

Physically-based Modelling

Mass, Force, and Time

Young-Min Kang
Tongmyong University

Mass

- ❖ Mass
 - ❖ Resistance to being accelerated by force
 - ❖ accumulated density (kg / m³)
 - ❖ integration of density

$$m = \int \rho dV$$

Centre of Mass

❖ theory

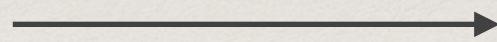
$$x_c = \frac{\int x_0 dm}{m}$$

$$y_c = \frac{\int y_0 dm}{m}$$

$$z_c = \frac{\int z_0 dm}{m}$$

$$c_g = \frac{\sum (c_{g_i} m_i)}{m}$$

❖ computation

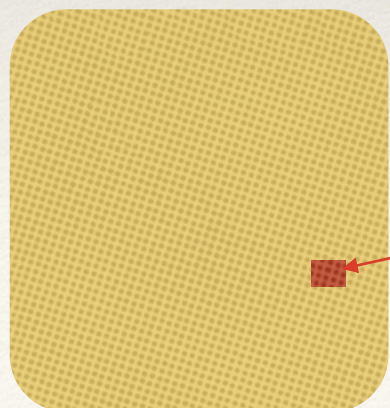


$$x_c = \frac{\sum x_i m_i}{\sum m_i}$$

$$y_c = \frac{\sum y_i m_i}{\sum m_i}$$

$$z_c = \frac{\sum z_i m_i}{\sum m_i}$$

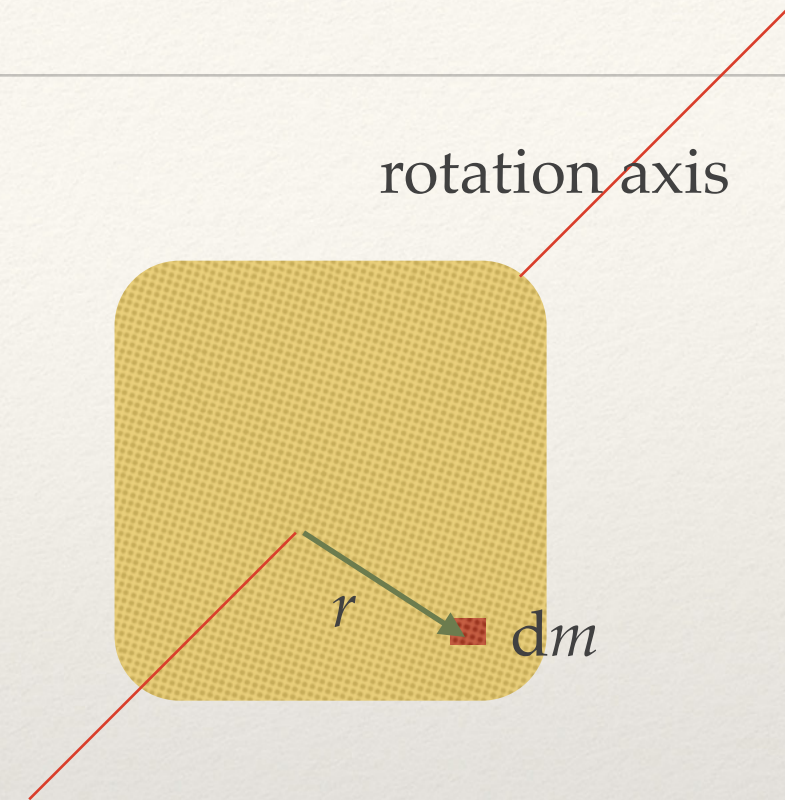
V



dm at (x_0, y_0, z_0)

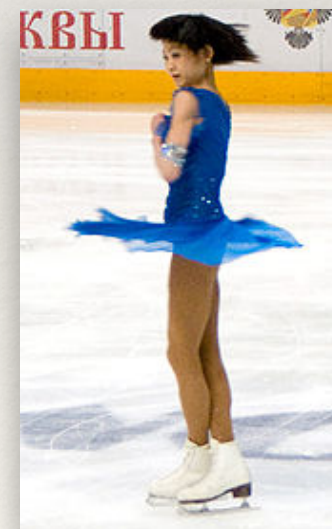
Moment of Inertia

- ❖ angular mass
- ❖ rotation axis is important
- ❖ resistance to angular acceleration along the axis

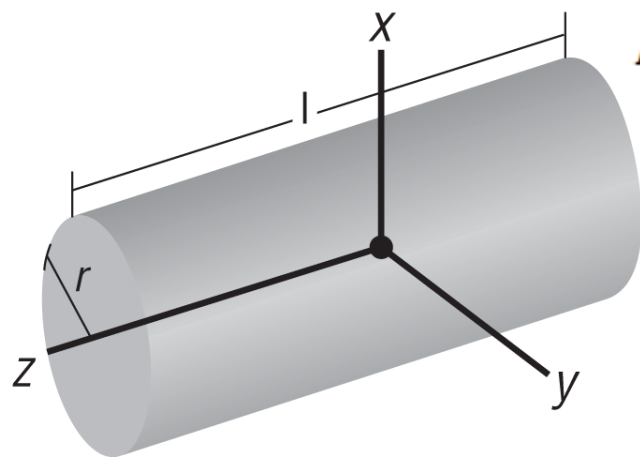


❖ 2nd

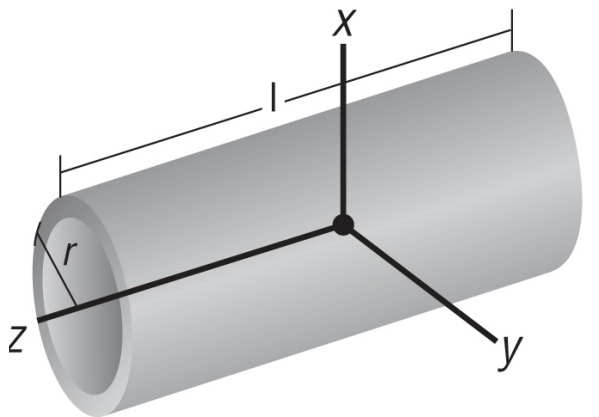
$$I = \int_m r^2 dm$$



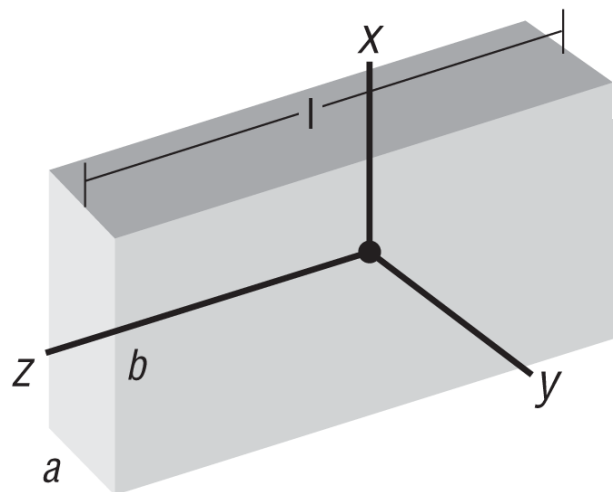
Examples



$$I_{xx} = I_{yy} = (1/4) mr^2 + (1/12) ml^2; I_{zz} = (1/2) mr^2$$

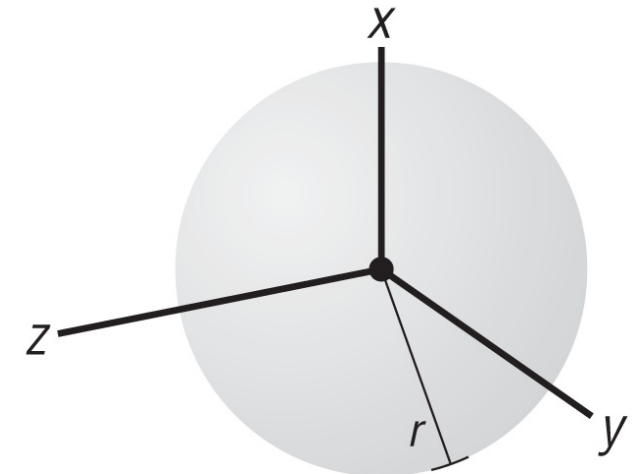


$$I_{xx} = I_{yy} = (1/4) mr^2 + (1/12) ml^2; I_{zz} = (1/2) mr^2$$



$$I_{xx} = I_{yy} = (1/4) mr^2 + (1/12) ml^2; I_{zz} = (1/2) mr^2$$

sphere $I_{xx} = I_{yy} = I_{zz} = (2/5) mr^2$



spherical shell $I_{xx} = I_{yy} = I_{zz} = (2/3) mr^2$

Newton's 2nd law of motion

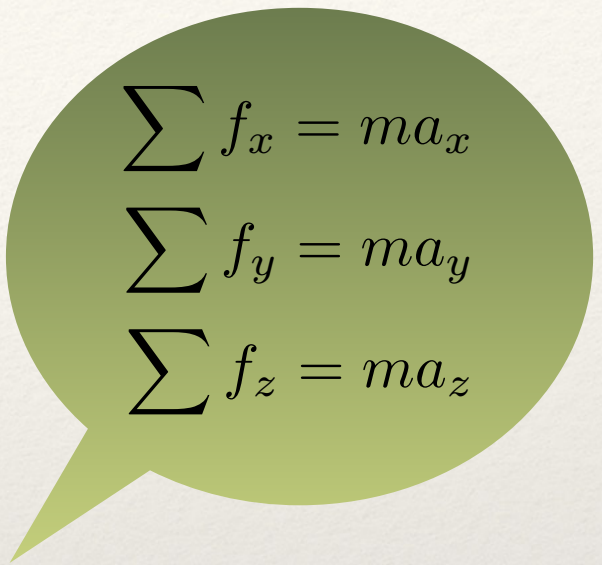
- ❖ force = mass x acceleration

$$\mathbf{f} = m\mathbf{a}$$

- ❖ total force \mathbf{f}_{total}

- ❖ sum of all the exerted forces

$$\mathbf{f}_{total} = \sum_{i=1}^n \mathbf{f}_i$$


$$\begin{aligned}\sum f_x &= ma_x \\ \sum f_y &= ma_y \\ \sum f_z &= ma_z\end{aligned}$$

- ❖ acceleration is determined by the total force

$$\mathbf{a} = \frac{\mathbf{f}_{total}}{m}$$

Linear Momentum and its Derivative

- ❖ Linear Momentum: \mathbf{G}

- ❖ mass \times velocity

- ❖ $\mathbf{G} = m\mathbf{v}$

- ❖ Derivative of Linear Momentum with respect to time

$$\frac{d\mathbf{G}}{dt} = \frac{dm\mathbf{v}}{dt} = m \frac{d\mathbf{v}}{dt} = m\mathbf{a}$$

- ❖ therefore,

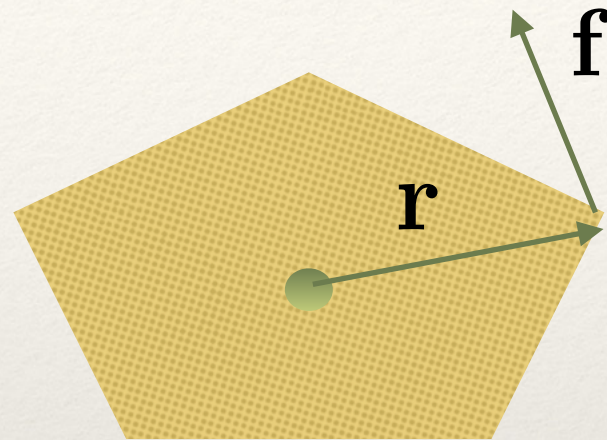
$$\frac{d\mathbf{G}}{dt} = \sum \mathbf{f}$$

Rotational Motion

❖ Torque: τ

❖ “force” in rotation

$$\tau = \mathbf{r} \times \mathbf{f}$$

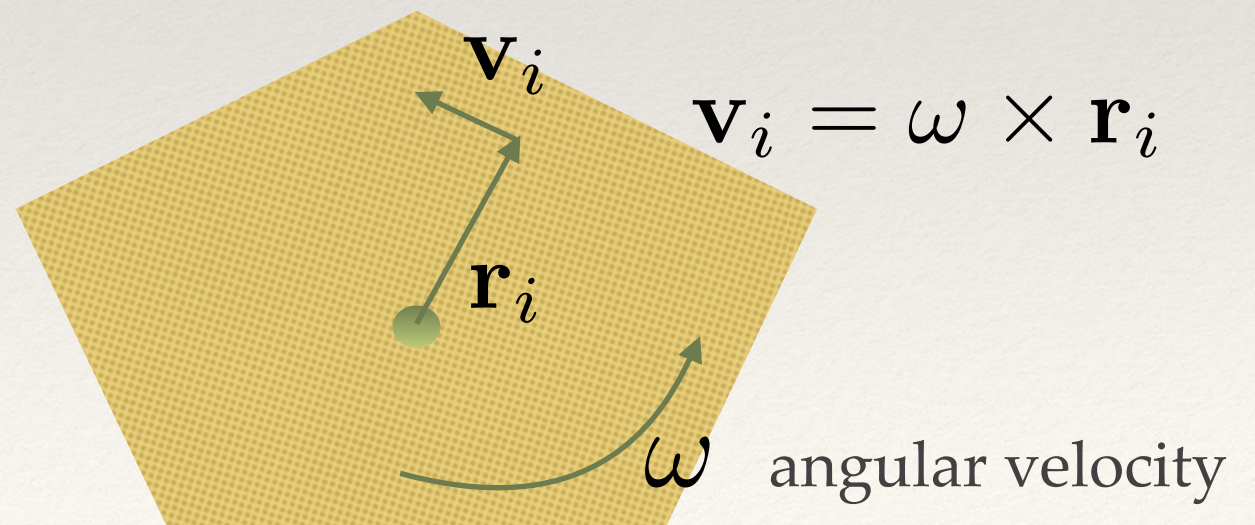


❖ angular momentum

❖ sum of moments of momentum of all particles

$$\mathbf{H} = \sum \mathbf{r}_i \times m_i \mathbf{v}_i$$

$$\mathbf{H} = \sum \mathbf{r}_i \times m_i (\boldsymbol{\omega} \times \mathbf{r}_i)$$



Angular Momentum = A. Mass x A. Velocity

❖ Linear Momentum: \mathbf{G}

❖ $\mathbf{G} = m\mathbf{v}$

❖ Angular Momentum

$$\mathbf{H} = \sum \mathbf{r}_i \times m_i (\boldsymbol{\omega} \times \mathbf{r}_i)$$



$$\mathbf{H} = \int \boldsymbol{\omega} \mathbf{r}^2 dm$$

$$= \boldsymbol{\omega} \int \mathbf{r}^2 dm$$

$$= \boldsymbol{\omega} \mathbf{I} = \mathbf{I} \boldsymbol{\omega}$$

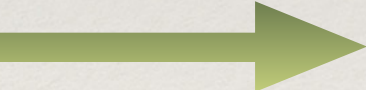
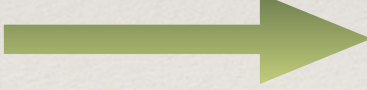
Inertia Tensor
(Angular Mass)

Angular Velocity

Derivative of Angular Momentum

- ❖ $d\mathbf{G} / dt = \text{total force}$
- ❖ $d\mathbf{H} / dt = \text{total torque}$

$$\frac{d\mathbf{H}}{dt} = \frac{d\mathbf{I}\omega}{dt} = \mathbf{I} \frac{d\omega}{dt} = \mathbf{I}\alpha$$


$$\sum \tau = \mathbf{I}\alpha$$

$$\alpha = \mathbf{I}^{-1} \sum \tau$$

Tensor

- ❖ Tensor

- ❖ mathematical expression that has magnitude and direction
- ❖ its magnitude may not be unique depending on the direction
- ❖ typically used to represent properties of materials where these properties have different magnitudes in different directions.

- ❖ Isotropic vs. Anisotropic

- ❖ isotropic: properties are the same in all direction
- ❖ anisotropic: properties vary depending on direction

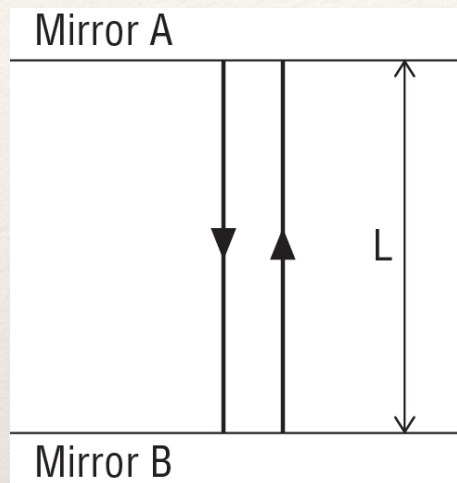
- ❖ Moment of Inertia

- ❖ inertia tensor (in 3D)
- ❖ nine components to fully describe it for any arbitrary rotation. (3x3 matrix)
- ❖ property of the body that varies with the axis of rotation.

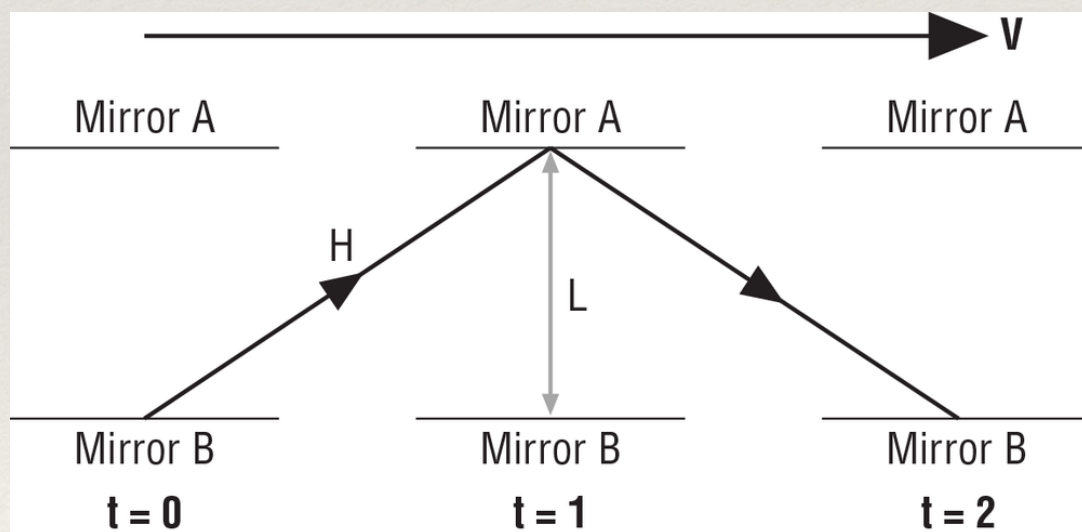
Time

- ❖ Classic Physics
 - ❖ Time is “constant”
- ❖ Modern Physics
 - ❖ Time is “variable”
 - ❖ “Speed of Light” is constant: c
 - ❖ $c = 299,792,458 \text{ m/s}$

Relativistic Time



$$c = 2L / \Delta t$$



mirrors are travelling in a spaceship

in the spaceship

$$c = 2L / \Delta t_s$$

on earth

$$c = 2H / \Delta t_e$$

if c is constant
time should be DIFFERENT

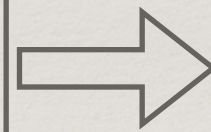
Time Dilation

- ❖ Simple geometry

$$H^2 = L^2 + \mathbf{v}^2 \Delta t_e^2 \quad \Rightarrow \quad \left(\frac{c \Delta t_e}{2} \right)^2 = L^2 + \left(\frac{\mathbf{v} \Delta t_e}{2} \right)^2$$

- ❖ time dilation

$$\begin{aligned} 4L^2 &= c^2 \Delta t_e^2 - \mathbf{v}^2 \Delta t_e^2 \\ 4L^2 &= (c^2 - \mathbf{v}^2) \Delta t_e^2 \\ \frac{4L^2}{c^2} &= \left(1 - \frac{\mathbf{v}^2}{c^2} \right) \Delta t_e^2 \end{aligned}$$

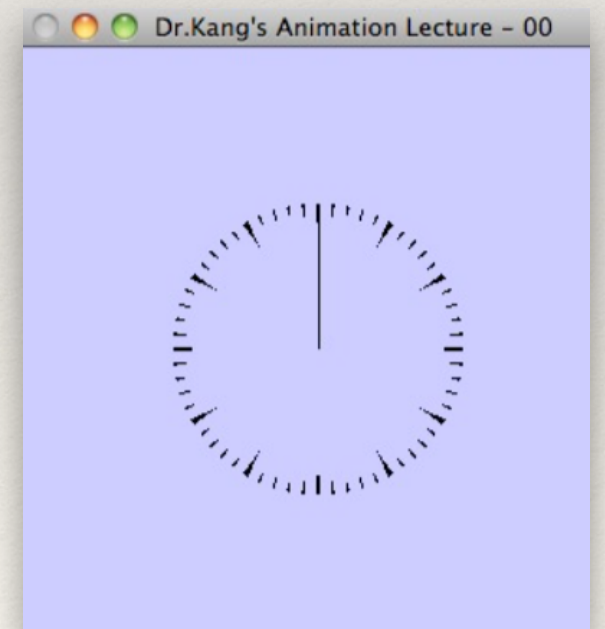


$$\begin{aligned} \frac{2L}{c} &= \sqrt{1 - \frac{\mathbf{v}^2}{c^2}} \Delta t_e \\ \Delta t &= \sqrt{1 - \frac{\mathbf{v}^2}{c^2}} \Delta t_e \end{aligned}$$

$$\Delta t_e = \frac{1}{\sqrt{1 - \frac{\mathbf{v}^2}{c^2}}} \Delta t$$

Time and Animation

- ❖ Animation
 - ❖ change over time
- ❖ Computer Animation
 - ❖ computing the physical state in accordance with “time”
- ❖ We must measure the time
 - ❖ Stop watch is needed
 - ❖ Let's make our own stop watch for animation



Stop Watch (header)

```
#ifndef _STOPWATCH_YMKANG_H
#define _STOPWATCH_YMKANG_H

#ifdef WIN32    // Windows system specific
#include <windows.h>
#else          // Unix based system specific
#include <sys/time.h>
#endif

class Stopwatch {
#ifdef WIN32
    LARGE_INTEGER frequency;           // ticks per second
    LARGE_INTEGER startCount;          //
    LARGE_INTEGER endCount;            //
#else
    timeval startCount;                 //
    timeval endCount;                   //
#endif
    double startTimeInMicroSec;
    double endTimeInMicroSec;
public:
    Stopwatch();
    void start();                       // start Stopwatch and record time to "startCount"
    void stop();                        // stop Stopwatch and record time to "endCount"
    double getElapsedTime();            // return the elapsed time at the last stop since the last start (microsec)
};

#endif
```


Stop Watch (implementation)

```
/*
 * Stopwatch.cpp
 * Young-Min Kang
 * Tongmyong University
 *
 */

#include "StopWatch.h"

StopWatch::StopWatch() {
#ifdef WIN32
    QueryPerformanceFrequency(&frequency);
    startCount.QuadPart = 0;
    endCount.QuadPart = 0;
#else
    startCount.tv_sec = startCount.tv_usec = 0;
    endCount.tv_sec = endCount.tv_usec = 0;
#endif
    startTimeInMicroSec = endTimeInMicroSec = 0.0;
}

void StopWatch::start() {
#ifdef WIN32
    QueryPerformanceCounter(&startCount);
#else
    gettimeofday(&startCount, NULL);
#endif
}

void StopWatch::stop() {
#ifdef WIN32
    QueryPerformanceCounter(&endCount);
#else
    gettimeofday(&endCount, NULL);
#endif
}

double StopWatch::getElapsedTime(){
#ifdef WIN32
    startTimeInMicroSec = startCount.QuadPart * (1000000.0 / frequency.QuadPart);
    endTimeInMicroSec = endCount.QuadPart * (1000000.0 / frequency.QuadPart);
#else
    startTimeInMicroSec = (startCount.tv_sec * 1000000.0) + startCount.tv_usec;
    endTimeInMicroSec = (endCount.tv_sec * 1000000.0) + endCount.tv_usec;
#endif
    return endTimeInMicroSec - startTimeInMicroSec;
}
```


Visualise Your Stop Watch

- ❖ Implement your stop watch

