# Particle Dynamics

[Physics Basics]

- Mass (질량) [kg]

- Gravitional Acceleration (중력 가속도) [$m/s^2$

- Force (힘) [$kg*m/s^2$]

- Weight (무게) [kgf]

- Ground Reaction Force (지면 반력) [$kg*m/s^2$]

지면과 물체가 맞닿을 지, 지면 방향의 반대로 작용하는 힘

- Linear Momentum (선운동량) [N*s],[kg*m/s]

- Law of Momentum Conservation (운동량 보존의 법칙)

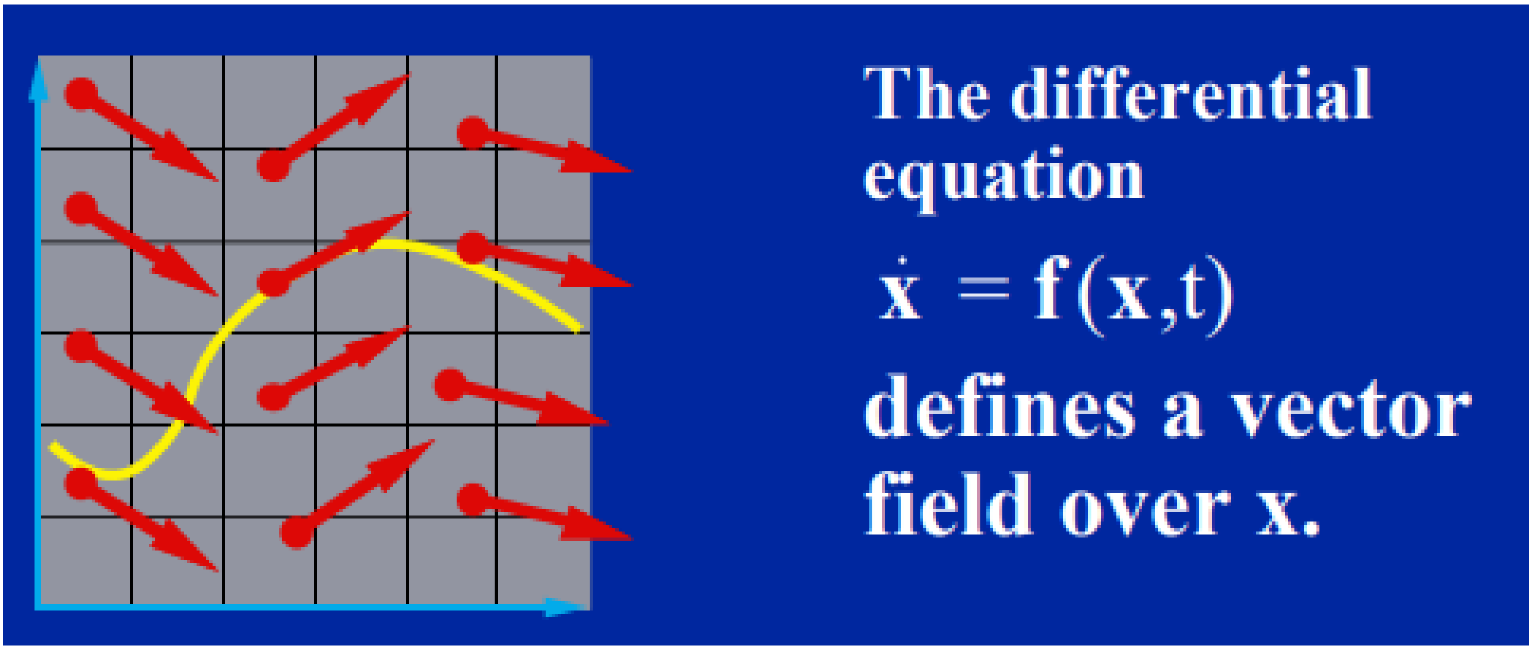
어떤 계의 외부에서 힘이 가해지지 않는다면,
계의 총 운동량은 보존

- Impulse (충격량) [N*s]

# Particle Dynamics

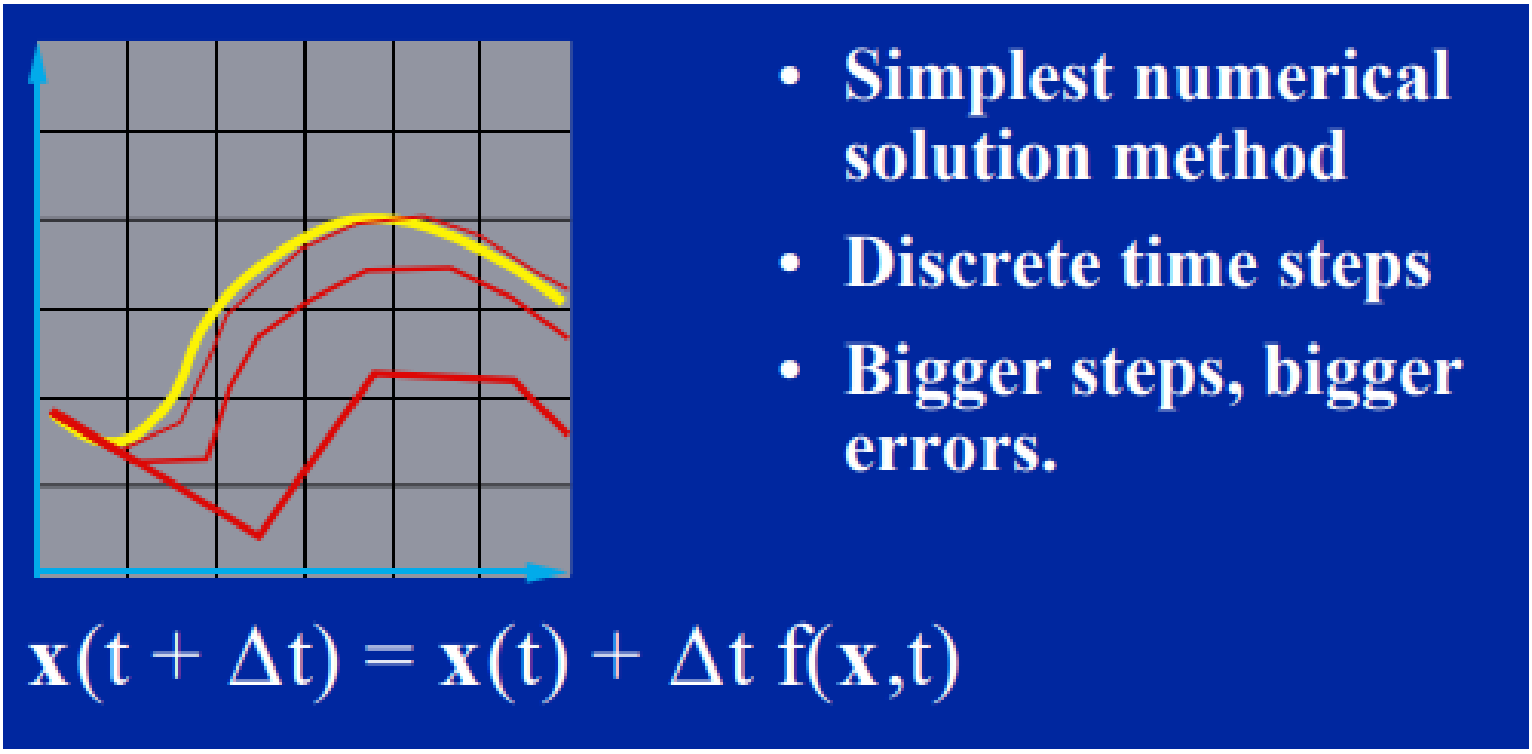
[Differential Equation]

- Vector Field




> Position(좌표 정보), Velocity(속도 정보), Force(힘 정보)

- Euler's Method

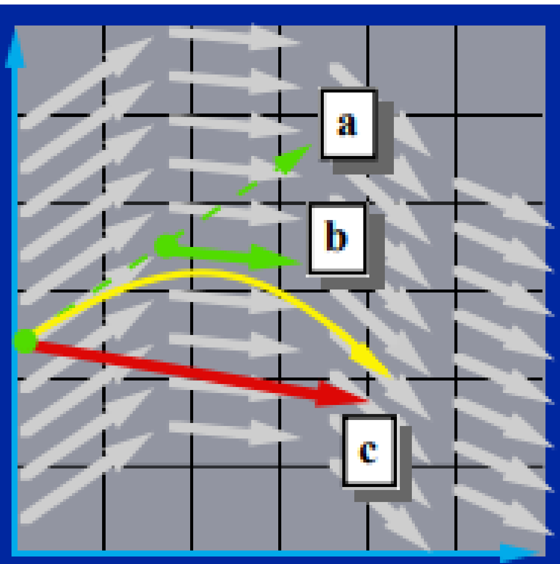



> x(좌표, 속도) 값을 초기 기울기와 시간간격을 통해 구함
> 정확성X, 안전성X의 문제점을 지니고 있음

# Particle Dynamics

## [Differential Equation]

- **The Mid Point Method**




**a. Compute an Euler step**

$$\Delta x = \Delta t\, f(x,t)$$

**b. Evaluate f at the midpoint**

$$f_{mid} = f\left(\frac{x + \Delta x}{2}, \frac{t + \Delta t}{2}\right)$$

**c. Take a step using the midpoint value**

$$x(t + \Delta t) = x(t) + \Delta t\, f_{mid}$$

**> x(좌표, 속도) 값을 중간값의 기울기를 통해서 구함**

# Particle Dynamics

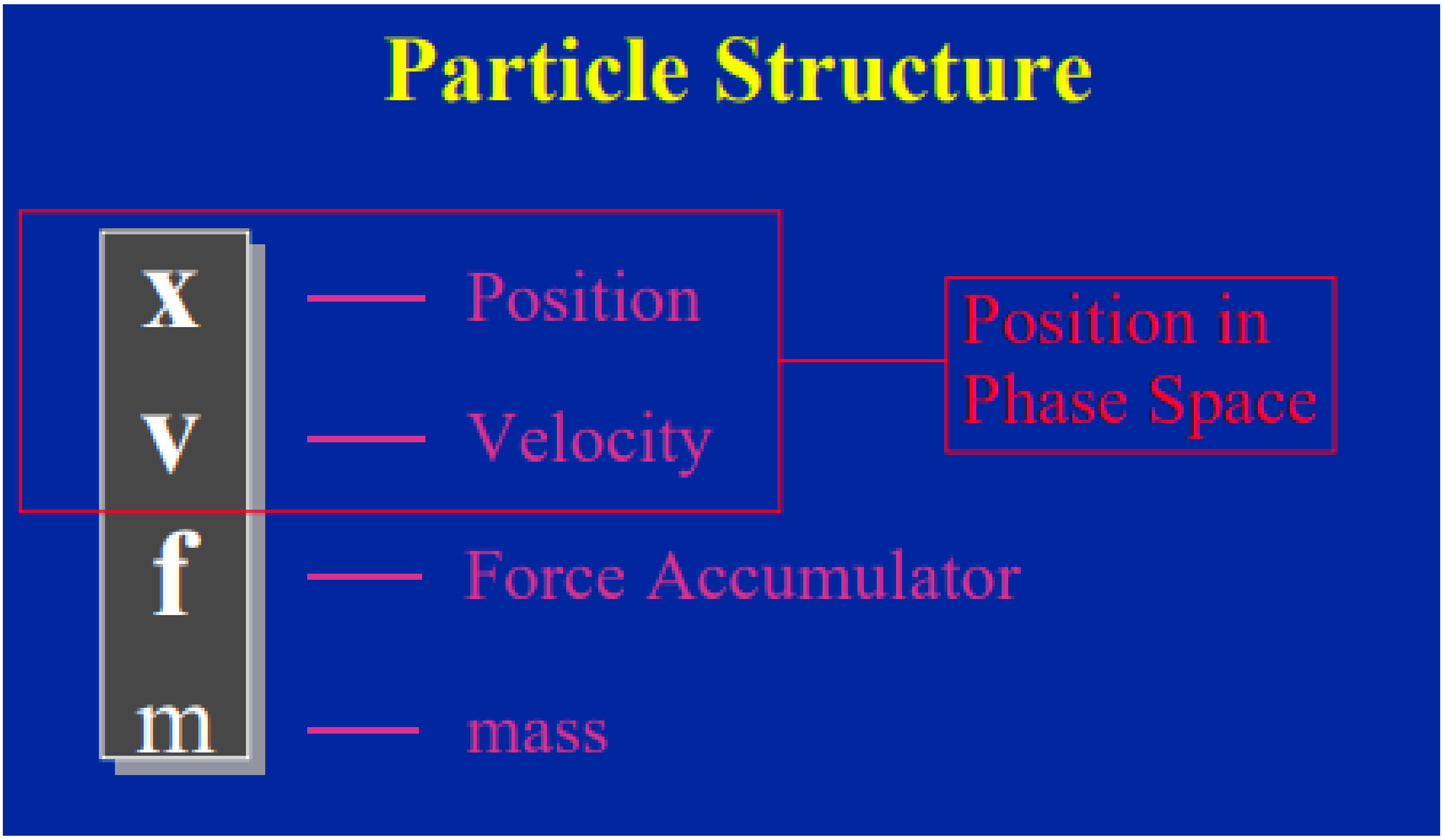
[Particle Dynamics]

- A Newtonian Particle

$$\ddot{x} = \frac{f(x, \dot{x}, t)}{m}$$

> 가속도 : 위치, 속도, 시간, 무게로 결정

- Particle Structure




> 입자 1개가 지니는 정보
>> 위치 : <x,y,z> 형태의 Vector
>> 속도 : <x,y,z> 형태의 Vector
>> 힘 : <x,y,z> 형태의 Vector
>> 무게

# Particle Dynamics

## [Particle Dynamics] - Force

- Gravity

*Force Law:*

$$f_{grav} = mG$$

> 중력 : 무게 * G(중력 상수) [ y축 방향]

- Drag Force

*Force Law:*

$$f_{drag} = -k_{drag}\mathbf{v}$$

> 항력 : -Kd(항력계수) * 속도
>> (-) : 운동방향과 반대로 작용
>> Kd : 항력 계수
>> v : 속도

- Damped Spring

*Force Law:*

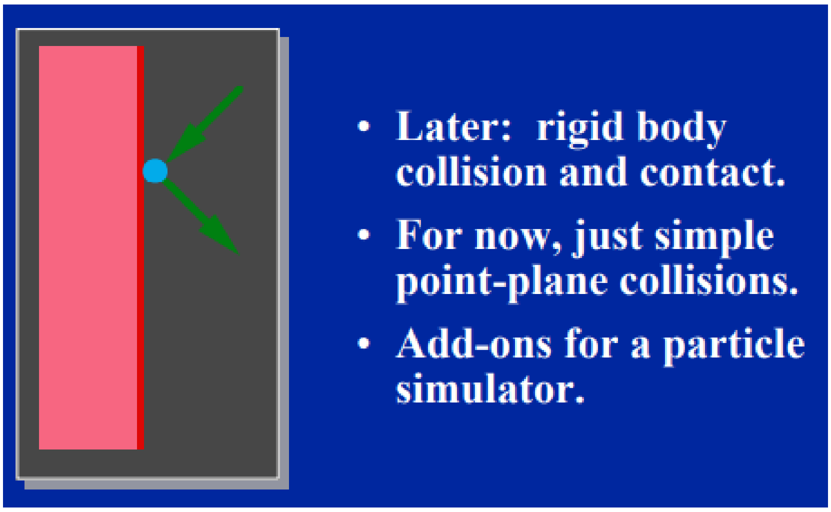
$$f_1 = -\left[k_s(|\Delta x| - r) + k_d\left(\frac{\Delta v \cdot \Delta x}{|\Delta x|}\right)\right]\frac{\Delta x}{|\Delta x|}$$

$$f_2 = -f_1$$


> 탄성력
>> (-) : (현재 길이 – 원래 길이)의 반대 방향으로 작용
>> Ks : 용수철 상수
>> Δx : 용수철 길이의 차이(1점과 2점)
>> r : 원래 용수철 길이
>> Kd : 항력 계수
>> Δv : 속도의 차이(1점과 2점)
>> Δv* Δx : 속도차와 길이차 벡터의 내적
>> 1점에 작용하는 탄성력 = (-1) * 2점에 작용하는 탄성력

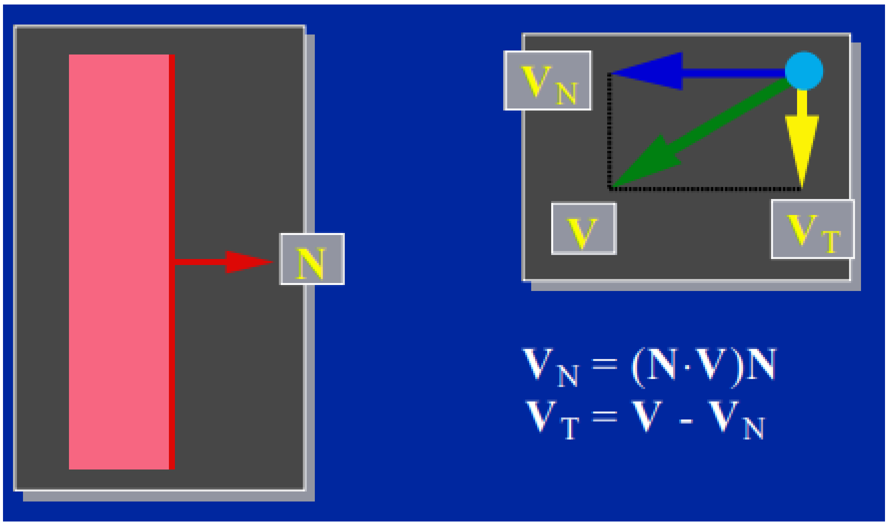
# Particle Dynamics

## [Particle Dynamics] - Bouncing off the walls




> 반응의 결과
>> 충돌
>> 접촉

- Normal and Tangential Components



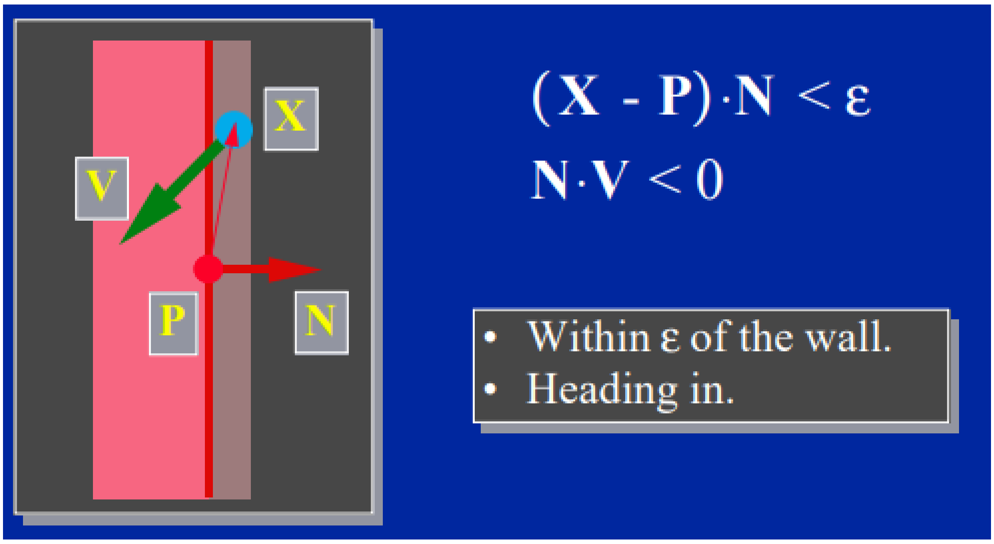

> 충돌하기까지의 속도
>> N : 충돌면 법선벡터
>> VN : 충돌면 수직속도
>> VT : 충돌면 수평속도

>> 수평속도 유지
>> 수직속도 반대로

# Particle Dynamics

## [Particle Dynamics] - Collision and Contact

- Collision




> 충돌 판정

>> (X-P)*N < e

>>> X : 점의 위치

>>> P : 충돌면의 임의의 점(중간점으로 생각)

>>> N : 충돌면의 법선 벡터

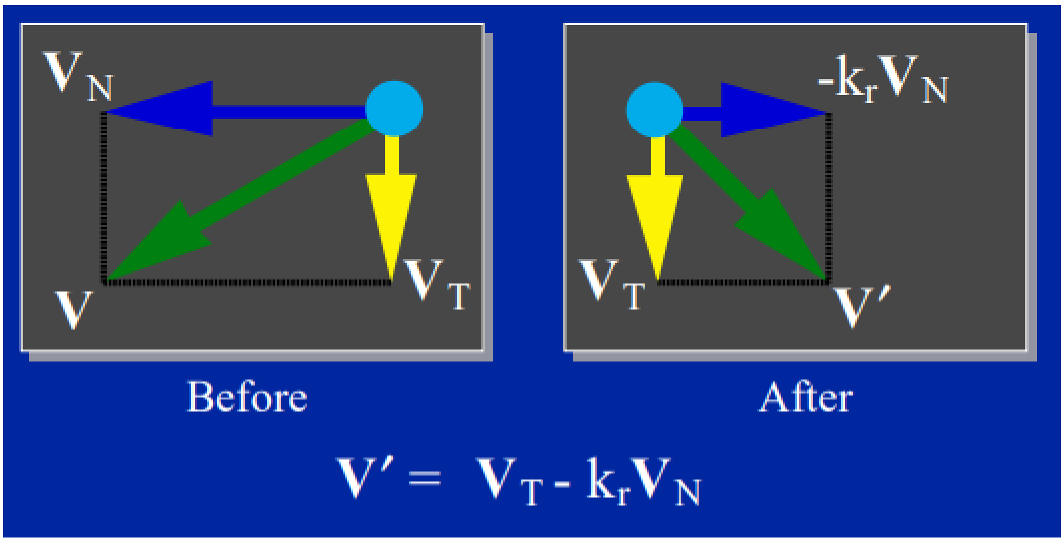
>>> e : 0으로 넣을 시, 인식하기 어려움
임의의 작은 값으로 지정 or 점의 반지름

>>> 뜻 : 점(X)가 벽면에 거의 근접한 상태인가?

>> N*V < 0

>>> V : 점의 속도

>>> 뜻 : 점(X)가 벽면에 가까워지는가?




> 반응의 결과

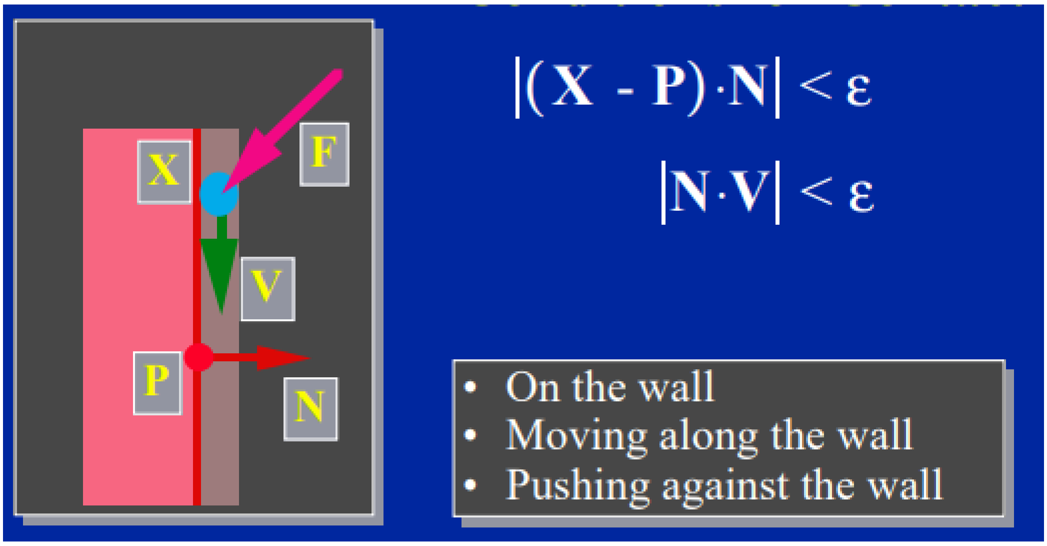
>> Kr : 반발계수??

>> 수직속도 반대

# Particle Dynamics

## [Particle Dynamics] - Collision and Contact

\- Contact




\> 접촉 판정

\>> |(X-P)*N| < e

\>>> X : 점의 위치

\>>> P : 충돌면의 임의의 점(중간점으로 생각)

\>>> N : 충돌면의 법선 벡터

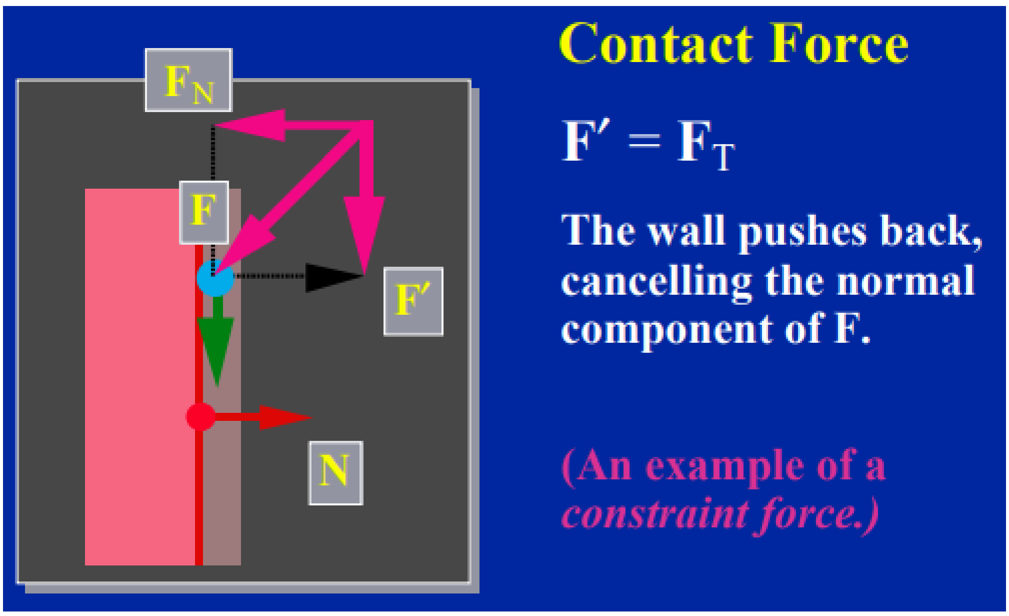
\>>> e : 0으로 넣을 시, 인식하기 어려움
임의의 작은 값으로 지정 or 점의 반지름

\>>> 뜻 : 점(X)가 벽면에 거의 근접한 상태인가?

\>> |N*V| < e

\>>> V : 점의 속도

\>>> 뜻 : 점(X)의 N방향 속도가 거의 없는가?




\> 반응의 결과

\>> 지면 반력 작용

\>> N방향 총힘 0

\>> T방향 힘만 有

# Particle Dynamics

## [Particle Dynamics] - 구현

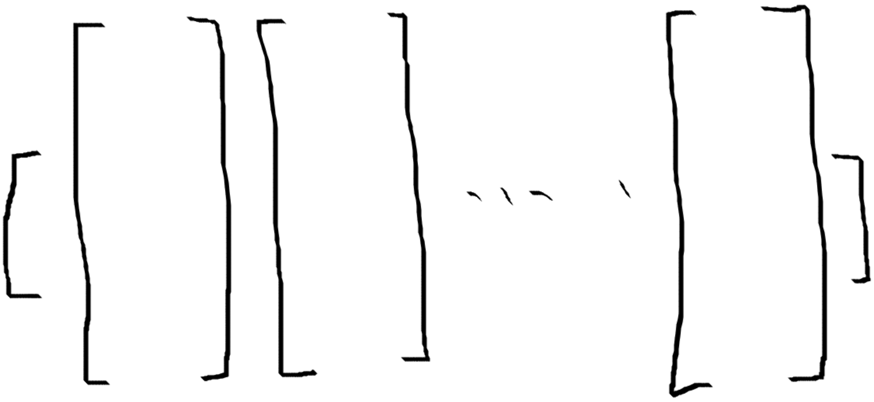
- 입자 정보

$$\begin{bmatrix} x \\ v \\ f \\ m \end{bmatrix}$$

> **다중 배열(선택)**
>
> X : [3] 배열 (x,y,z)
> V : [3] 배열 (x,y,z)
> F : [3] 배열 (x,y,z)
> m : [1] 배열 (m,m,m)

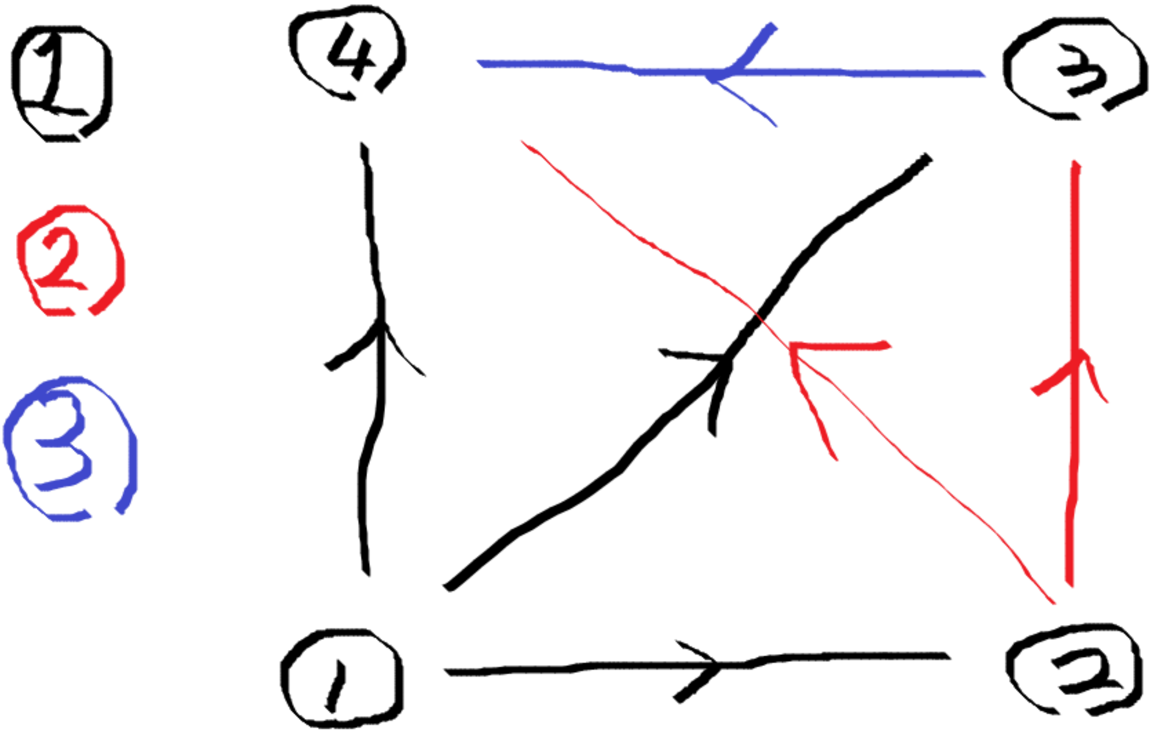
> **포인터 배열**
>
> X : [3] Vector의 주소
> V : [3] Vector의 주소
> F : [3] Vector의 주소
> m : m의 주소

- 입자 정보를 모은 배열



# Particle Dynamics

[Particle Dynamics] - 구현
- 점 및 선 그리기



> 순차적으로
>> 점
>>> 순서에 맞춰 점 찍음

>> 선
>>> 순서에 맞춰 선 그림
>>>> 낮은 것에서 높은 것으로 그림

[Particle Dynamics] - 코드
- 탄성력, 중력만 구현
https://github.com/Winteradio/_OPENGL_make_Spring2D

- 탄성력, 중력, 항력만 넣음(아직 구현 X)
https://github.com/Winteradio/_OPENGL_make_Spring2D_Force

# 강화 학습

## [Chapter 1. 강화 학습이란]

### [1.1] 지도 학습과 강화 학습

- **지도 학습 : 지도자(혹은 정답)가 있는 상태에서 배우는 것**
  > 학습 데이터
  >> 학습 데이터 안의 인풋과 아웃풋 사이 관계를 통해 학습

  > 테스트 데이터
  >> 정답을 맞히고자 하는 데이터
  >> 학습 데이터 안에 없는 데이터

- **강화 학습 : 홀로 시행착오를 통해 배우는 것**

### [1.2] 순차적 의사결정 문제

- **순차적 의사결정 문제**
  > 각 상황의 행동이 다음 상황에도 영향을 줌
  > 이어진 행동을 잘 선택해야 하는 문제

### [1.3] 보상

- **보상 : 의사결정을 얼마나 잘하고 있는지 알려주는 신호**
  > 어떻게 X, 얼마나 O
  > 스칼라 : 다양한 변수 속에서 가중치를 주어 하나의 숫자로
  > 희소 & 지연

- **누적 보상 : 과정에서 받은 보상의 총합**

# 강화 학습

## [Chapter 1. 강화 학습이란]

### [1.4] 에이전트와 환경

- **에이전트(Agent) : 강화학습의 주체**

  > 어떤 액션을 할지 정하는 것이 주된 역할

- **환경(Environment) : 에이전트를 제외한 모든 요소**

  > 액션이 주어졌을 시, 결과에 영향을 주는 모든 요소

### [1.5] 강화학습의 위력

- **병렬성의 힘 : 머린 하나, 몸은 여러 개**

  > 서로 다른 경험을 쌓는 시간 단축

- **자가학습(self-learning)의 매력**

  > 다양한 시도를 통해 스스로 성장

# 강화 학습

## [Chapter 2. 마르코프 결정 프로세스]

### [2.1] 마르코프 프로세스 MP = (S,P)

- 상태의 집합 S
  > 가능한 상태를 모두 모은 집합

- 전이 확률 행렬 P
  > 상태 전이가 이루어질 확률 : 전이 확률
  > 각 상태에 대한 표로 표현할 수 있음(행렬)

- **마르코프 성질**
  > **"미래는 오직 현재에 의해 결정된다."(조건부 확률)**
  > 마르코프한 상태 : 체스판
  > 마르코프하지 않은 상태 : 특정 시점에 따라 움직이는 차

### [2.2] 마르코프 리워드 프로세스 MRP = (S,P,R,γ)

- 상태의 집합 S

- 전이 확률 행렬 P

- 보상함수 R
  > 어떤 상태 S에 도착했을 때 받는 보상

- **감쇠인자 γ**
  > **0 ~ 1 사이 숫자**
  > **현재 당장 얻는 보상이 더 중요함을 나타내는 인자**

# 강화 학습

## [Chapter 2. 마르코프 결정 프로세스]

### [2.2] 마르코프 리워드 프로세스 MRP = (S,P,R,γ)

- **리턴 Gt**
  > 에피소드 : 하나의 강화학습 시작부터 끝까지의 과정
  > 리턴 : t시점부터 미래에 받을 감쇠된 보상의 합
  >> $G_t = R_{t+1} + \gamma * R_{t+2} + \gamma^2 * R_{t+3} + \dots$

- **Γ의 존재 이유**
  > 수학적 편리성
  >> Gt의 유한성을 나타냄
  >> MRP가 무한한 스탭 동안 진행되어도 Gt는 유한한 값 지님

  > 사람의 선호 반영
  >> 사람은 미래의 보상보다 현재의 보상을 더 선호

  > 미래에 대한 불확실성 반영
  >> 미래의 보상에 대한 가치가 달라질 수 있음

- **밸류 Value**
  > 상태에 대한 가치(미래에 받을 보상에 대한)
  > 리턴의 기댓값에 대한 평가

- **상태 가치 함수**
  > 상태 S로부터 시작하여 얻는 리턴의 기댓값
  > $V(S) = E[\,G_t \mid S_t = S\,]$

# 강화 학습

## [Chapter 2. 마르코프 결정 프로세스]

### [2.3] 마르코프 결정 프로세스 MDP = (S,A,P,R,γ)

- 상태의 집합 S

- 액션의 집합 A

> 에이전트가 취할 수 있는 액션들을 모은 것

- 전이 확률 행렬 P

> 현상태 S, 에이전트 액션 A 선택 > 다음 상태 S′ 확률
> 같은 상태에서 같은 액션을 해도 다른 상태로 도달 가능

- 보상함수 R

> 상태 S에서 액션 A를 선택할 시 받는 보상

- 감쇠인자 γ

# 강화 학습

## [Chapter 2. 마르코프 결정 프로세스]

### [2.3] 마르코프 결정 프로세스 MDP = (S,A,P,R,γ)

- **정책 함수**

> 각 상태에서 어떤 액션을 선택할 지 정해주는 함수
> 각 상태에서 할 수 있는 모든 액션의 확률 값의 합 : 1
> $\pi(a|s) = P[A_t = a \mid S_t = s]$

- **상태 가치 함수**

> 상태 s부터 끝까지 π를 따라 움직일 때의 리턴의 기댓값
> $V_\pi(s) = E_\pi[G_t \mid S_t = s]$

- **액션 가치 함수**

> 상태 s에서 a를 선택, 그 후 π를 따라갈 때 리턴의 기댓값
> $q_\pi(s,a) = E_\pi[G_t \mid S_t = s, A_t = a]$

### [2.4] Prediction 과 Control

- **Prediction**

> π가 주어졌을 때 각 상태의 밸류를 평가하는 문제

- **Control**

> 최적 정책 π'를 찾는 문제
> 최적 가치 함수

# 강화 학습

## [Chapter 3. 벨만 방정식]

### [3.1] 벨만 기대 방정식

- 재귀함수 : 자기 자신을 호출하는 함수

- 0 단계

> $V\pi(St) = E\pi[\, r_{t+1} + \gamma * V\pi(S_{t+1})\,]$
> $q\pi(St, at) = E\pi[\, r_{t+1} + \gamma * q\pi(S_{t+1}, a_{t+1})\,]$

- 1 단계

> $V\pi(s) = \Sigma\, \pi(a|s) * q\pi(s,a)$
> $q\pi(s, a) = r(a,s) + \gamma * \Sigma\, P(a,s,s') * V\pi(s')$

- 2 단계

> $V\pi(s) = \Sigma\, \pi(a|s)\, (\, r(a,s) + \gamma * \Sigma\, P(a,s,s') * V\pi(s')\,)$
> $q\pi(s, a) = r(a,s) + \gamma * \Sigma\, P(a,s,s') * (\, \Sigma\, \pi(a'|s') * q\pi(s',a')\,)$

### [3.1] 벨만 최적 방정식

- 0 단계

> $V^*\pi(St) = \max(a)[\, E\pi[\, r_{t+1} + \gamma * V^*\pi(S_{t+1})\,]]$
> $q^*\pi(St, at) = E\pi[\, r_{t+1} + \gamma * q^*\pi(S_{t+1}, a_{t+1})\,]$

- 1 단계

> $V^*\pi(s) = \max(a)[\, q^*\pi(s,a)\,]$
> $q^*\pi(s, a) = r(a,s) + \gamma * \Sigma\, P(a,s,s') * V\pi(s')$

- 2 단계

> $V^*\pi(s) = \max(a)\,[\, r(a,s) + \gamma * \Sigma\, P(a,s,s') * V^*\pi(s')\,]$
> $q^*\pi(s, a) = r(a,s) + \gamma * \Sigma\, P(a,s,s') * (\max(a')\,[\, q^*\pi\,(s',a')])$

# 강화 학습

## [Chapter 4. MDP를 알 때의 플래닝]

### [4.1] 밸류 평가하기 – 반복적 정책 평가

- 반복적 정책 평가
  > 테이블의 값들을 초기화
  > 벨만 기대 방정식을 반복적으로 사용
  > 테이블의 값들을 조금씩 업데이트
  > MDP에 대한 모든 정보를 알 때 사용 가능

- 코드 제목 : MDP-그리드 테이블.ipynb
https://github.com/Winteradio/_REINFORMENT_LEARING_FOLDER/tree/main

### [4.2] 최고의 정책 찾기 – 정책 이터레이션

- 그리드 정책
  > 그리드 정책 : π′은 원래의 정책 π보다 나은 정책
  > 현재의 이익 최대화

- 평가와 개선의 반복
  > 정책 평가 : 반복적 정책 평가
  > 정책 개선 : 그리드 정책 생성
  >> 최적 정책, 최적 가치 생성

# 강화 학습

## [Chapter 4. MDP를 알 때의 플래닝]

### [4.3] 최고의 정책 찾기 – 밸류 이터레이션

- **밸류 이터레이션**

  **> 최적 정책이 만들어내는 최적 밸류만 쫓아감**

- **최적 밸류**

  **> 최적 정책을 따랐을 때 얻는 밸류**

- **코드 MDP-그리드 테이블(최적).ipynb**

**https://github.com/Winteradio/_REINFORMENT_LEARING_FOLDER/tree/main**