Rigid bodies
  - Centre of mass
- Rigidbody dynamics
  - Rotational motion
  - Moment of inertia
  - Collision response

Rigid Bodies

- Large objects
  - ➢ Unlike particles (point mass)
    - Rigid bodies cover an extended area
    - Shape and size important for physics simulations
  - ➢ Need a location that represents the object's position
    - Object's 'origin'
    - Need not be inside or even on the object itself
    - Other coordinates of the object are relative to this origin
    - Theoretically, can choose any point on the object
      - – However for physically simulated objects, there's one point that dramatically simplifies the calculations



Rigid Bodies

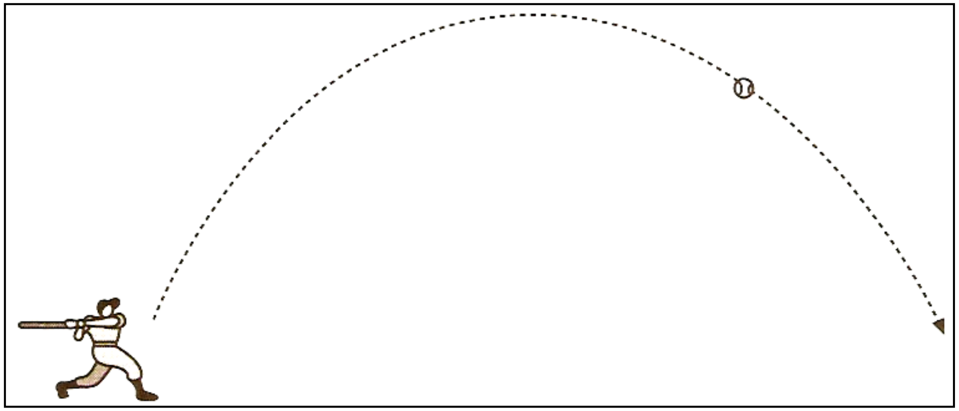
- Rigid bodies
  - ➢ System of particles
    - Remain at fixed distances from each other with no relative translation or rotation among them
    - Perfectly solid, does not change shape (deform)
      - – Has implications on the collision system
  - ➢ When considering rigid bodies
    - Dimensions and orientation important
    - Must consider both **linear** motion and **angular** motion
    - Displacement, velocity and acceleration still apply
      - – The difference is that the point tracked is the centre of mass

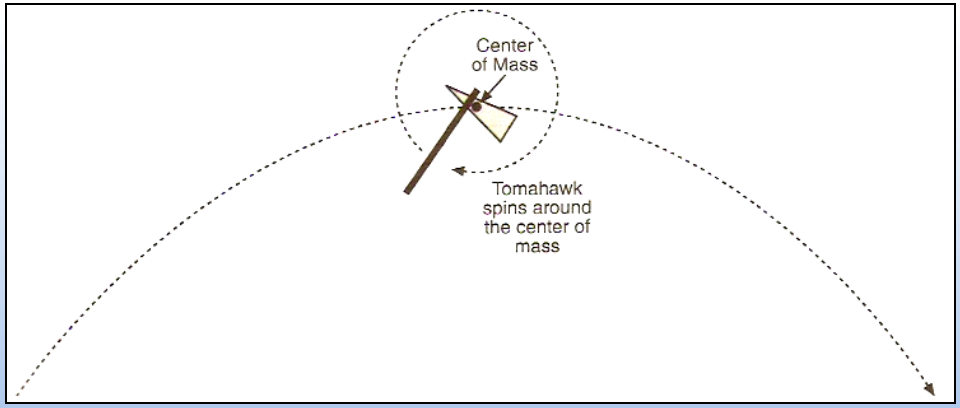
Rigid Bodies

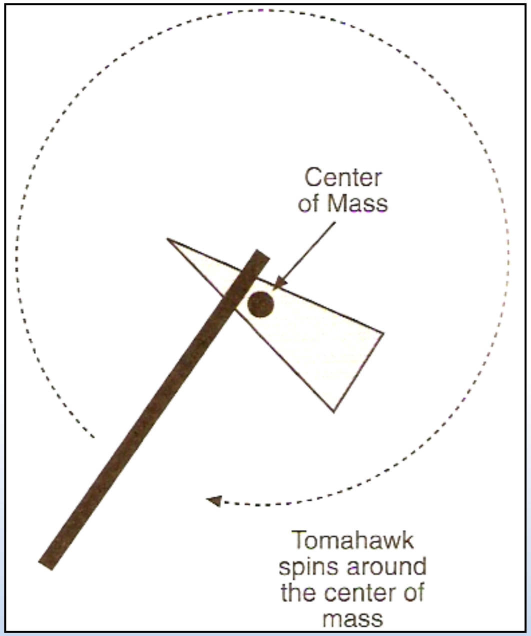
- Centre of mass
  - A specific point at which the system's mass behaves as if it were concentrated
    - Splitting the object in two through this point produces two objects with the same mass
    - Can balance the object at this point
  - Often also called the “centre of gravity”
  - Not always the geometric centre
  - Fixed position in relation to object (for rigid bodies)
    - Not necessarily in contact with the object

Rigid Bodies

- The centre of mass behaves like a particle
  - Same linear motion formulae as for particles
  - Allows separation of linear motion and angular motion calculations







Rigid Bodies

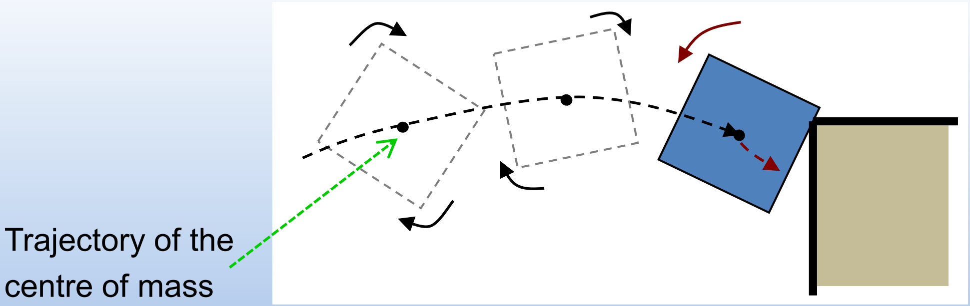
- Unity
  - Mass

    `Rigidbody.mass`
    - Guidelines: keep this below 10,000 kg and above 0.01 kg
      - Otherwise, may cause instability due to floating point errors
  - Centre of mass

    `Rigidbody.centerOfMass`
    - May not be the geometric centre of a mesh
      - Calculated automatically from all colliders attached to the Rigidbody
    - Public variable that can be changed using code

Rigidbody Dynamics

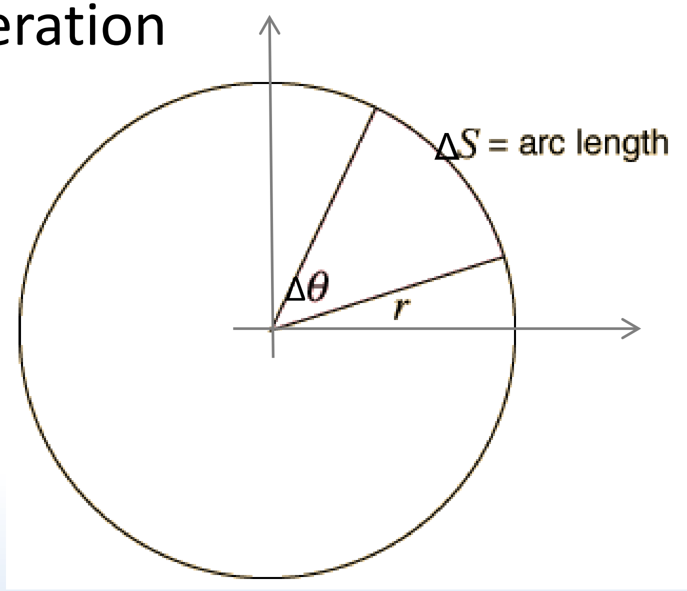
- Rigidbody dynamics
  - ➢ Combine linear motion and rotational (angular) motion
    - Given information, e.g., mass, shape, position and applied forces
      - How to calculate the trajectory, velocity and acceleration of every point that belongs to the rigid body?
    - Application of force
      - The same force acting on different points of a rigid body may result in different trajectories
      - Different points of a moving rigid body may have different trajectories



Rigidbody Dynamics

- 2D rotational motion (rotation in a plane)
  - Rotation is described in terms of angular displacement, angular velocity, and angular acceleration
  - Angular position $\theta = \frac{S}{r}$
  - Angular displacement

$$\Delta\theta = \theta_2 - \theta_1 = \Delta S / r$$



- By convention
  - Positive θ is counterclockwise
  - Angle θ is measured in radians

Rigidbody Dynamics

- Angular velocity
  - Rate of change of angular displacement (*rad/sec*)

$$\omega_{average} = \frac{\Delta\theta}{\Delta t} \qquad \omega = \lim_{\Delta t \to 0} \frac{\Delta\theta}{\Delta t} = \frac{d\theta}{dt}$$

- Angular acceleration
  - Rate of change of angular velocity (*rad/sec²*)

$$\alpha_{average} = \frac{\Delta\omega}{\Delta t} \qquad \alpha = \lim_{\Delta t \to 0} \frac{\Delta\omega}{\Delta t} = \frac{d\omega}{dt}$$

Rigidbody Dynamics

- Equations of motion
  - Only valid for constant angular acceleration

Comparison of linear motion and angular motion equations

$$\theta = \omega_{average} t \qquad \vec{s} = \vec{v}_{average} t$$

$$\omega = \omega_{initial} + \alpha t \qquad \vec{v} = \vec{v}_{initial} + \vec{a} t$$

$$\theta = \omega_{initial} t + \frac{1}{2} \alpha t^2 \qquad \vec{s} = \vec{v}_{initial} t + \frac{1}{2} \vec{a} t^2$$

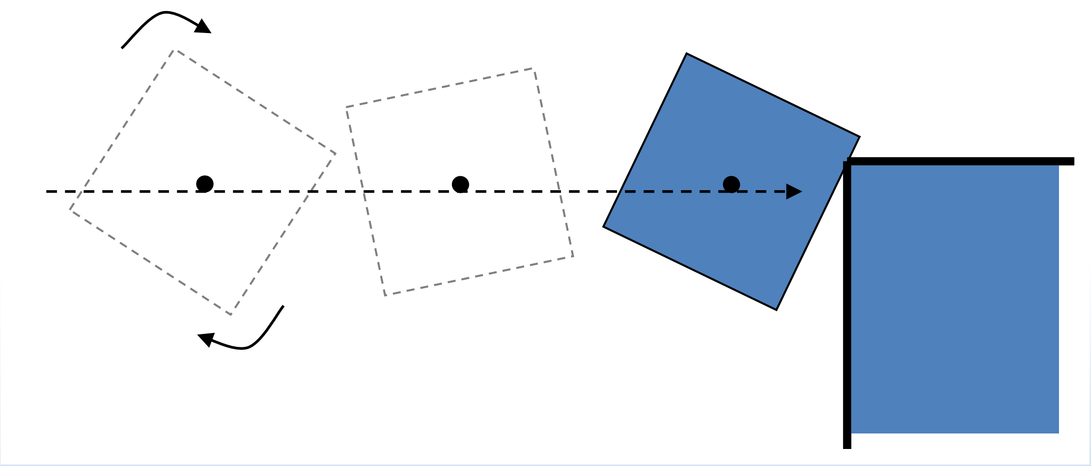
$$\omega^2 = \omega_{initial}^2 + 2\alpha\theta \qquad \vec{v}^2 = \vec{v}^2_{initial} + 2\vec{a}\vec{s}$$

Rigidbody Dynamics

- Unity
  - Angular velocity
    - `Rigidbody.angularVelocity`
      - Measured in radians per second
    - `Rigidbody.maxAngularVelocity`
      - Default max is 7

Rigidbody Dynamics

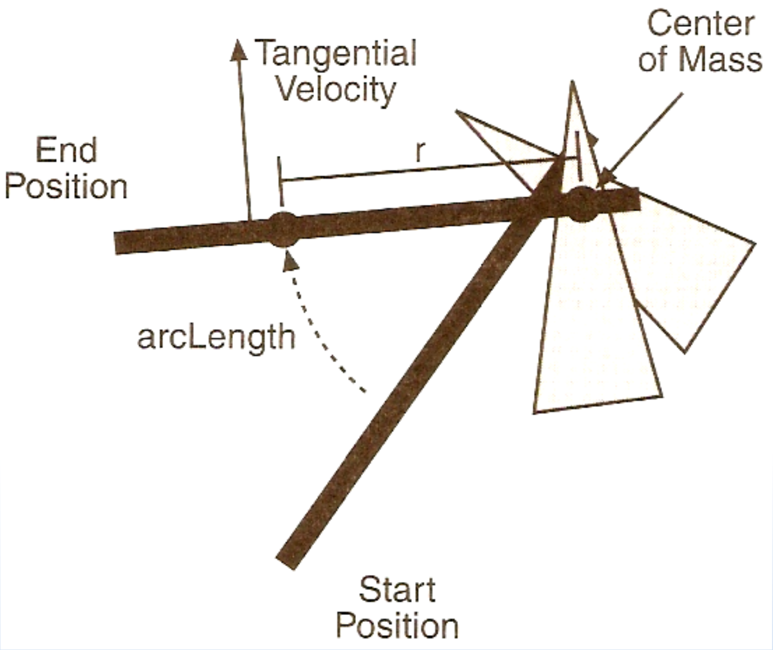
- Force
  - ➢ Need to be able to apply force at a distance away from the centre of mass
    - E.g. at a point on the surface of the rigid body



Rigidbody Dynamics

- 2D rotational motion
  - Equations of angular velocity and angular position

$$\omega = \frac{d\theta}{dt} \qquad \theta = \frac{arcLength}{r}$$



  - Combining gives
    - r is constant, doesn't change with respect to time

$$\omega = \left(\frac{1}{r}\right)\frac{darcLength}{dt}$$

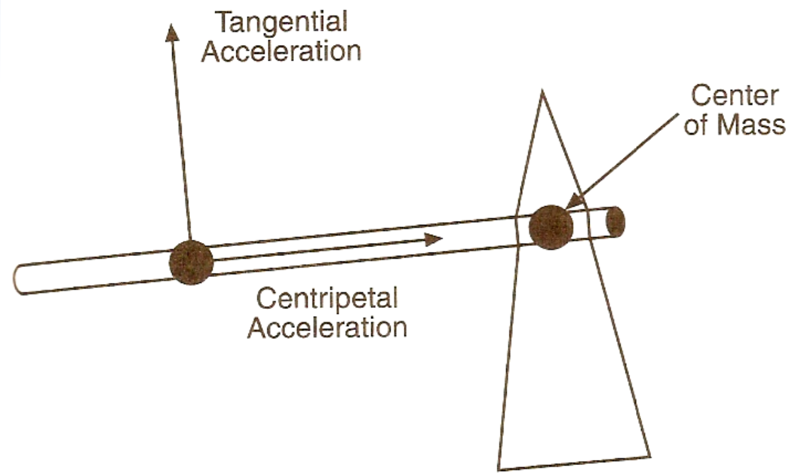
  - Tangential velocity

$$\boxed{v_t = \omega r}$$

Rigidbody Dynamics

- Tangential acceleration
  - Changes the magnitude of the velocity at the point

$$\frac{dv}{dt} = r\frac{d\omega}{dt} \qquad \boxed{a_t = \alpha r}$$

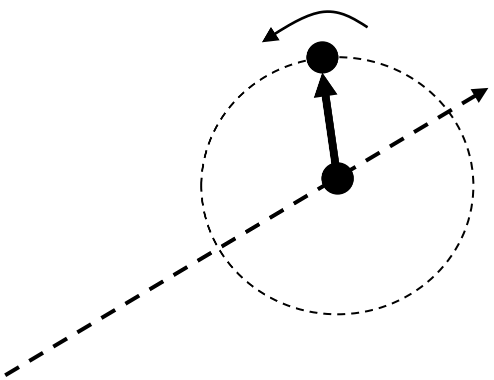


- Centripetal acceleration
  - Changes the direction of the tangential velocity, but not its magnitude
  - Corresponds to centripetal force which causes point to turn from its straight path

$$\boxed{a_c = \frac{{v_t}^2}{r} = \omega^2 r}$$

Rigidbody Dynamics

- 3D rotational motion
  - ➢ In world coordinates
    - Linear and angular velocity and acceleration applied to a point at a distance, *r*, from the centre of mass

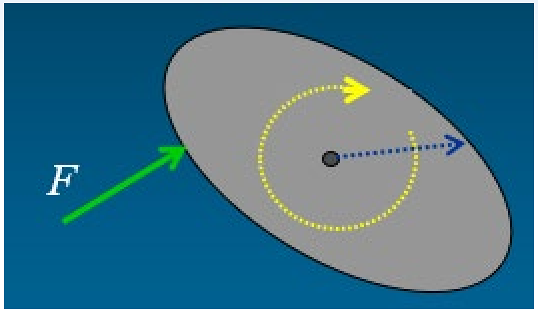


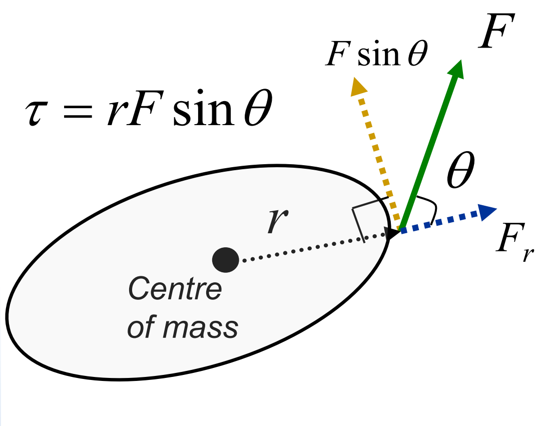
$$\vec{v}_{world} = \vec{v}_{CentreOfMass} + \vec{\omega} \times \vec{r}$$

$$\vec{a}_{world} = \vec{a}_{CentreOfMass} + \vec{\alpha} \times \vec{r} + \vec{\omega} \times \left(\vec{\omega} \times \vec{r} \right)$$

Rigidbody Dynamics

- Torque
  - Influence which tends to change the rotational motion of an object
  - Every force applied to an object not through the centre of mass will generate a rotational force known as torque
    - Force changes linear acceleration, torque changes angular acceleration
    - The turning or twisting action on an object about a rotation axis due to a force ('twist force')





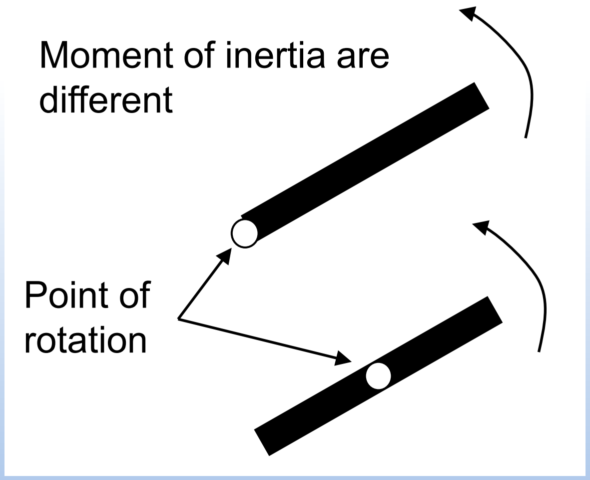
$$\boxed{\tau = rF\sin\theta}$$

Rigidbody Dynamics

- Moment of inertia
  - Similar role in rotational dynamics as mass in linear dynamics
  - Used to determine the relationship between
    - Angular momentum and angular velocity
    - Torque and angular acceleration
  - It depends on the mass distribution and the axis of rotation

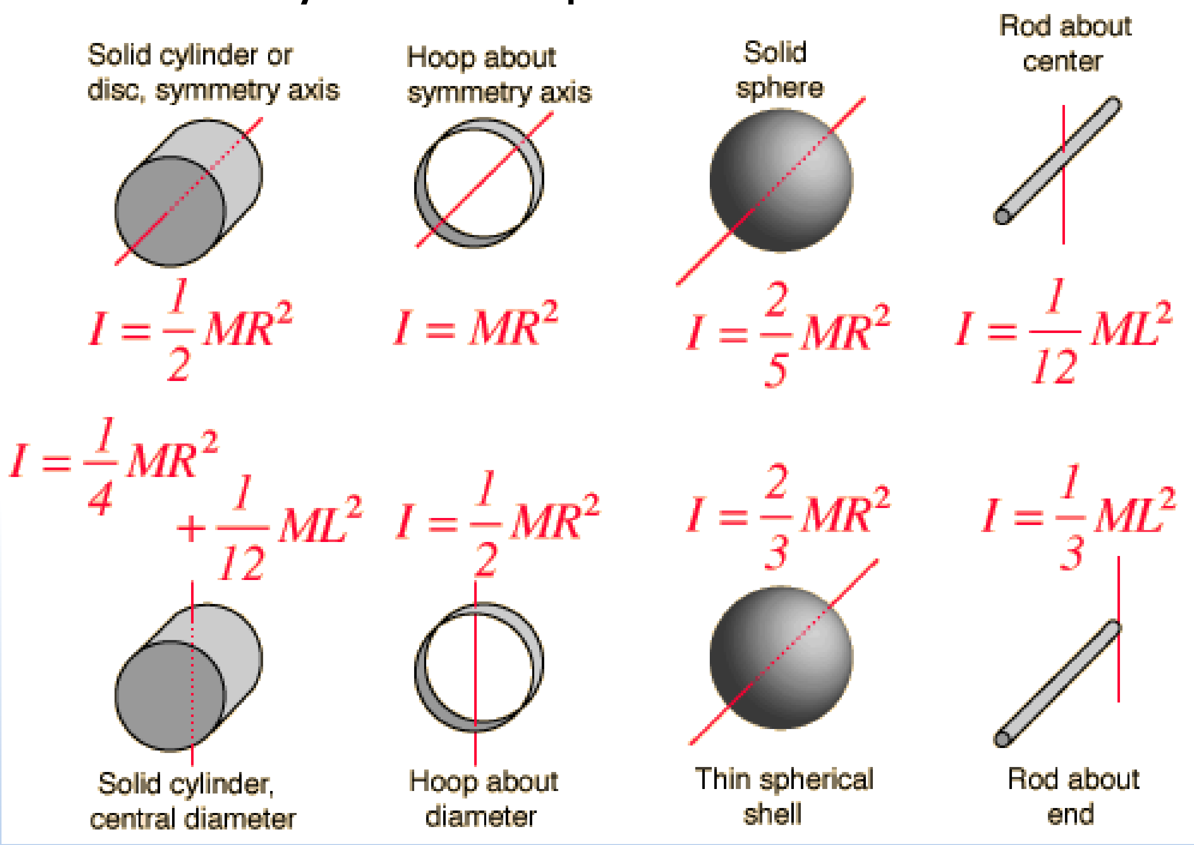
$$I = \sum_{i=1}^{n} m_i r_i^2$$

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



Rigidbody Dynamics

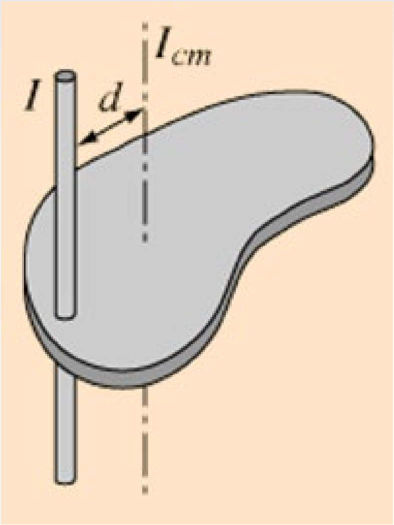
- Moment of inertia
  - For commonly used shapes



http://hyperphysics.phy-astr.gsu.edu/hbase/mi.html#mi

Rigidbody Dynamics

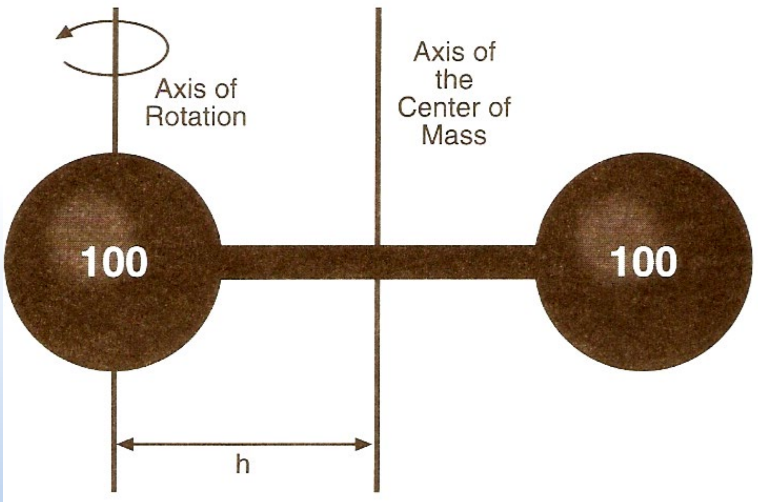
- Parallel-axis theorem
  - If the moment of inertia of an object about any axis that passes through its centre of mass is known
    - Can find its moment of inertia about any other parallel axis



$$I_{ParallelAxis} = I_{CentreOfMass} + Md^2$$

Where

- M is the object's mass
- d is the perpendicular distance between the two parallel axes



http://hyperphysics.phy-astr.gsu.edu/hbase/parax.html#pax

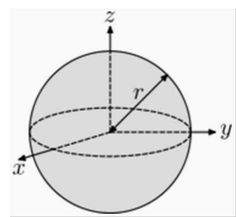
Rigidbody Dynamics

- Moment of inertia tensor
  - ➢ Torque

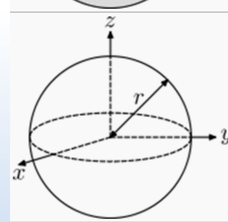
$$\vec{\tau} = I\vec{\alpha}$$

$$I = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{xy} & I_{yy} & -I_{yz} \\ -I_{xz} & -I_{yz} & I_{zz} \end{bmatrix}$$

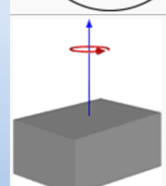
    - Where *I* is the 'moment of inertia tensor'
  - ➢ Common moment of inertia tensors for symmetric objects
    - Solid sphere
    - Hollow sphere
    - Solid cuboid



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



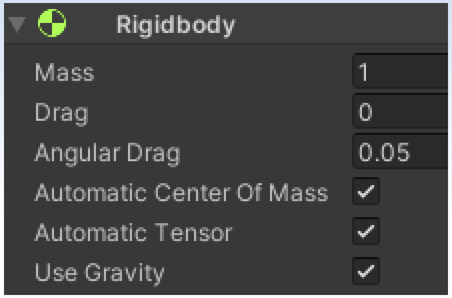
$$I = \begin{bmatrix} \frac{2}{3}mr^2 & 0 & 0 \\ 0 & \frac{2}{3}mr^2 & 0 \\ 0 & 0 & \frac{2}{3}mr^2 \end{bmatrix}$$



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

http://en.wikipedia.org/wiki/List_of_moment_of_inertia_tensors

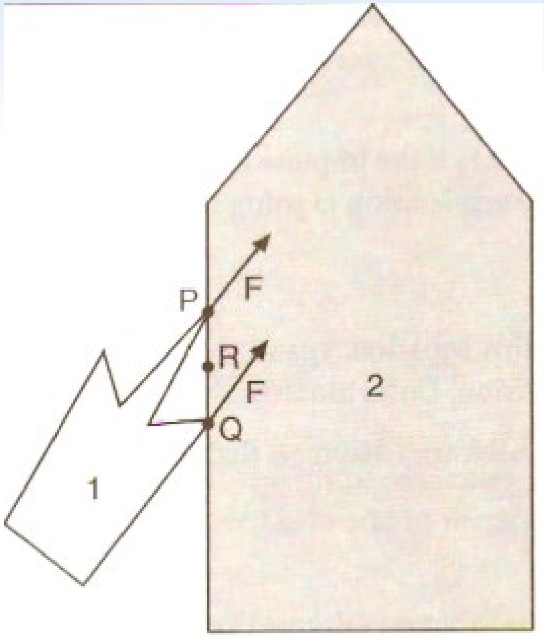
Rigidbody Dynamics

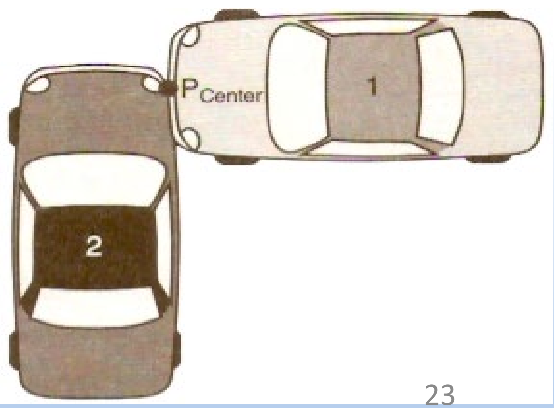


- Unity
  - Inertia tensor
    - Unity doesn't use a 3x3 inertia tensor matrix directly
    - Calculates diagonal elements for symmetric objects
      - Diagonal elements are stored in a `Vector3` variable `inertiaTensor`
      - Elements are calculated automatically by the physics engine from all Colliders attached to the Rigidbody
    - Unity uses another variable `inertiaTensorRotation` of type `Quaternion` to simulate asymmetric objects
  - Torque
    - `Rigidbody.AddTorque`
      - Direction is based on the left-hand rule

Rigidbody Dynamics

- Collision response
  - Must model both linear and angular dynamics
    - When dealing with angular forces, cannot treat objects as point masses
    - Rigid bodies can have multiple contact points
    - Impulses
      - Applied to each point of impact
      - On both objects are equal in magnitude but opposite in direction
    - Game engines usually simplify calculations by using a point that represents the centre of collision





Rigidbody Dynamics

- Linear and angular collision response

$$J = \frac{-v_r(e+1)}{\frac{1}{m_1} + \frac{1}{m_2} + \hat{\vec{n}} \bullet \left[\left(\frac{\left(r_1 \times \hat{\vec{n}}\right)}{I_1}\right) \times r_1\right] + \hat{\vec{n}} \bullet \left[\left(\frac{\left(r_2 \times \hat{\vec{n}}\right)}{I_2}\right) \times r_2\right]}$$

- Where
  - v_r is the relative velocities before impact $v_r = v_{1initial} - v_{2initial}$
  - ***n*** is the contact normal (unit vector along the line of action)
  - *e* is the coefficient of restitution

$$v_{1initial} = \left(\vec{v}_{1initialCentreOfMass} + \vec{\omega}_1 \times \vec{r}_1\right) \bullet \hat{\vec{n}}$$

$$v_{2initial} = \left(\vec{v}_{2initialCentreOfMass} + \vec{\omega}_2 \times \vec{r}_2\right) \bullet \hat{\vec{n}}$$

Rigidbody Dynamics

- Impulse momentum theorem
  - Change in momentum of each object in a collision is equal to the impulse that acts on that object

$$\vec{p}_f - \vec{p}_i = \Delta\vec{p} = \vec{J}$$

$$m\left(\vec{v}_f - \vec{v}_i\right) = \vec{J}$$

$$\boxed{\vec{v}_f = \vec{v}_i + \frac{\vec{J}}{m}}$$

$$\vec{L}_f - \vec{L}_i = \Delta\vec{L} = \vec{r} \times \vec{J}$$

$$I\left(\vec{\omega}_f - \vec{\omega}_i\right) = \vec{r} \times \vec{J}$$

$$\boxed{\vec{\omega}_f = \vec{\omega}_i + \frac{\left(\vec{r} \times \vec{J}\right)}{I}}$$

Rigidbody Dynamics

- Using the impulse, *J*, the change in linear and angular velocities of the objects can be calculated as follows

$$\vec{v}_{1\,final} = \vec{v}_{1initial} + \frac{\left(J\hat{\vec{n}}\right)}{m_1} \qquad \vec{v}_{2\,final} = \vec{v}_{2initial} + \frac{\left(-J\hat{\vec{n}}\right)}{m_2}$$

$$\vec{\omega}_{1\,final} = \vec{\omega}_{1initial} + \frac{\left(r_1 \times J\hat{\vec{n}}\right)}{I_1} \qquad \vec{\omega}_{2\,final} = \vec{\omega}_{2initial} + \frac{\left(r_2 \times -J\hat{\vec{n}}\right)}{I_2}$$

References

- Among others, material sourced from
  - https://docs.unity3d.com
  - Jason Gregory, Game Engine Architecture, A.K. Peters
  - Ian Millington, Game Physics Engine Development, Morgan Kaufmann
  - David Conger, Physics Modeling for Game Programmers, Thomson Learning
  - C.R. Nave, HyperPhysics

    http://hyperphysics.phy-astr.gsu.edu/hbase/hph.html
  - David Halliday, Robert Resnick and Jearl Walker, Fundamentals of Physics, Wiley