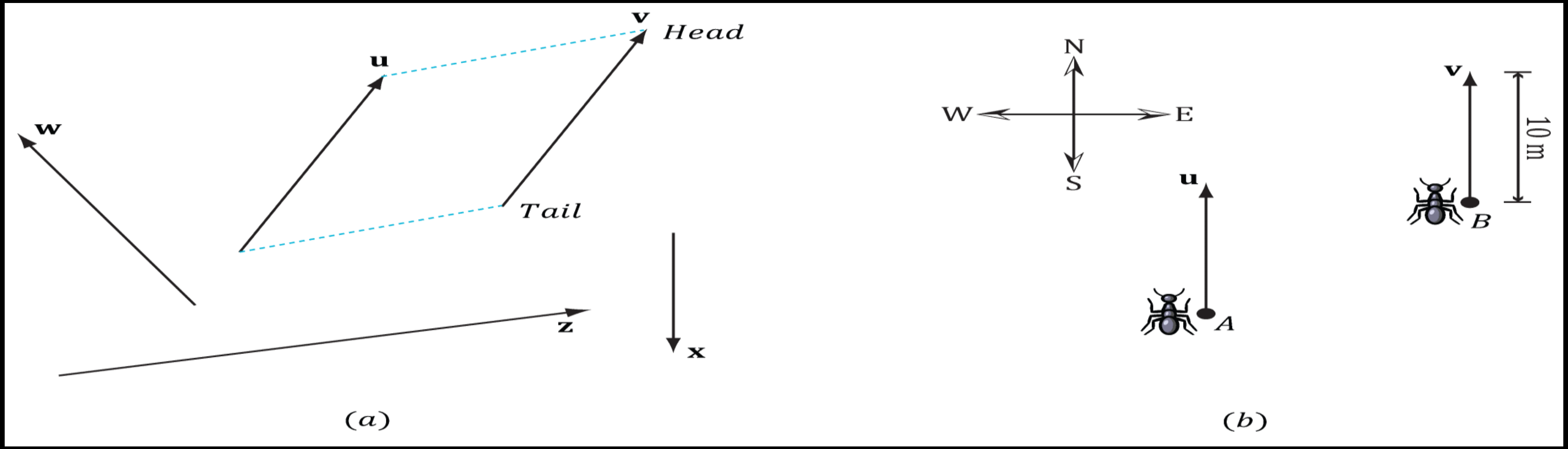
벡터

쿠재아이

김재경

벡터(Vector)란?

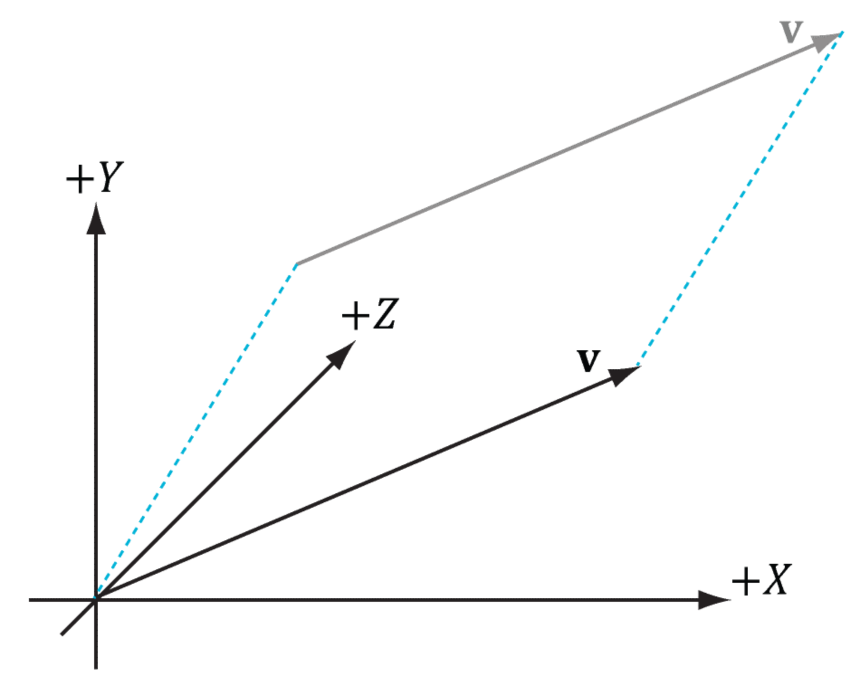
- 크기(magnitude)와 방향(direction)을 모두 가진 수량
- 예) 힘(force), 변위(displacement), 속도(velocity)
- 이동해도 크기와 방향이 같으면 같은 벡터다(밑에 그림에서 u = v)

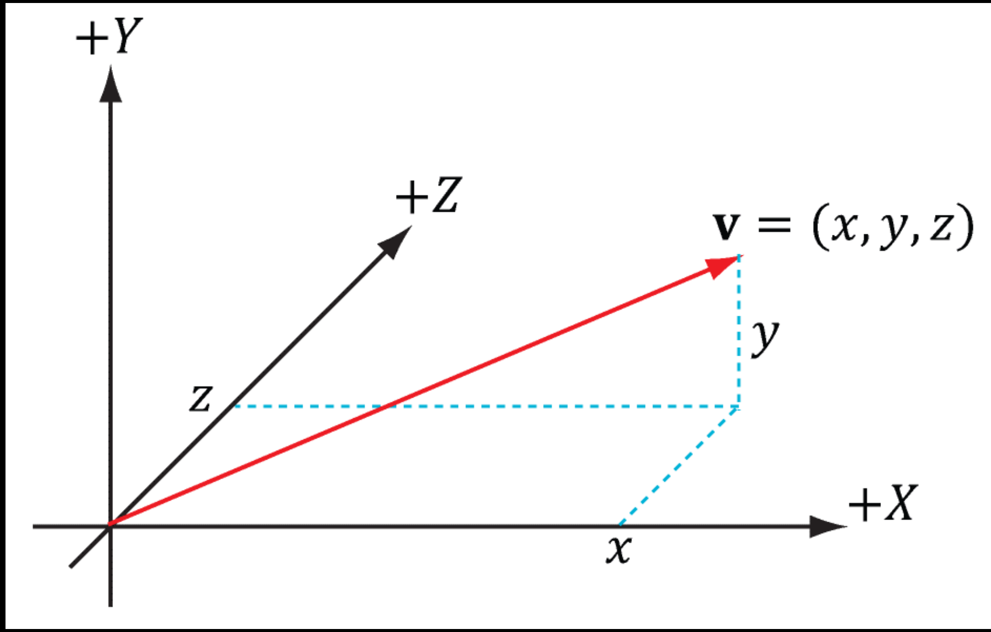


(a) 2차원 평면에 그려진 벡터들.   (b) 개미가 북쪽으로 10미터 나아가는 이동을 나타내는 벡터들

벡터와 좌표계

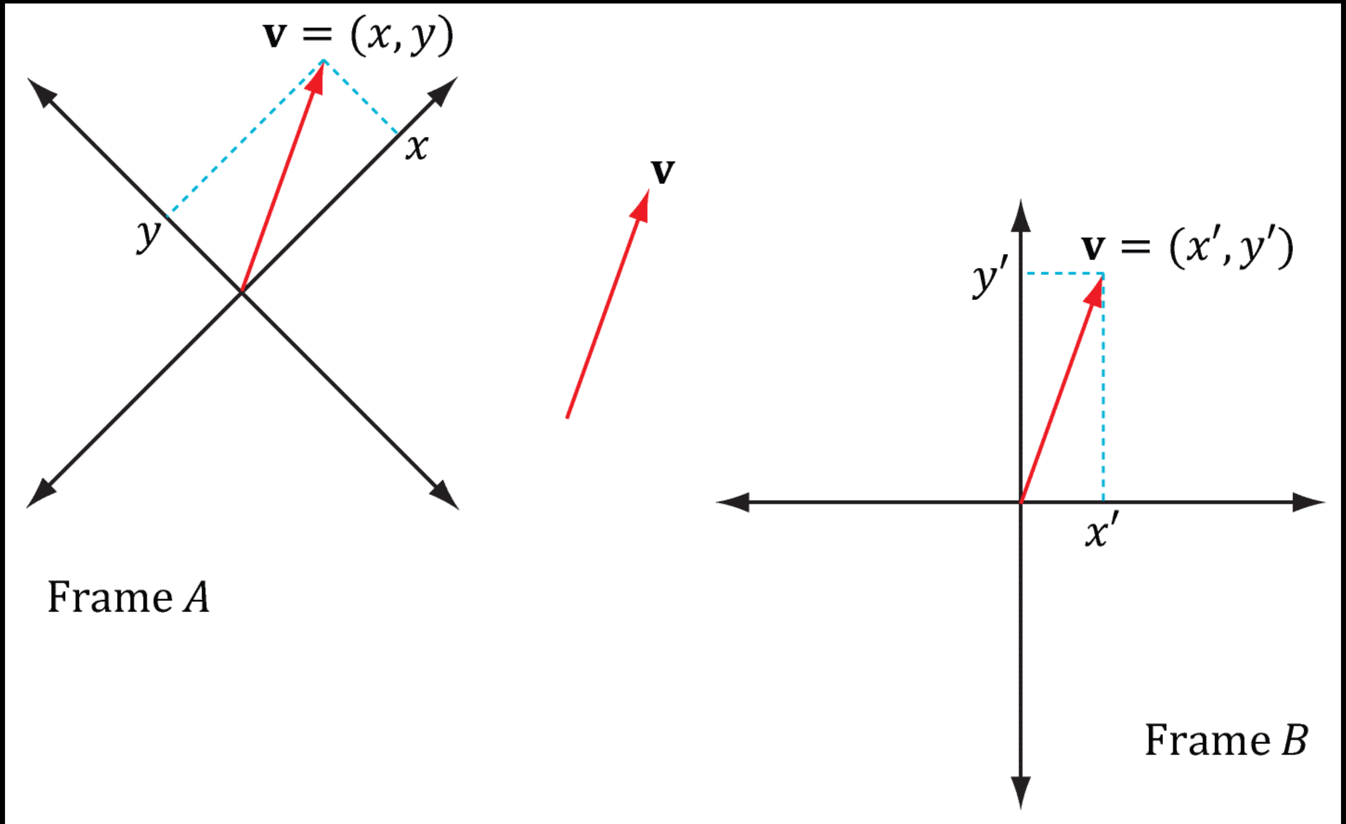
- 공간에 하나의 3차원 좌표계를 도입하고
- 모든 벡터를 그 꼬리가 그 좌표계의 원점과 일치하도록 이동하도록 함
- v = (x, y, z)로 표기할 수 있으며 부동소수점 값 3개로 표현할 수 있다





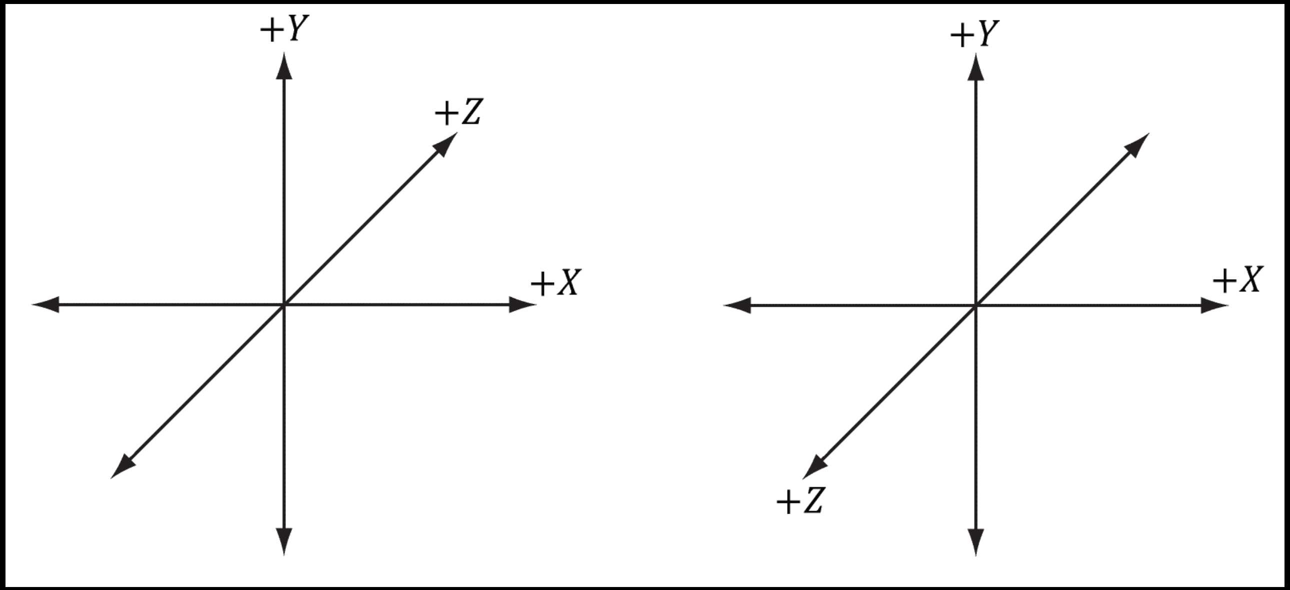
벡터와 좌표계

- 같은 벡터 v라도 기준계에 따라 좌표가 다르다



왼손잡이 좌표계 vs 오른손잡이 좌표계

- Directx3D에서는 왼손잡이 좌표계를 사용
- 왼손을 펴서 양의 x축 방향을 가리키게 하고 손가락들을 90° 구부려서 양의 y축 방향을 가리키게 하면 엄지손가락의 방향이 양의 z축 방향에 해당



왼손잡이 좌표계

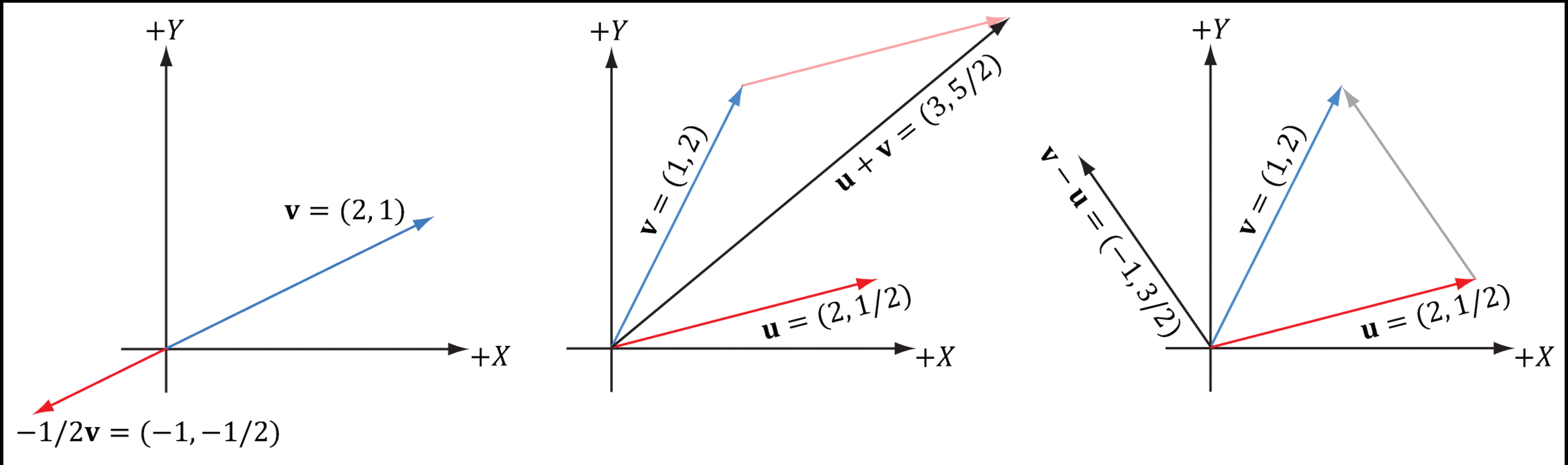
오른손잡이 좌표계

기본적인 벡터 연산들

- u = (u_x, u_y, u_z) 이고 v = (v_x, v_y, v_z) 일 때

1. $u_x = v_x$, $u_y = v_y$, u_z = v_z 이면 u = v
2. u + v = $(u_x + v_x, u_y + v_y, u_z + v_z)$
3. ku = (ku_x, ku_y, ku_z)
4. u − v = u + (−1 · v) = u + (−v) = $(u_x - v_x, u_y - v_y, u_z - v_z)$

기본적인 벡터 연산들



스칼라 곱셈의 기하학적 해석

벡터 덧셈의 기하학적 해석

벡터 뺄셈의 기하학적 해석

길이와 단위벡터

- 벡터의 크기(길이)는 이중 수직 선으로 표기 ($\|u\|$)
- 벡터 u = (x, y, z) 일 때
- $\|u\| = \sqrt{x^2 + y^2 + z^2}$

- 벡터를 방향을 나타내는 용도로만 사용하는 경우에는 길이가 중요하지 않음
- 이런 '방향 전용' 벡터는 길이를 1 (단위 길이)로 맞추어 두면 편리함
- 크기가 1인 벡터를 단위벡터(unit vector)라고 부름
- 임의의 벡터를 단위 벡터로 만드는 것을 정규화(normalization)라고 함
- $\|\widehat{u}\| = \frac{u}{\|u\|} = (\frac{x}{\|u\|}, \frac{y}{\|u\|}, \frac{z}{\|u\|})$

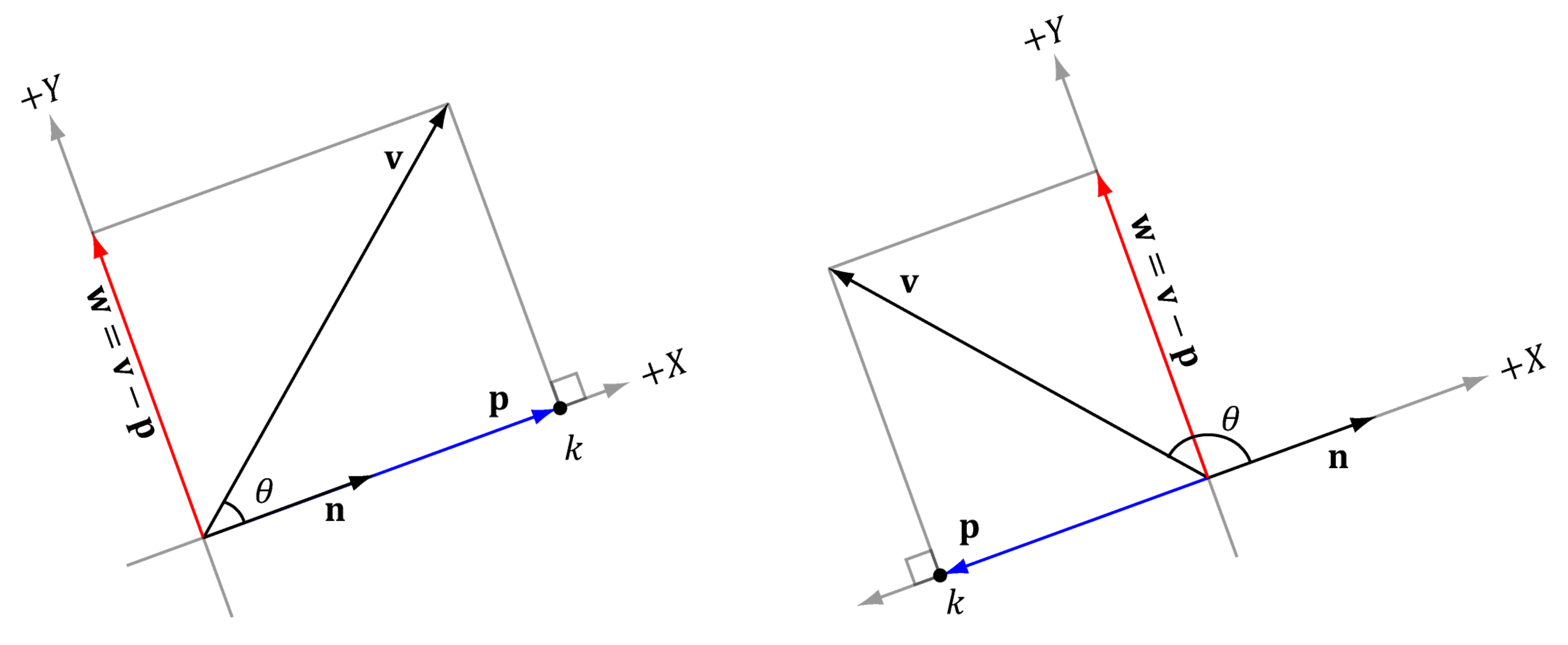
내적

- inner product. 점곱(dot product)이라고도 함
- 스칼라 값을 내는 벡터 곱셈

- u = (u_x, u_y, u_z)이고 v = (v_x, v_y, v_z) 일 때
- u · v = $u_x v_x + u_y v_y + u_z v_z$ 또는
- u · v = $\|u\|\|v\| \cos\theta$ (단, $0 \leq \theta \leq \pi$)

1. 만일 u · v = 0이면 u ⊥ v이다(즉, 두 벡터는 직교이다)
2. 만일 u · v $>$ 0이면 두 벡터 사이의 각도 θ는 90도보다 작다
3. 만일 u · v $<$ 0이면 두 벡터 사이의 각도 θ는 90도보다 크다

내적



벡터 v와 단위벡터 n이 주어졌을 때 p를 내적을 이용해서 v와 n으로 표현하는 공식 구하기

내적

1. p = kn을 만족하는 스칼라 k가 존재함을 알 수 있다
2. $\|n\|$ = 1이므로 반드시 $\|p\|$ = $\|kn\|$ = $|k|\|n\|$ = $|k|$
3. 삼각함수 법칙들을 적용하면 k = $\|v\| \cos\theta$
4. 따라서 p = kn = $(\|v\| \cos\theta)$n
5. 그런데 n은 단위벡터이므로
6. p = $(\|v\| \cos\theta)$n = $(\|v\| \cdot 1 \cos\theta)$n = $(\|v\|\|n\| \cos\theta)$n = (v · n)n
7. $\therefore k$ = (v · n)

- 이러한 p를 n에 대한 v의 직교투영(orthographic projection; 또는 정사영)이라고 한다
- p = proj_n(v)
- v를 하나의 힘으로 간주한다면 p는 힘 v 중에서 방향 n으로 작용하는 부분이라고 할 수 있다

내적

- 벡터 w = $perp_n$(v) = v − p는 힘 v 중에서 n의 수직 방향으로 작용하는 부분
- perp는 perpendicular(수직)을 뜻함
- v = p + w 로 분해될 수 있다

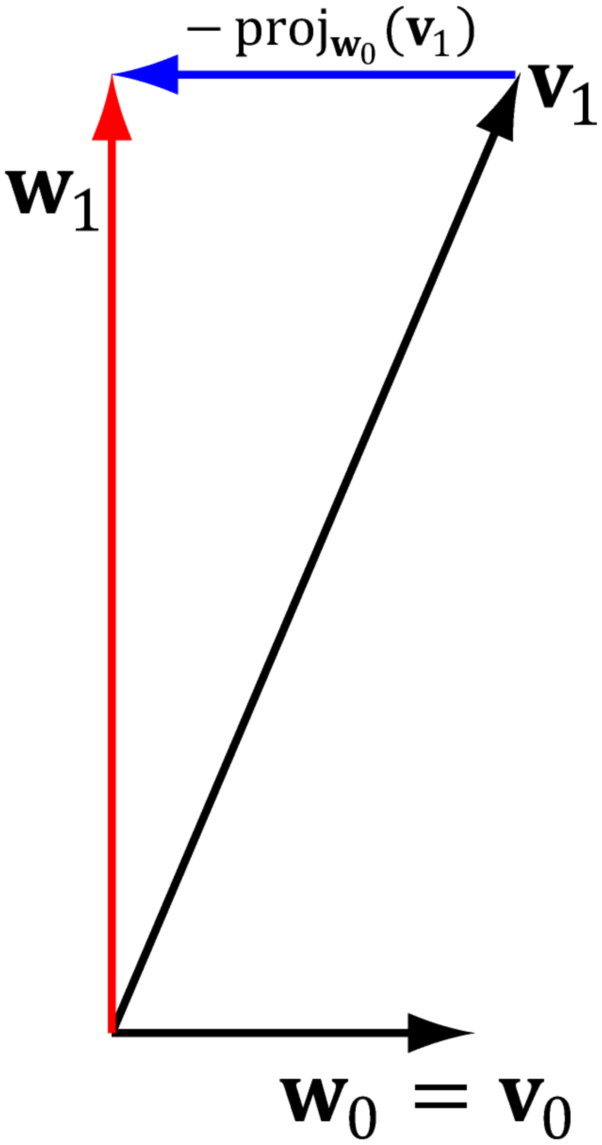
- n이 단위 길이가 아니면, 먼저 n을 정규화해서 단위 길이로 만들면 된다
- 위의 투영 공식에서 n을 단위 벡터 $\frac{n}{\|n\|}$으로 대체하면 좀 더 일반적인 투영 공식이 나온다

- p = $proj_n$(v) $= (v \cdot \frac{n}{\|n\|})\frac{n}{\|n\|} = \frac{(v \cdot n)}{\|n\|^2}n$

직교화

- 벡터 집합 $\{v_0, \cdots, v_{n-1}\}$의 모든 벡터가 단위 길이이고 서로 직교일 때 그러한 벡터 집합을 정규직교(orthonormal) 집합이라고 부른다
- 주어진 벡터 집합이 정규직교에 가깝지만 완전히 정규직교는 아닌 경우도 흔히 만나게 되는데 그런 벡터 집합을 정규직교벡터 집합으로 만드는 것을 직교화(orthogonalization)라고 한다
- 3D 그래픽에서는 정규직교 집합으로 시작했지만 수치 정밀도 문제 때문에 집합이 점차 정규직교가 아니게 되는 경우도 생긴다

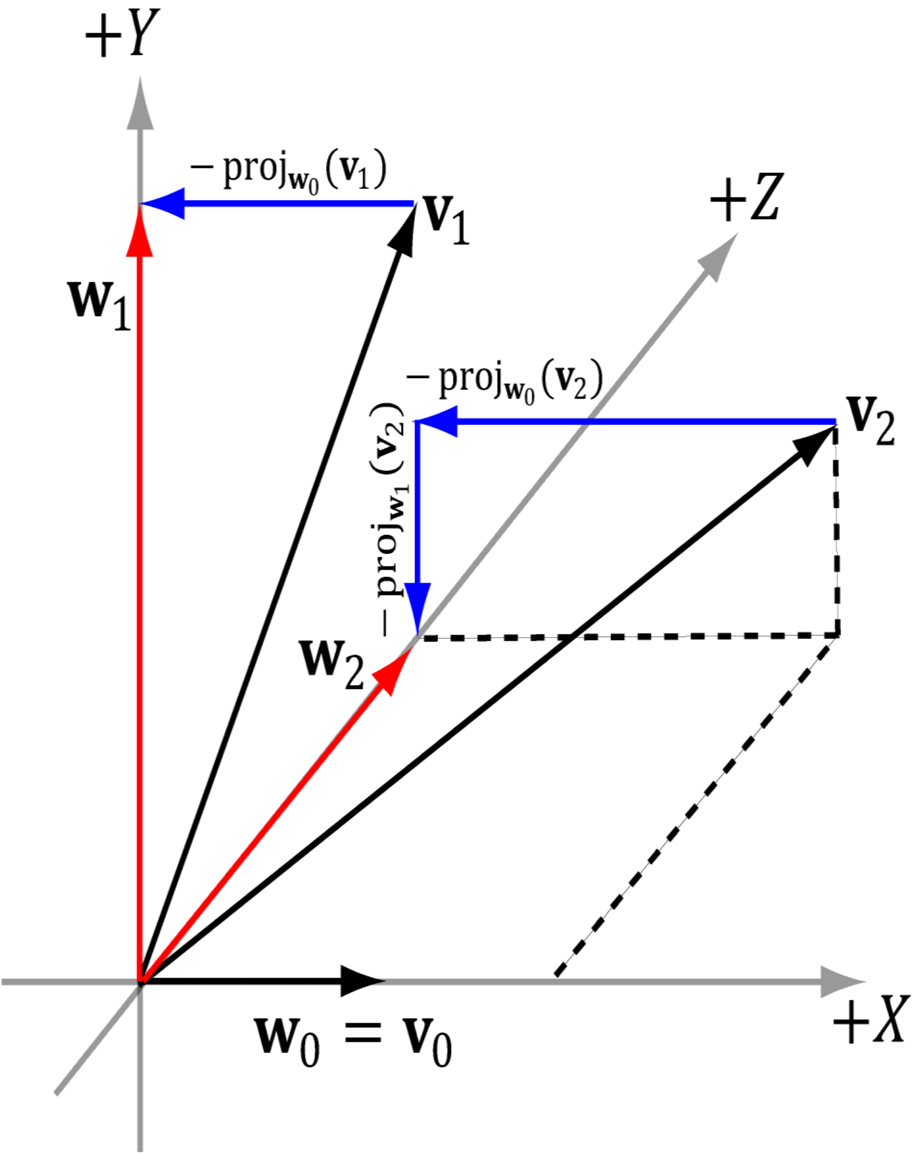
직교화



- 2차원
- 벡터 집합 $\{v_0, v_1\}$을 직교화해서 정규직교 집합 $\{w_0, w_1\}$을 얻는 과정

1. $w_0 = v_0$으로 시작해서, 벡터 v_1이 w_0과 직교가 되게 만든다
2. 이를 위해, w_0의 방향으로 작용하는 부분을 v_1에서 뺀다
3. $w_1 = v_1 - \mathrm{proj}_{w0}(v_1)$
4. 서로 직교인 벡터들의 집합 $\{w_0, w_1\}$이 만들어졌다
5. w_0 과 w_1을 정규화해서 단위 길이로 만들면 정규직교 집합이 완성된다

직교화



- 3차원
- 벡터 집합 $\{v_0, v_1, v_2\}$을 직교화해서 정규직교 집합 $\{w_0, w_1, w_2\}$을 얻는 과정

1. $w_0 = v_0$으로 시작해서, 벡터 v_1이 w_0과 직교가 되게 만든다
2. 이를 위해, w_0의 방향으로 작용하는 부분을 v_1에서 뺀다
3. $w_1 = v_1 - \text{proj}_{w0}(v_1)$
4. v_2가 w_0과 w_1 모두에 직교가 되게 한다
5. 이를 위해, w_0 방향으로 작용하는 부분과 w_1 방향으로 작용하는 부분을 v_2에서 뺀다
6. $w_2 = v_2 - \text{proj}_{w0}(v_2) - \text{proj}_{w1}(v_2)$
7. 서로 직교인 벡터들의 집합 $\{v_0, v_1, v_2\}$가 만들어졌다
8. w_0과 w_1, w_2를 정규화해서 단위 길이로 만들면 정규직교 집합이 완성된다

직교화

- 이전 방법을 일반화해서, n개의 벡터들의 집합 $\{v_0, \cdots, v_{n-1}\}$을 정규직교 집합 $\{w_0, \cdots, w_{n-1}\}$으로 직교화할 때는 그람–슈미트 직교화라고 하는 공정을 적용한다

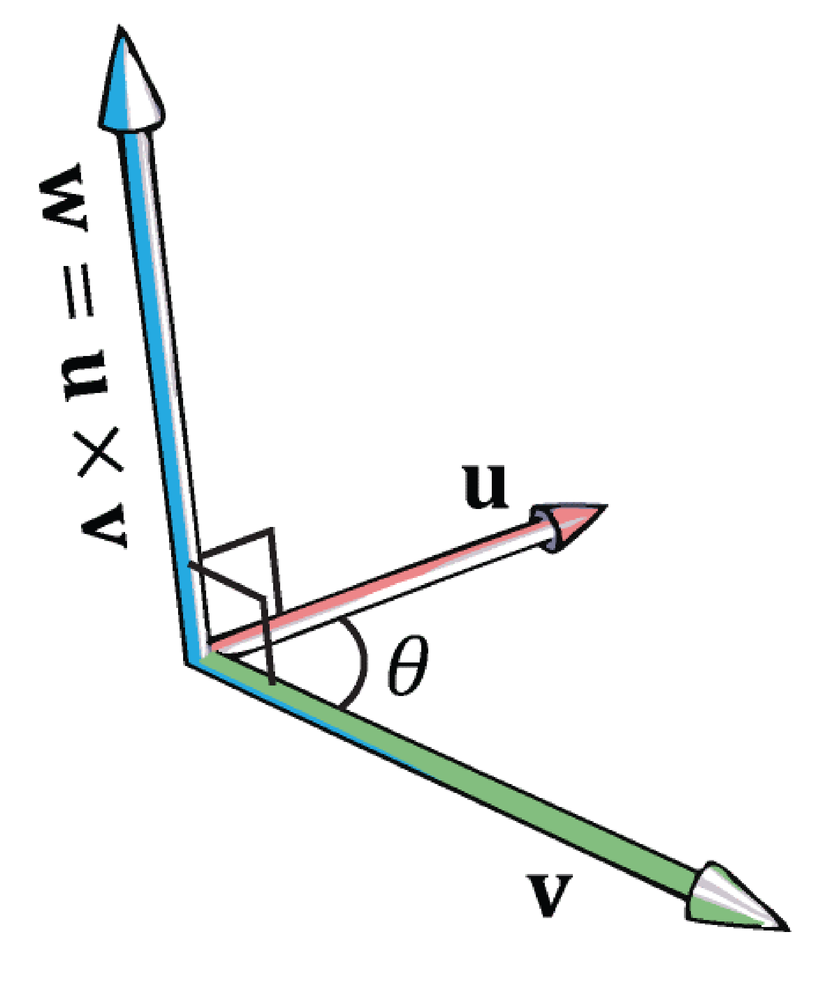
1. 기본 단계 : w_0 = v_0으로 설정한다
2. 1 ≤ i ≤ n−1에 대해 w_i = v_i – $\sum_{j=0}^{i-1}$ proj$_{wj}$ (v_i)로 설정한다
3. 정규화 단계 : w_i = $\frac{w_i}{\|w_i\|}$로 설정한다

외적

- Outer product. 가위곱(cross product)이라고도 함
- 외적의 결과는 벡터이다
- 오직 3차원 벡터에 대해서만 정의된다(2차원에서는 없음)

- 두 3차원 벡터 u와 v의 외적의 취하면 u와 v 모두에 직교인 또 다른 벡터 w가 나온다
- u = (u_x, u_y, u_z)이고 v = (v_x, v_y, v_z) 일 때
- w = u x v = $(u_y v_z - u_z v_y, u_z v_x - u_x v_z, u_x v_y - u_y v_x)$
- u x v $\neq$ v x u

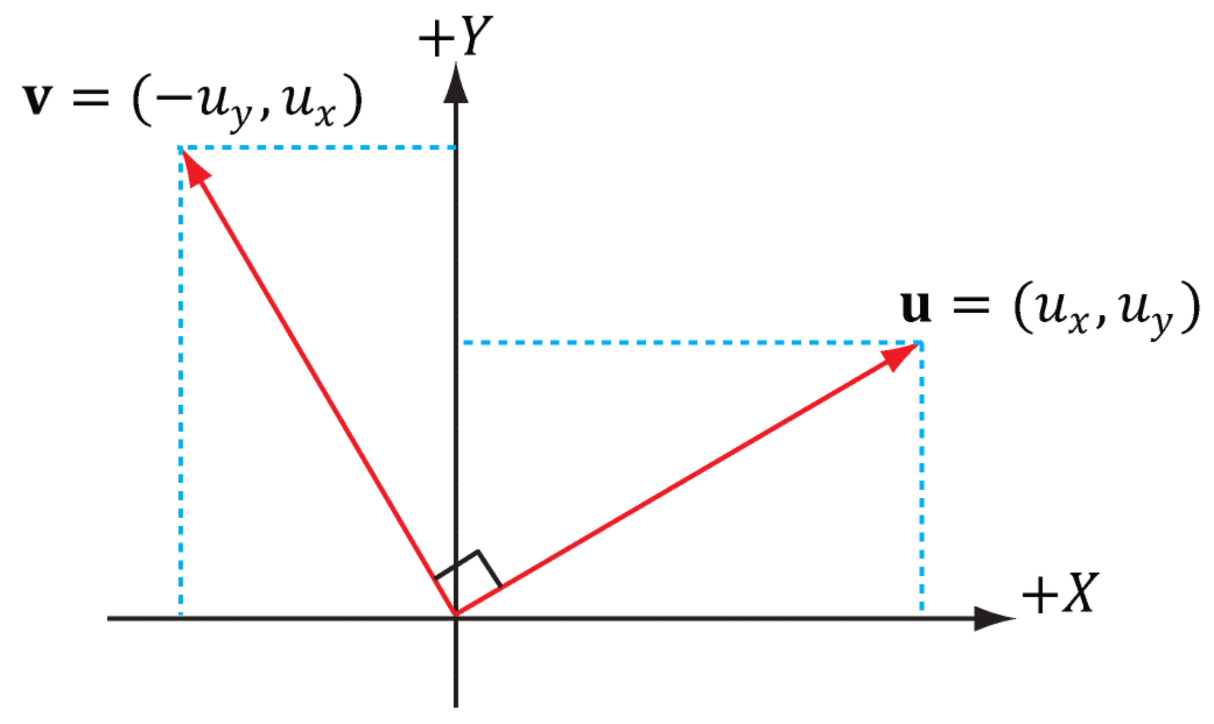
외적



- 왼손을 펼쳐서 첫 벡터 u의 방향을 가리킨 상태에서 손가락들을 둘째 벡터 v의 방향으로 말아쥐었을 때 엄지손가락이 가리키는 방향이 바로 w = u x v 방향이다.
- 이를 왼손 엄지 법칙이라고 부른다

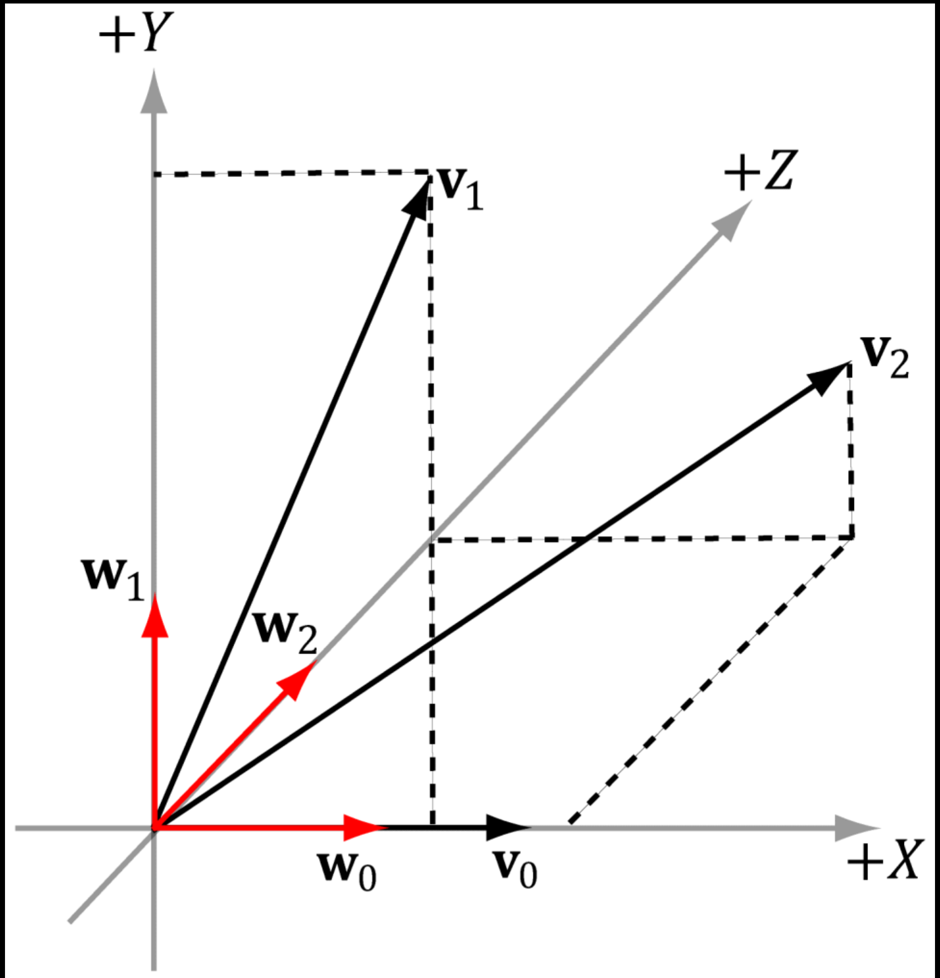
2차원 유사 외적(pseudo 2D cross product)

- 2차원에서는 두 벡터의 수직인 벡터가 존재하지 않음
- 그러나 하나의 2차원 벡터 u = (u_x, u_y)에 수직인 벡터 v는 구할 수 있다



- v = $(-u_y, u_x)$
- u ⊥ v
- u ⊥ -v

외적을 이용한 직교화



1. $w_0 = \dfrac{v_0}{\|v_0\|}$으로 설정한다
2. $w_2 = \dfrac{w_0 \times v_1}{\|w_0 \times v_1\|}$ 로 설정한다
3. $w_1 = w_2 \times w_0$로 설정한다
4. 이제 벡터 집합 $\{w_0, w_1, w_2\}$는 정규직교이다

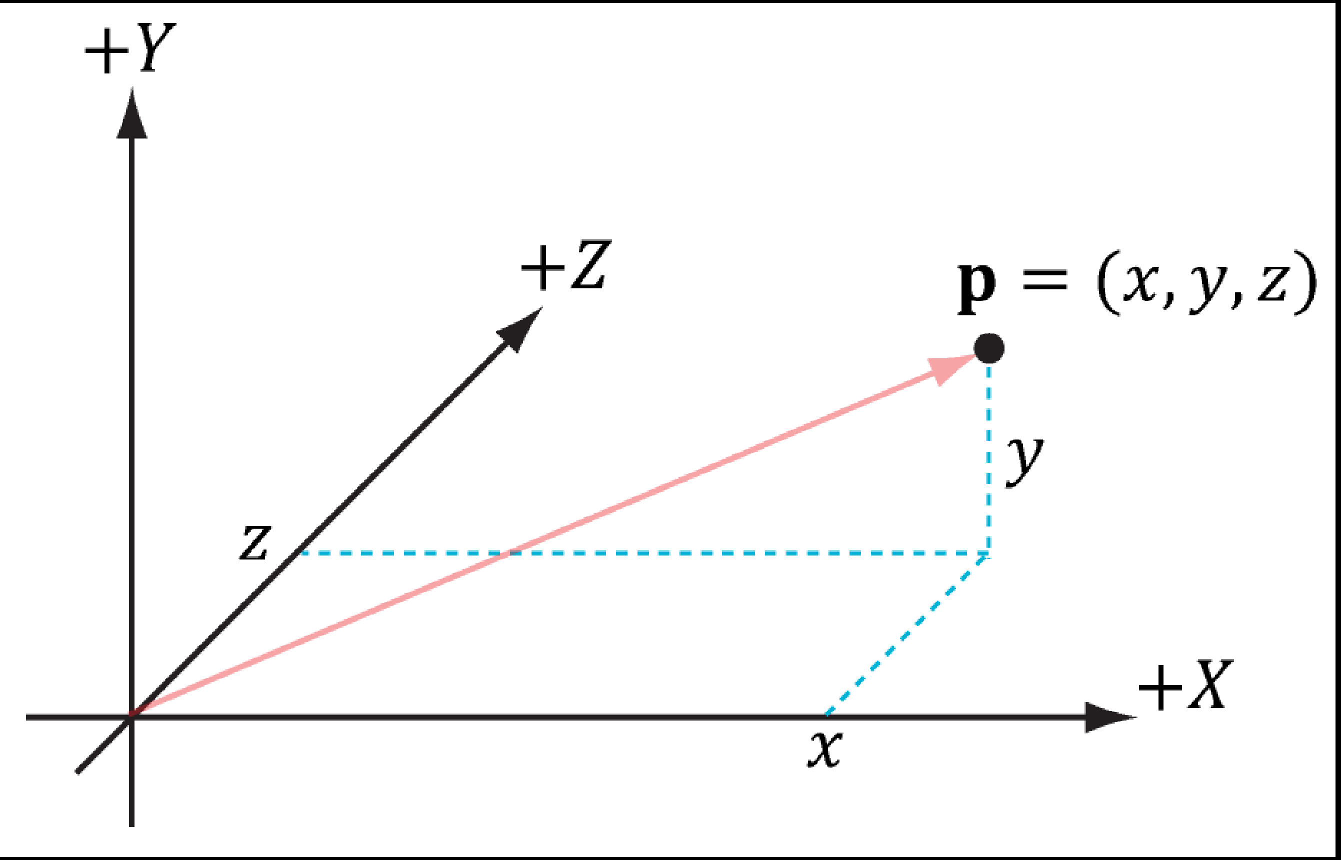
주의할 점

- 이 예는 $w_0 = \frac{v_0}{\|v_0\|}$으로 시작했는데, 이는 v_0에서 w_0으로 가는 방향이 변하지 않고 단지 그 길이만 변함을 뜻한다. 그러나 w_1과 w_2의 방향은 각각 v_1, v_2와 다를 수 있다. 응용의 성격에 따라서는, 방향이 바뀌지 않을 벡터로 어떤 벡터를 선택하느냐가 중요할 수 있다.

- 예를 들어 카메라의 방향을 세 개의 정규직교 벡터 $\{v_0, v_1, v_2\}$로 표현할 때, 여기서 셋째 벡터 v_2는 카메라가 바라보는 방향을 나타낸다. 이 벡터들을 직교화할 때 그 방향이 변하지 않게 하는 것이 바람직하므로, 이 직교화 알고리즘을 적용할 때에는 v_2로 시작하고 v_0과 v_1을 수정해서 벡터들을 직교화하는 것이 바람직하다.

점

- 지금까지 살펴본 벡터는 위치(position)를 서술하지 않음
- 그러나 3차원 그래픽 프로그램에서는 공간 안의 어떤 위치를 지정할 수 있어야 함



- 특정 좌표계를 기준으로 표준 위치에 있는 벡터를 3차원 공간 안의 한 위치를 나타내는 데 사용할 수 있다.
- 위치벡터라고 부름
- 벡터의 방향이나 크기가 아니라 벡터의 머리 끝의 좌표가 중요

점

- 점을 벡터로 표현하면 점에 대해 의미 없는 벡터 연산을 점(위치벡터)에 적용하는 실수를 할 수 있다
- ex) 기하학적으로 두 점의 합은 말이 되지 않음(아핀결합을 이용하면 가능하다고 함)

- 그러나 점에 대해서도 의미 있는 벡터 연산들도 존재함
- (그림 a) 두 점의 차 q - p 를, p에서 q로 가는 벡터라고 정의할 수 있음
- (그림 b) 점 p 더하기 벡터 v를 p의 위치를 v만큼 옮겼을 때 도달하는 점 q라고 정의 가능