Physically-based Modelling

2D Rigid Body

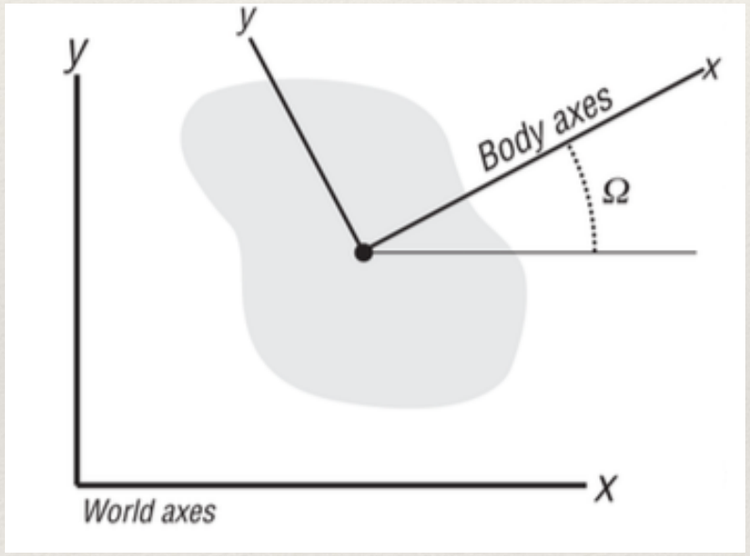
Young-Min Kang
Tongmyong University

Kinematics of Rigid Body

- difference between particle and rigid body
  - particle: no rotation
  - rigid body: rotates!
- rigid body
  - linear motion of mass centre (identical to particle)
  - rotational motion
    - torque τ (works like force **f**)
    - angular velocity ω (works like linear velocity **v**)
    - angular acceleration $\dot{\omega}$ (works like linear acceleration **a**)

Local Coordinates

- Rotation
  - about the origin of local coordinate system



- Rotation of 2d rigid body
  - about z-axis
  - rotation can be described as a single real number (angle) Ω

Angular Velocity and Acceleration

- linear velocity = rate of change of position in time
- angular velocity = rate of change of angle in time
  - so… $\omega = \frac{d\Omega}{dt}$
- angular acceleration

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

Linear Velocity due to Rotation

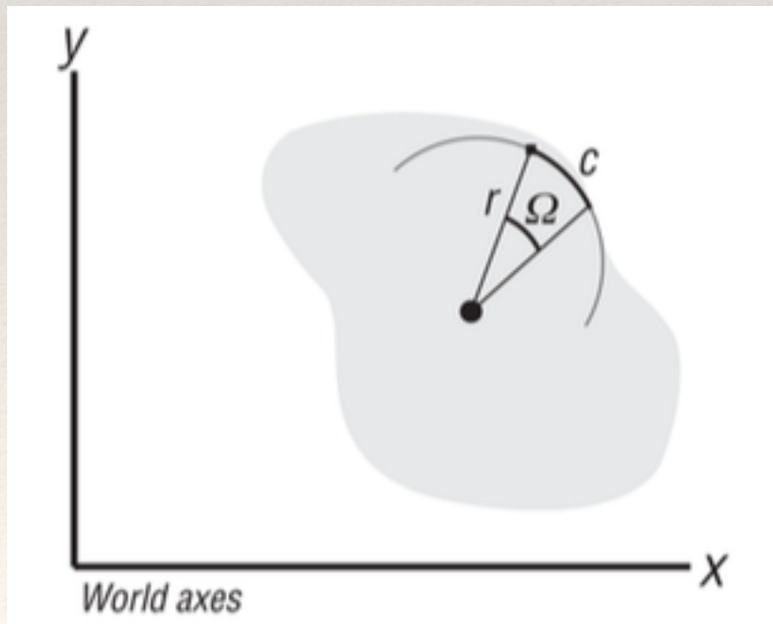
- Rotation with angle Ω
  - position at r from local origin move along the arc c
- Simple observation $c = r\Omega$

- Differentiate them $dc/dt = r d\Omega/dt = r\omega$

$$v = r\omega$$

- Acceleration
  - a = dv / dt

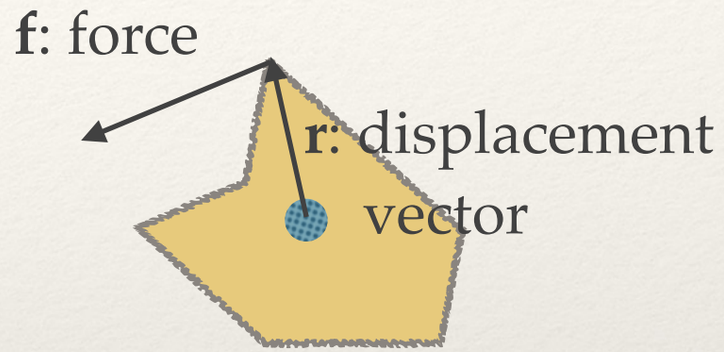
$$a = r\dot{\omega}$$



2D Rigid Body Simulation

- State $(\mathbf{x}, \mathbf{v}, \Omega, \omega)$
- Inertia
  - mass m : resistance to linear motion
  - moment of inertia I : resistance to rotational motion
- Simulation
  - compute force and torque $\mathbf{f}, \tau$

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



2D

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

Integration

- linear motion

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

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

- angular motion

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

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

- 2D
  - I: scalar ... $I^{-1} = \frac{1}{I}$

Implementation

- Dynamic Simulator

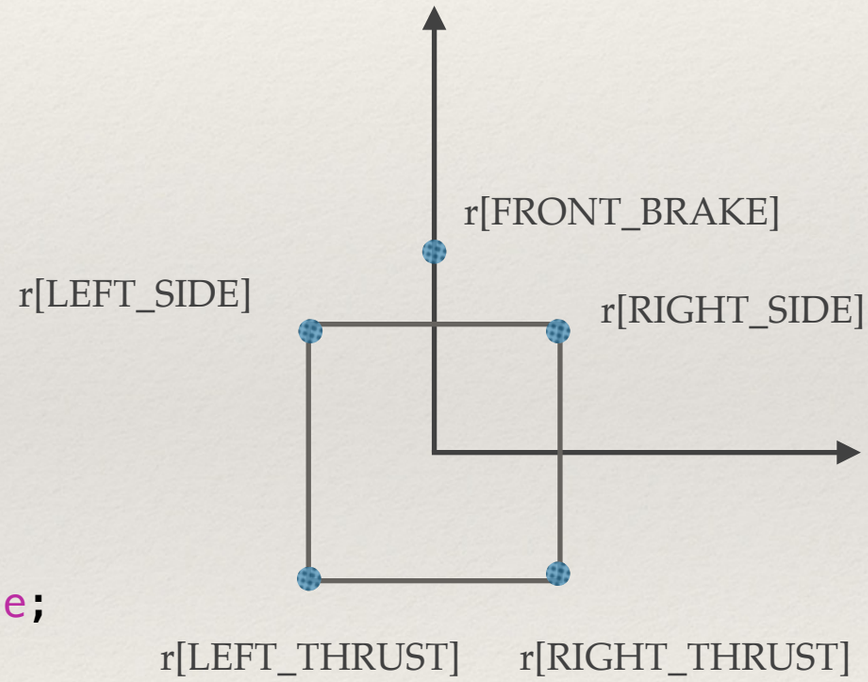
```
void CDynamicSimulator::doSimulation(double dt, double currentTime) {

    hover.simulate(dt);
}

void CDynamicSimulator::visualize(void) {

    hover.draw();

}

void CDynamicSimulator::control(unsigned char key) {
    int engineNumber = (int) (key-'1');
    hover.switchEngine(engineNumber, hover.isEngineOn(engineNumber)?false:true);
}

CVec3d CDynamicSimulator::getCameraPosition(void) {
    CVec3d loc;
    loc = hover.getLocation();
    return loc;
}
```

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);
};
```

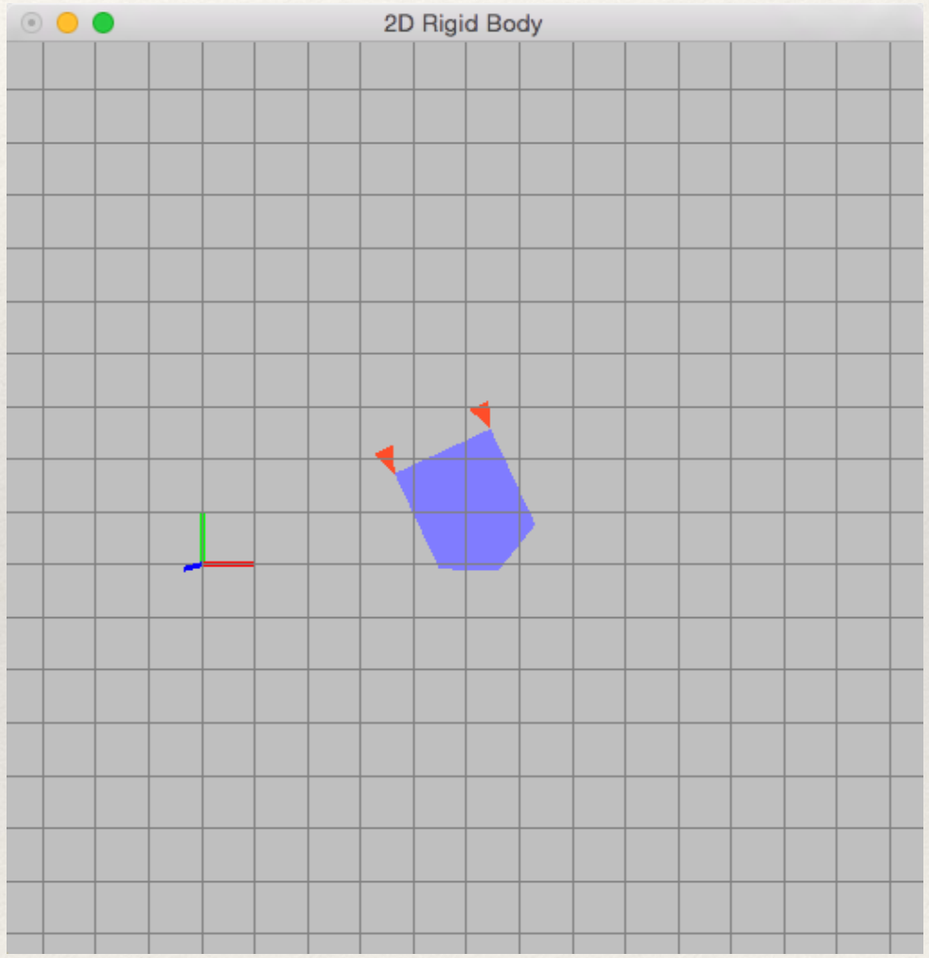
Hovercraft.cpp - constructor

```
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 - Simulation

```
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;
}
```



Animation Demo

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