

Physically-based Modelling

Kinematics

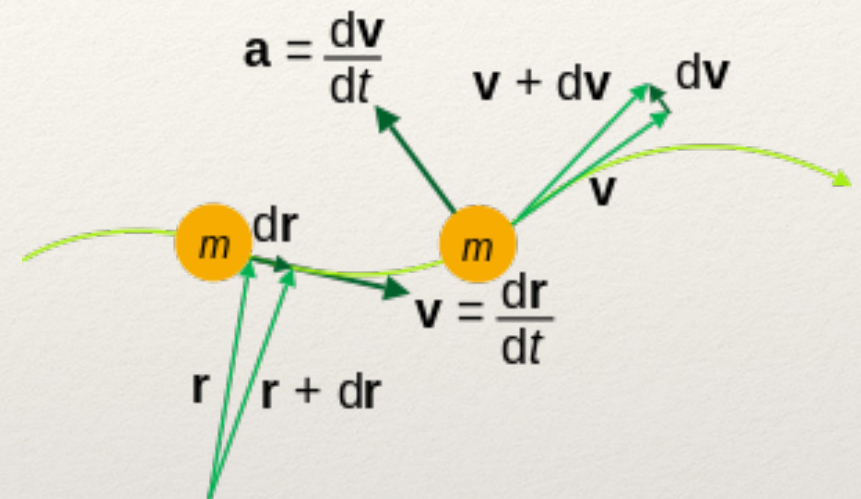
Young-Min Kang
Tongmyong University

Kinematics

- ❖ Kinematics vs. Dynamics

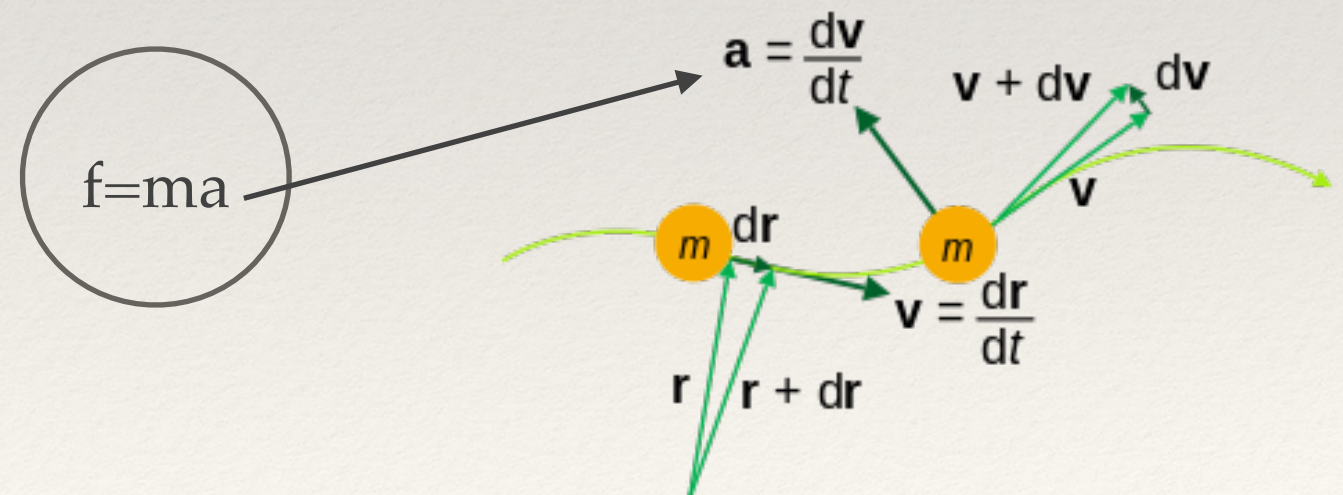
- ❖ kinematics

- ❖ force is not considered
 - ❖ location, velocity, and acceleration are considered as
 - ❖ functions of time
 - ❖ $x(t)$, $v(t)$, $a(t)$



- ❖ dynamics

- ❖ force plays the most essential role
 - ❖ compute $f(t)$
 - ❖ $a(t) = f(t)/m$
 - ❖ $v(t+dt) += a(t)dt$
 - ❖ $x(t+dt) += v(t+dt)dt$



Statics, Kinetics, Kinematics and Dynamics

- ❖ Statics

- ❖ the study of equilibrium and its relation to **forces**

- ❖ Kinetics

- ❖ the study of motion and its relation to **forces**

- ❖ Kinematics

- ❖ the study of observed motions without regard for circumstances causing them

classical mechanics

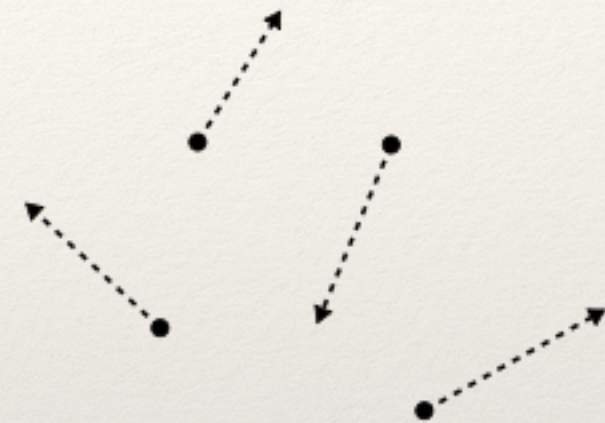
- ❖ Dynamics

- ❖ Statics + Kinetics

Particle and Rigid Body

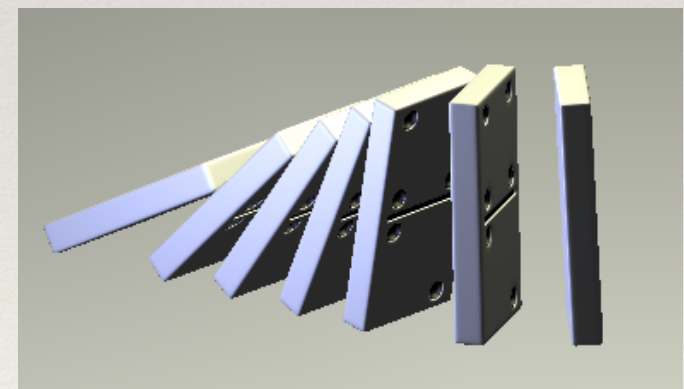
- ❖ Particle

- ❖ very very tiny object with mass
 - ❖ volume can be ignored
- ❖ rocket trajectory analysis
 - ❖ the volume of the rocket can be ignored
 - ❖ rocket can be regarded a particle



- ❖ Rigid body (= idealised solid body)

- ❖ object with mass and volume
- ❖ the shape does not change in any condition
- ❖ valid animation of rigid body = rotation and translation



Velocity

- ❖ velocity
 - ❖ a vector
 - ❖ speed: the magnitude of velocity
 - ❖ velocity = (speed + direction)
 - ❖ direction of velocity
 - ❖ moving direction
 - ❖ magnitude of velocity
 - ❖ ratio of moving distance to time

$$\mathbf{v} = \frac{\Delta \mathbf{s}}{\Delta t}$$

ratio of displacement to time interval

Instantaneous Velocity

- ❖ Reduce the time interval
 - ❖ the more precise velocity at the moment
 - ❖ if the time interval approaches to 0
 - ❖ instantaneous velocity

$$\mathbf{v} = \lim_{\Delta t \rightarrow 0} \frac{\Delta \mathbf{s}}{\Delta t} = \frac{d\mathbf{s}}{dt}$$

displacement

- ❖ integration of velocity

$$\int \mathbf{v} dt = \int d\mathbf{s}$$

- ❖ integration from t_1 to t_2

$$\int_{t_1}^{t_2} \mathbf{v} dt = \int_{s(t_1)}^{s(t_2)} d\mathbf{s}$$


- ❖ equivalent to the displacement

$$\int_{t_1}^{t_2} \mathbf{v} dt = \mathbf{s}(t_2) - \mathbf{s}(t_1) = \Delta \mathbf{s}$$

- ❖ If we know the velocity

- ❖ we can predict where the particle will be located in the future

$$\mathbf{v} = \lim_{\Delta t \rightarrow 0} \frac{\Delta \mathbf{s}}{\Delta t} = \frac{d\mathbf{s}}{dt}$$


$$\mathbf{v} dt = d\mathbf{s}$$

Acceleration

- ❖ Average acceleration

- ❖ ratio of velocity change to the time interval

$$\mathbf{a} = \Delta \mathbf{v} / \Delta t$$

- ❖ Instantaneous acceleration

$$\mathbf{a} = \lim_{\Delta t \rightarrow 0} \Delta \mathbf{v} / \Delta t = d\mathbf{v} / dt$$

- ❖ Integration of acceleration

$$\mathbf{a} dt = d\mathbf{v}$$

- ❖ velocity change

$$\int_{t_1}^{t_2} \mathbf{a} dt = \int_{\mathbf{v}(t_1)}^{\mathbf{v}(t_2)} d\mathbf{v} = \Delta \mathbf{v}$$

Constant Acceleration

- ❖ Simple problem for Kinematics
 - ❖ example of constant acceleration: gravity
 - ❖ gravitational acceleration
 - ❖ magnitude: 9.81 m/s^2
 - ❖ direction: downwards $(0, -1, 0)$
- ❖ We can easily “integrate the acceleration” to compute
 - ❖ velocity change at every time
 - ❖ this makes it possible to know the velocity at every time: $v(t)$

Gravitational Acceleration

- ❖ gravitational acceleration: g
 - ❖ velocity change and the integration of acceleration

$$\int_{\mathbf{v}(t_1)}^{\mathbf{v}(t_2)} d\mathbf{v} = \int_{t_1}^{t_2} \mathbf{g} dt$$
$$\mathbf{v}_2 - \mathbf{v}_1 = \mathbf{g}(t_2 - t_1)$$

- ❖ If we know every thing at time t_1
 - ❖ We can easily predict the velocity at time t_2

$$\mathbf{v}_2 = \mathbf{g}t_2 - \mathbf{g}t_1 + \mathbf{v}_1$$

- ❖ if $t_1=0$ ($t_2=t$)

$$\mathbf{v}_2 = \mathbf{v}_1 + \mathbf{g}t$$

Velocity as f(Displacement)

- ❖ Another differential equation

$$\mathbf{v} \frac{d\mathbf{v}}{dt} = \frac{ds}{dt} \mathbf{a}$$

$$\mathbf{v} d\mathbf{v} = \mathbf{a} ds$$

- ❖ Integration $\int_{\mathbf{v}_1}^{\mathbf{v}_2} \mathbf{v} d\mathbf{v} = \int_{s_1}^{s_2} \mathbf{a} ds$

$$\frac{1}{2} \mathbf{v}^2 \Big|_{\mathbf{v}_1}^{\mathbf{v}_2} = \mathbf{a} s \Big|_{s_1}^{s_2} = \mathbf{g} s \Big|_{s_1}^{s_2}$$

- ❖ Relation between velocity and displacement

$$\frac{1}{2} (\mathbf{v}_2^2 - \mathbf{v}_1^2) = \mathbf{g} (s_2 - s_1)$$

$$\mathbf{v}_2^2 = 2\mathbf{g} (s_2 - s_1) + \mathbf{v}_1^2$$

Location

- ❖ Velocity and displacement

$$\mathbf{v} dt = ds$$

$$(\mathbf{v}_1 + \mathbf{g}t) dt = ds$$

- ❖ Integration

$$\int_0^t (\mathbf{v}_1 + \mathbf{g}t) dt = \int_{\mathbf{s}_1}^{\mathbf{s}_2} ds$$

$$v_1 t + \frac{1}{2} \mathbf{g} t^2 = \mathbf{s}_2 - \mathbf{s}_1$$

- ❖ Location at time t

$$\mathbf{s}_2 = v_1 t + \frac{1}{2} \mathbf{g} t^2 + \mathbf{s}_1$$

Kinematic simulation

```
#include "KinematicsSimulator.h"

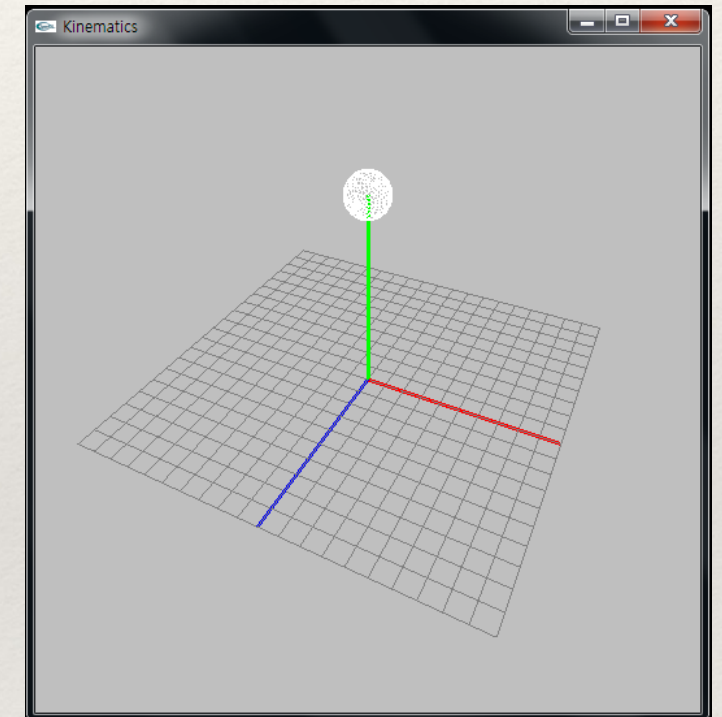
CKinematicSimulator::CKinematicSimulator() : CSimulator() {}

void CKinematicSimulator::init() {
    initialLoc.set(0,1,0);
    initialVel.set(0,3,0);
    gravity.set(0.0, -9.8, 0.0);
    currentLoc = initialLoc;
    particle.setPosition(currentLoc[0], currentLoc[1], currentLoc[2]);
    particle.setRadius(0.1);
}

void CKinematicSimulator::doBeforeSimulation(double dt, double currentTime) {}

void CKinematicSimulator::doSimulation(double dt, double currentTime) {
    currentLoc = initialLoc
    + currentTime*initialVel
    + (0.5 * currentTime * currentTime) * gravity ;
    particle.setPosition(currentLoc[0], currentLoc[1], currentLoc[2]);
    particle.drawWithGL();
}

void CKinematicSimulator::doAfterSimulation(double dt, double currentTime) {}
```



Particle explosion with Kinematics

❖ KinematicSimulator.cpp

```
#include "KinematicsSimulator.h"
```

```
CKinematicSimulator::CKinematicSimulator() : CSimulator() {}
```

```
void CKinematicSimulator::init() {
```

```
    for(int i=0;i<NUMPARTS;i++)particle[i].randomInit();
```

```
}
```

```
void CKinematicSimulator::doBeforeSimulation(double dt, double currentTime) { }
```

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

```
    for(int i=0;i<NUMPARTS;i++){
```

```
        particle[i].simulate(dt, currentTime);
```

```
        particle[i].drawWithGL(POINT_DRAW);
```

```
    }
```

```
}
```

```
void CKinematicSimulator::doAfterSimulation(double dt, double currentTime) { }
```

Particle explosion with Kinematics

❖ Particle.cpp

```
void CParticle::randomInit() {
    double speed = rand()%10000 / 10000.0 + 1.0;
    double theta = 2.0*3.141592 * (rand()%10000 / 10000.0);
    double phi = 2.0*3.141592 * (rand()%10000 / 10000.0);
    double vx,vy,vz; // spherical coord to cartesian coord
    vy = speed*cos(phi)+2.0;
    vx = speed*cos(theta)*sin(phi);
    vz = speed*sin(theta)*sin(phi);
    initialLoc.set(0,1,0);
    initialVel.set(vx, vy, vz);
    gravity.set(0.0, -9.8, 0.0);
    radius = 0.01;
    currentLoc = initialLoc;
}

void CParticle::simulate(double dt, double et) {
    currentLoc = initialLoc + et*initialVel + (0.5 * et * et) * gravity ;
    setPosition(currentLoc[0], currentLoc[1], currentLoc[2]);
}
```


Result

