물리기반 모델링

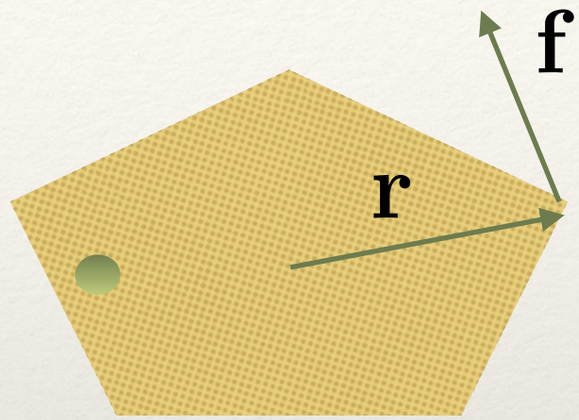
1.5 회전 운동

동명대학교
게임공학과

회전 운동

- 토크: τ
  - 회전 운동에서 힘의 역할

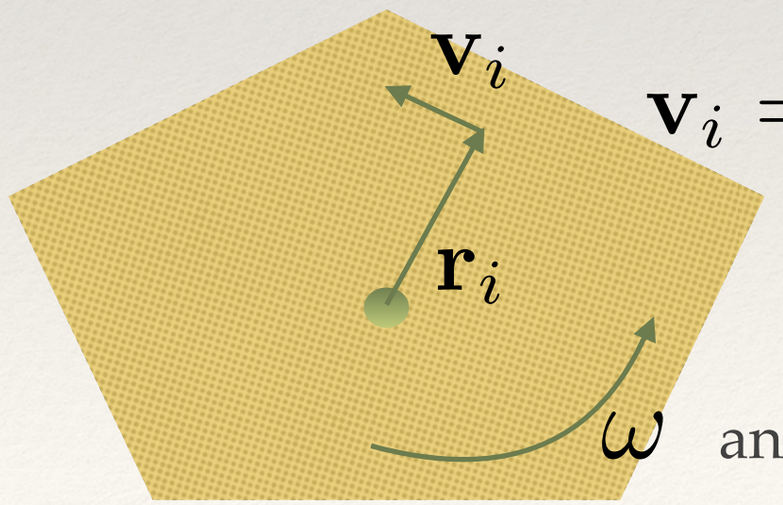
$$\tau = \mathbf{r} \times \mathbf{f}$$



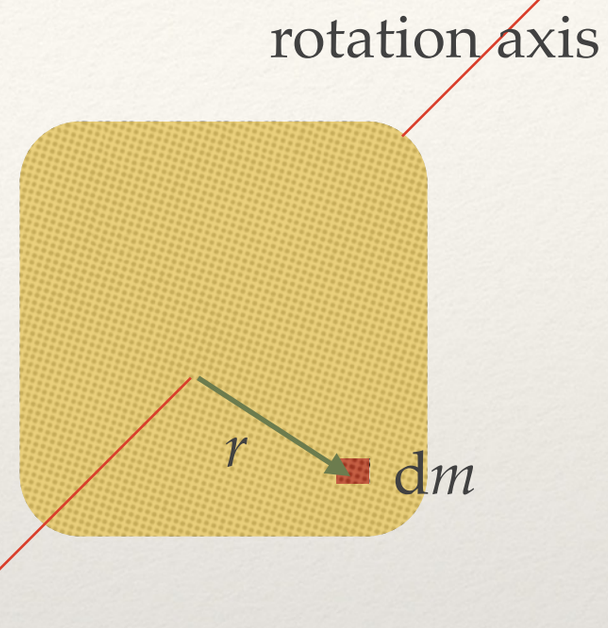
- 회전 운동량
  - 모든 입자의 운동량 모멘트의 합

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

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

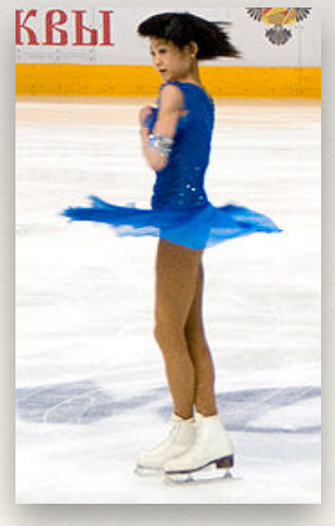


관성 모멘트 (회전질량)

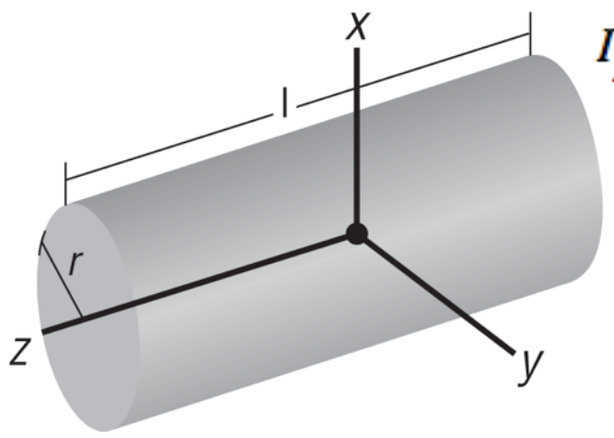


- 회전 질량
  - 회전축에 따라 달라짐
  - 축을 중심으로 하는 회전 가속에 저항하는 특성

$$I = \int_m r^2 \mathrm{d}m$$

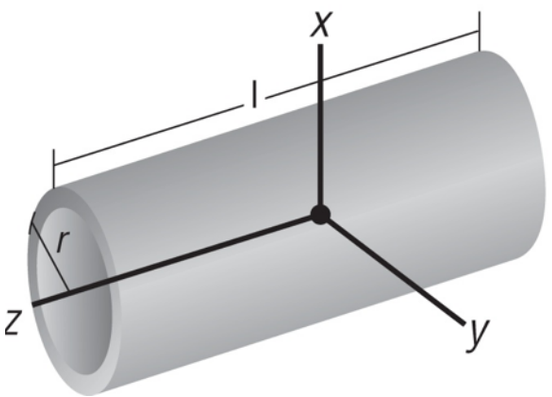


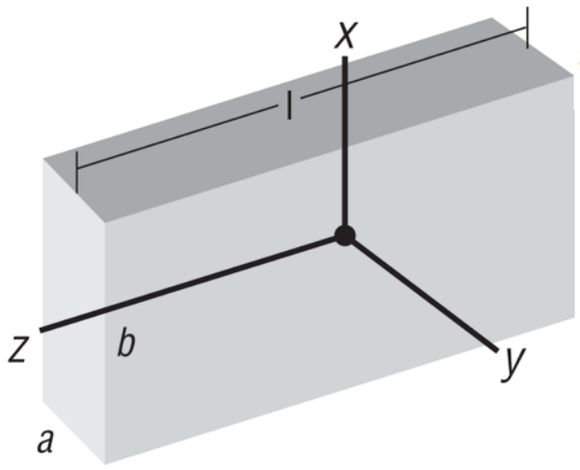
회전 질량의 예 (3개의 축에 대해)



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

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

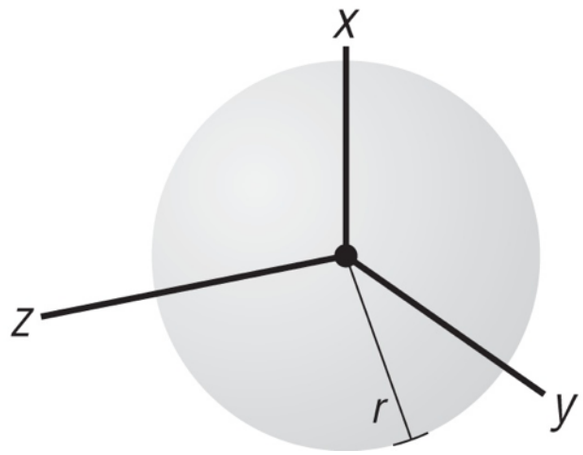




$$I_{xx} = I_{yy} = (1/4)\, mr^2 + (1/12)\, ml^2 ;\ I_{zz} = (1/2)\, mr^2$$

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

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



회전 운동량= 회전질량 x 회전속도

- 선운동량: **G**
  - $\mathbf{G} = m\mathbf{v}$
- 회전운동량

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

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

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

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

회전 질량
(관성 모멘트)

회전 속도(각속도)

회전 운동량의 미분

- $d\mathbf{G}/dt$ = 힘
- $d\mathbf{H}/dt$ = 토크(torque)

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

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

텐서(tensor)

- 텐서
  - 크기와 방향을 가진 수학적 표현
  - 방향에 따라 그 크기가 동일하지 않을 수 있음
  - 다른 방향에 대해 다른 크기를 갖는 물체의 특성을 표현할 때 사용
- 등방성(isotropic) 특성과 이방성(anisotropic) 특성
  - 등방성: 모든 방향으로 동일한 특성
  - 이방성:방향에 따라 달라지는 특성
- 관성 모멘트
  - 관성 텐서(3차원)
  - 아홉 개의 요소가 모든 방향으로의 특성을 표현할 수 있음
  - 회전 축에 따라 달라지는 강체의 특성을 표현

물리기반 모델링

1.6 강체 - 2차원

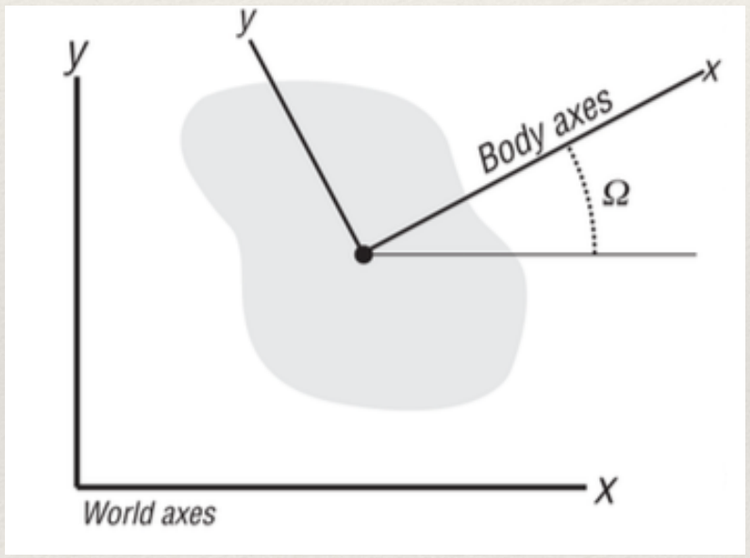
동명대학교
강영민

강체의 운동

- 입자와 강체의 차이
  - 입자: 회전이 없음
  - 강체: 회전
- 강체의 운동
  - 질량 중심의 선운동 (입자와 동일)
  - 회전 운동
    - 토크 τ (선운동에서 힘 f와 같은 작용)
      - 각 속도 ω (속도 **v**와 같은 역할)
      - 각 가속도 $\dot{\omega}$ (가속도 **a**와 같은 역할)

지역 좌표계

- 회전
  - 지역 좌표계의 원점을 중심으로 회전



- 2차원 강체의 회전
  - z 축 회전
- 회전은 원래의 상태에서 회전된 각을 표현하는 하나의 실수 Ω 로 표현 가능

각 속도와 각 가속도

- 선속도 = 시간에 대한 위치의 변화 비
- 각 속도 = 시간에 대한 회전 각의 변화 비
  - >> $\omega = \frac{d\Omega}{dt}$
- 각 가속도

$$\dot{\omega} = \frac{d\omega}{dt}$$

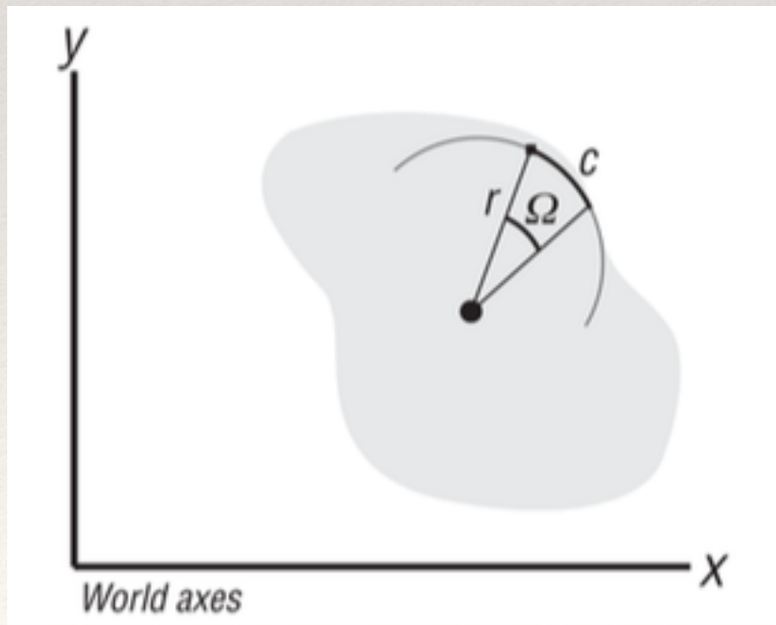
회전에 의한 선 속도

- 각도 Ω만큼의 회전
  - 지역 좌표 중심에서 r만큼 떨어진 위치는 원호 c를 따라 이동
- 간단한 관찰 $$c = r\Omega$$
- 미분하면.... $$dc/dt = r d\Omega/dt = r\omega$$

$$v = r\omega$$

- 가속
  - a = dv/dt

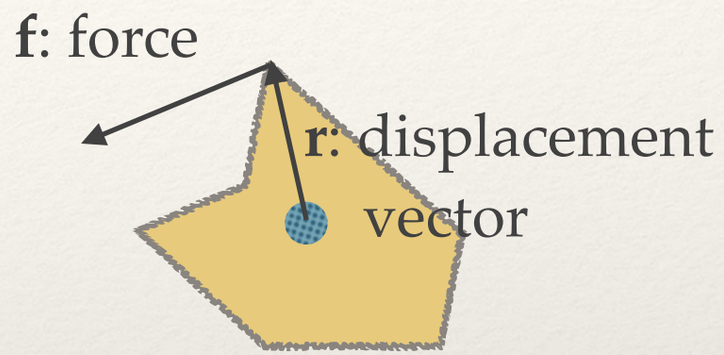
$$a = r\dot{\omega}$$



2차원 강체 시뮬레이션

- 상태 $(\mathbf{x}, \mathbf{v}, \Omega, \omega)$
- 관성
  - 질량 m : 선 운동에 대한 저항
  - 관정 모멘트 I : 회전 운동에 대한 저항
- 시뮬레이션
  - 힘과 토크를 계산 $\mathbf{f}, \tau$

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



2차원에서는....

$$\mathbf{f} = (f_x, f_y, 0)$$
$$\mathbf{r} = (r_x, r_y, 0)$$
$$\tau = (0, 0, \tau_z)$$

적분

- 선운동

$$\mathbf{v}(t+dt) = \mathbf{v}(t) + \frac{\mathbf{f}}{m}dt$$

$$\mathbf{x}(t+dt) = \mathbf{x}(t) + \mathbf{v}(t+dt)dt$$

- 회전운동

$$\omega(t+dt) = \omega(t) + I^{-1}\tau dt$$

$$\mathbf{\Omega}(t+dt) = \mathbf{\Omega}(t) + \omega(t+dt)dt$$

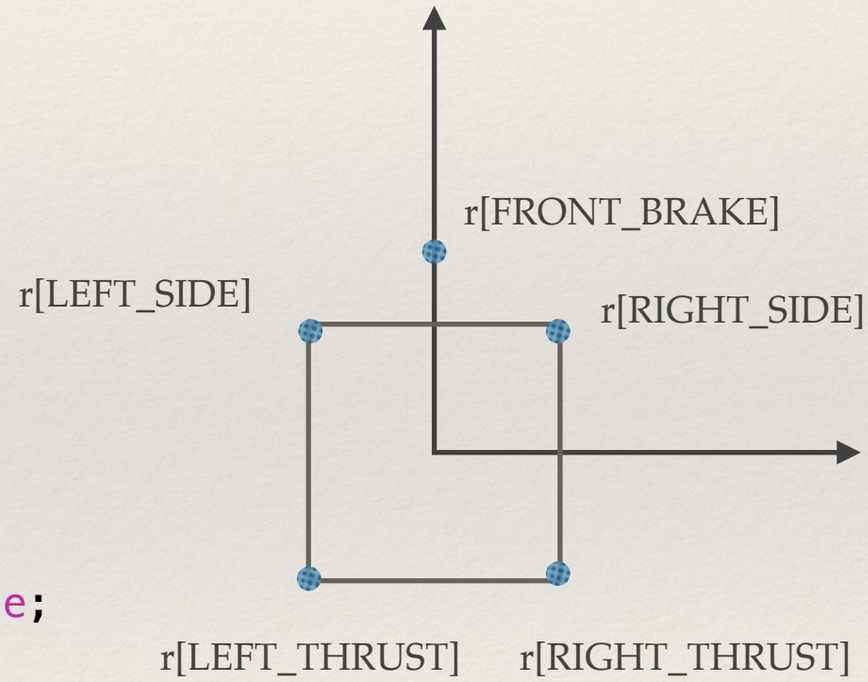
- 2D
  - I: 스칼라 … $I^{-1} = \frac{1}{I}$

Hovercraft.h

```
enum ENGINE_NUMBER {
    LEFT_THRUST,
    RIGHT_THRUST,
    RIGHT_SIDE,
    FRONT_BRAKE,
    LEFT_SIDE,
    NUMBER_OF_ENGINES
};

class CHovercraft {
    double mass;
    double inertia;
    CVec3d loc;
    CVec3d vel;
    CVec3d force;
    double angle;
    double aVel;
    double torque;

    CVec3d r[NUMBER_OF_ENGINES];
    CVec3d fLocal[NUMBER_OF_ENGINES];
    bool   on[NUMBER_OF_ENGINES];
    CVec3d localVectorToWorldVector(const CVec3d &lV);
public:
  …
    void draw(void);
    void switchEngine(int engineNumber, bool switch_state);
    bool isEngineOn(int engineNumber);
    void simulate(double dt);
    void setLocation(CVec3d location);
    CVec3d getLocation(void);
};
```

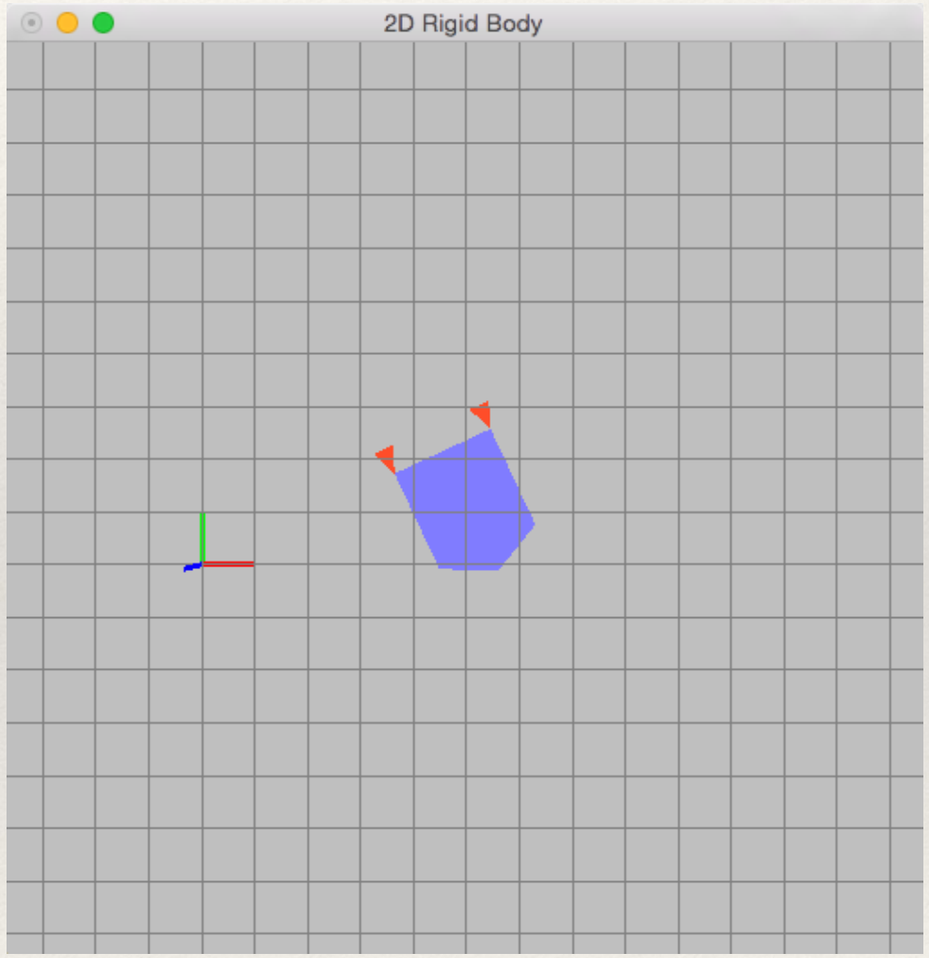
Hovercraft.cpp - 생성자

```
CHovercraft::CHovercraft() :
mass(1.0), inertia(1.0), angle(0.0), aVel(0.0), torque(0.0) {
    loc.set(0.0, 0.0, 0.0);
    vel.set(0.0, 0.0, 0.0);
    force.set(0.0, 0.0, 0.0);

    r[LEFT_THRUST].set(-1.0, -1.0, 0.0);
    r[RIGHT_THRUST].set(1.0, -1.0, 0.0);
    r[LEFT_SIDE].set(-1.0, 1.0, 0.0);
    r[RIGHT_SIDE].set(1.0, 1.0, 0.0);
    r[FRONT_BRAKE].set(0.0, 1.5, 0.0);

    fLocal[LEFT_THRUST].set( 0.0, 1.0, 0.0);
    fLocal[RIGHT_THRUST].set(0.0, 1.0, 0.0);
    fLocal[LEFT_SIDE].set(   1.0, 0.0, 0.0);
    fLocal[RIGHT_SIDE].set( -1.0, 0.0, 0.0);
    fLocal[FRONT_BRAKE].set( 0.0,-1.0, 0.0);

    for (int i=0; i<NUMBER_OF_ENGINES; i++) on[i] = false;
}

CHovercraft::~CHovercraft() {

}
```



Hovercraft.cpp - 시뮬레이션 부분

```
void CHovercraft::simulate(double dt) {
    force.set(0.0, 0.0, 0.0);
    torque = 0.0;

    // rigid body
    CVec3d fWorld;
    CVec3d torqueVec;
    for (int i=0; i<NUMBER_OF_ENGINES; i++) {
        if(on[i]) {
            fWorld = localVectorToWorldVector(fLocal[i]);
            force = force + fWorld;
            torqueVec = r[i]*fLocal[i];
            torque += torqueVec[2];
        }
    }
    // drag force
    double kd = 0.5;
    force  = force -kd*vel;
    torque += -kd*aVel;

    // numerical integration
    vel = vel + (dt/mass)*force;
    loc = loc + dt * vel;
    aVel = aVel + (dt/inertia)*torque;
    angle = angle + dt * aVel;
}
```



애니메이션 결과

- https://www.youtube.com/watch?v=xbu_-VP7Ed0
- http://goo.gl/s8TTAi