1. 3D 그래픽 기초(3D Graphics Fundamentals)

(4) 벡터(Vector)의 이해

① 렌더링의 수학적 표현

사람은 실세계에서 그림을 그릴 때 눈(감각 기관)을 통하여 본 것을 그림으로 그린다. 그러나, 컴퓨터 프로그램으로 3차원의 가상 세계를 렌더링하여 다음과 같은 그림을 그리려면, 3차원의 가상 세계의 표현과 렌더링의 과정이 모두 수학적으로 표현되어야 한다. 게임 객체, 메쉬(모델), 조명(빛: Light), 물질(Material), 빛의 반사(Reflection), 빛의 굴절(Refraction), 그림자(Shadow) 등의 표현과 처리 과정이 수학적인 표현을 사용해야 한다(컴퓨터는 숫자를 잘 처리할 수 있는 기계 장치이기 때문에). 결론적으로 이러한 모든 것을 표현할 수 있는 수학적 수단은 벡터(Vector)와 행렬(Matrix)이다. 먼저, 3D 게임 프로그래밍에서 유용하고 필수적인 벡터의 내용에 대하여 이해를 해보자.

② 좌표계(Coordinate System)

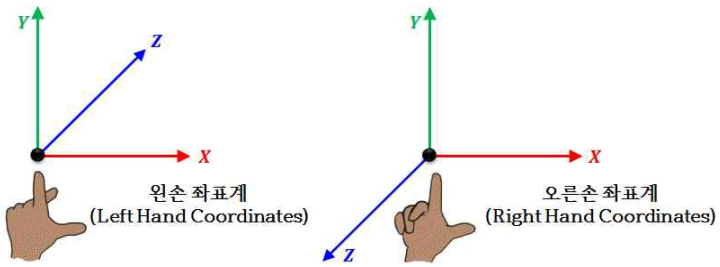
3D 그래픽 프로그램 또는 게임 프로그램을 작성하기 위해서는 위치(Position)와 방향(Orientation)을 수학적으로 표현할 수 있어야 한다. 일반적으로 위치는 점(Point)으로 표현되고 방향은 벡터(Vector)로 표현된다(물론 점을 점 벡터(Point vector)로 표현하기도 한다). 그러므로 점과 벡터에 대한 의미를 이해하고 구별하는 것이 필요하다. 이러한 위치와 방향을 수학적으로 표현하기 위하여 좌표계를 사용한다. 좌표계에는 직교 좌표계(Cartesian coordinate system), 극 좌표계(Polar coordinates system), 구면 좌표계(Spherical coordinates system)등이 있다, 일반적으로 이해하기 쉬운 직교 좌표계를 사용한다.

- 직교 좌표계(Cartesian Coordinate System)

  직교 좌표계는 좌표계의 축(Axis)들이 서로 직교하는 좌표계이다. 직교 좌표계는 축들의 교점(원점)과 축의 방향으로 정의할 수 있다. 직교 좌표계에서 위치와 방향을 표현하기 위하여 벡터(Vector)를 사용한다. 2차원 직교 좌표계(평면)는 실수의 집합 R과 R의 곱집합 R^2으로 표현할 수 있다. 그러므로 2차원 직교 좌표계의 한 점은 순서쌍

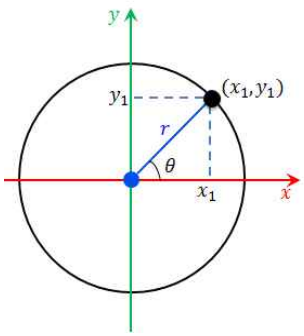
(x, y)로 표현할 수 있다. 3차원 직교 좌표계(공간)는 곱집합 R^3으로 표현할 수 있다. 그러므로 3차원 직교 좌표계의 한 점은 순서쌍 (x, y, z)로 표현할 수 있다.

일반적으로 3차원 그래픽에서 왼손 좌표계(Left hand coordinate) 또는 오른손 좌표계(Right hand coordinate)를 사용한다. 3차원 직교 좌표계에서 왼손 좌표계와 오른손 좌표계의 차이는 축의 방향이 다르다는 것이다. 다음 그림은 왼손 좌표계와 오른손 좌표계를 보여주고 있다. 왼손좌표계는 왼손의 엄지, 검지, 그리고 중지를 서로 직각이 되도록 펼쳤을 때, 기본적으로 엄지가 x-축, 검지가 y-축, 그리고 중지가 z-축이 되는 좌표계이다. 그러나 엄지가 y-축, 검지가 z-축, 그리고 중지가 x-축이 되어도 왼손 좌표계이다. 즉, 순서가 x-축, y-축, z-축이라면 시작이 어디든 상관없다. 즉, 왼손 좌표계에서 중지가 y-축이라면 엄지가 z-축, 검지가 x-축이어야 한다. Direct3D에서는 왼손 좌표계를 사용한다. 왼손좌표계와 오른손좌표계의 차이는 z-좌표축의 방향이 다르다는 것이다. 이 과정(Direct3D)에서는 왼손 좌표계를 사용한다. 다른 그래픽 라이브러리 또는 도구(Tool)에서는 오른손 좌표계를 사용하기도 한다.



- 극 좌표계(Polar Coordinates System)

  극 좌표계는 2차원 좌표계이다. 2차원 직교 좌표계의 한 점 (x, y)는 좌표계의 원점에서 반지름이 r이고 x-축과의 각도가 θ인 원 위의 점으로 표현할 수 있다. 2차원 직교 좌표계의 한 점 (x, y)는 (r, θ)로 표현할 수 있다. (r, θ)로 표현하는 좌표계를 극 좌표(Polar coordinate)라고 한다.



극 좌표 (r, θ)는 직교 좌표 (x, y)로 다음과 같이 변환할 수 있다.

$$(r, \theta) \rightarrow (x, y)$$

$$x = rcos(\theta)$$
$$y = rsin(\theta)$$

직교 좌표 (x, y)는 극 좌표 (r, θ)로 다음과 같이 변환할 수 있다.

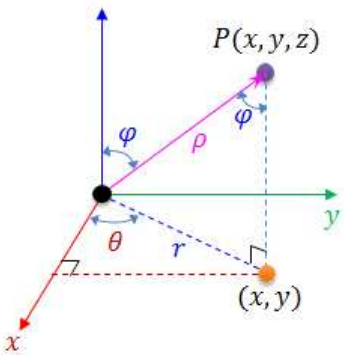
$$(x, y) \rightarrow (r, \theta)$$
$$r = \sqrt{x^2 + y^2}$$
$$\theta = \cos^{-1}\left(\frac{x}{r}\right)$$
$$\frac{y}{x} = \tan(\theta)$$
$$\theta = \tan^{-1}\left(\frac{y}{x}\right)$$

- 구면 좌표계(Spherical Coordinates System)

  구면 좌표계는 3차원 좌표계이다. 3차원 직교 좌표계의 한 점 (x, y, z)은 좌표계의 원점에서 반지름이 ρ이고 x-축과의 각도가 $\theta(0 \le \theta \le 2\pi)$, z-축과의 각도가 $\phi(0 \le \phi \le \pi)$인 구 표면의 점으로 표현할 수 있다. 3차원 직교 좌표계의 한 점 (x, y, z)은 (ρ, θ, ϕ)로 표현할 수 있다. (ρ, θ, ϕ)로 표현하는 좌표계를 구면 좌표(Spherical Coordinate)라고 한다.

구면 좌표 (ρ, θ, ϕ)는 직교 좌표 (x, y, z)로 다음과 같이 변환할 수 있다.

$$(\rho, \theta, \phi) \rightarrow (x, y, z)$$
$$x = \rho \sin(\phi)\cos(\theta)$$
$$y = \rho \sin(\phi)\sin(\theta)$$
$$z = \rho \cos(\phi)$$



직교 좌표 (x, y, z)는 구면 좌표 (ρ, θ, ϕ)로 다음과 같이 변환할 수 있다.

$$(x, y, z) \rightarrow (\rho, \theta, \phi)$$

$$\rho = \sqrt{x^2 + y^2 + z^2}$$

$$\sin(\phi) = \frac{r}{\rho}$$

$$r = \sqrt{x^2 + y^2}$$

$$= \sqrt{\{\rho \sin(\phi)\cos(\theta)\}^2 + \{\rho \sin(\phi)\sin(\theta)\}^2}$$

$$= \sqrt{\{\rho \sin(\phi)\}^2\{\cos^2(\phi) + \sin^2(\theta)\}} = \rho \sin(\phi)$$

$$\cos(\theta) = \frac{x}{r}$$

$$x = r\cos(\theta)$$

$$r = \rho \sin(\phi)$$

$$\frac{r}{z} = \frac{\rho \sin(\phi)}{\rho \cos(\phi)} = \tan(\phi) \rightarrow \phi = \arctan2(r, z)$$

$$\frac{y}{x} = \frac{\sin(\theta)}{\cos(\theta)} = \tan(\theta) \rightarrow \theta = \arctan2(y, x)$$

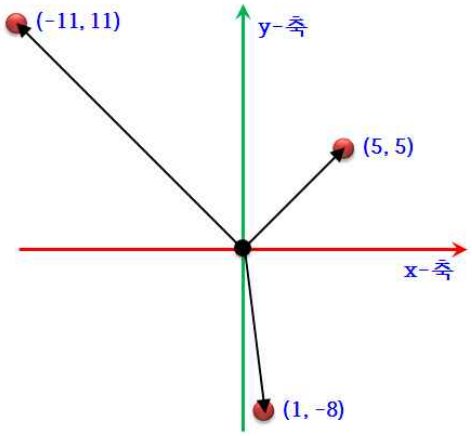
③ 벡터(Vector)

벡터는 어떤 공간(좌표계)에서 방향과 크기를 함께 표현하는 개념이다. 벡터는 점(위치)을 표현하기 위하여 사용할 수 있다. 벡터가 점(위치)을 표현하면 위치 벡터 또는 점 벡터라고 하며, 벡터가 방향을 표현하면 방향 벡터라고 한다. 두 가지 경우 모두 단순히 줄여서 벡터라고 표현하기도 한다(문맥 또는 의미에 따라 구별해야 한다). 직교 좌표계의 원점에 어떤 벡터를 더한 결과는 점(Point)이 된다. 수학적으로 점과 벡터를 표현하기 위하여 순서쌍(Ordered Pair)을 사용한다. 순서쌍의 원소의 개수를 벡터의 차원(Dimension)이라고 한다. $m-$차원 벡터는 $\boldsymbol{a} = (a_1, a_2, ..., a_{m-1}, a_m)$ 형태의 순서쌍으로 표현하고 $(1 \leq i \leq m)$일 때 a_i를 벡터의 원소(Element) 또는 요소(Component)라고 한다. 일반적으로 2-차원 벡터는 (x, y)의 순서쌍 형태로 표현하고, 3-차원 벡터는 (x, y, z)의 순서쌍 형태로 표현한다.

어떤 순서쌍이 벡터를 표현하는 것인지 또는 점을 표현하는 것인지는 문맥(Context)에 따라 결정된다. 순서쌍이 점을 표현하면 점 벡터(Point vector)라고 하고, 방향을 표현하면 방향 벡터 또는 벡터라고 한다. 즉, 벡터는 순서쌍에 저장된 값을 어떻게 해석하는 가에 따라 공간상의 한 점(위치)인지 또는 방향과 크기를 나타내는 것인지가 결정된다. 공간상의 한 점은 항상 벡터로 표현할 수 있다. 공간상의 한 점은 원점에서 그 점으로의 방향과 원점에서 그 점까지의 거리를 크기로 가지는 벡터로 표현할 수 있다. 보통 점(위치)을 벡터로 표현할 때 대문자를 사용하고 방향을 벡터로 표현할 때 소문자를 사용한다.

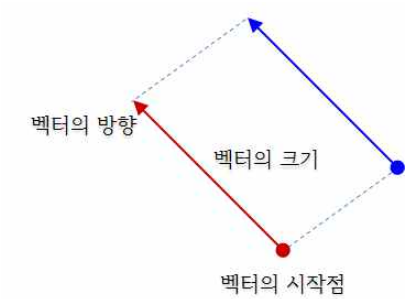
벡터를 도식화하기 위하여 다음 그림과 같이 벡터의 시작점에서 벡터가 표현하려는 방향으로 벡터가 표현하려는 크기를 갖는 화살표를 그린다.

다음 그림은 2-차원 직교 좌표계의 점들을 벡터로 나타낸 예이다.



수학적으로 3-차원 벡터는 (x, y, z)의 순서쌍 형태로 표현할 수 있다. 영(Zero) 벡터는 크기가 0이고 방향이 없는 벡터이며 $(0, 0, 0)$의 순서쌍으로 표현한다. 영 벡터는 $\mathbf{0}$으로 표현한다. 두 벡터의 크기와 방향이 서로 같으면 두 벡터는 같다고 정의한다. 같은 두 벡터는 평행 이동으로 겹쳐질 수 있다. 즉, 시작점의 위치에 상관없이 방향과 크기가 같은 모든 벡터는 같다. 3-차원 벡터 (x_1, y_1, z_1)과 벡터 (x_2, y_2, z_2)가 같으면 다음이 성립한다.

$$(x_1 = x_2) \wedge (y_1 = y_2) \wedge (z_1 = z_2)$$



벡터는 기본적으로 상대적인 개념을 표현하고 있음에 주의하라. 시작점의 위치에 상관없이 방향과 크기가 같으면 두 개의 벡터는 같으므로 벡터는 힘(Power), 속도(Velocity), 방향(Direction) 등의 상대적인 개념을 표현하기 위하여 사용될 수 있다.

점(위치) A를 벡터(순서쌍)로 표현하는 것은 원점 O에서 점 A로의 방향과 원점 O에서 점 A까지 거리를 크기로 갖는 벡터(순서쌍)로 표현하는 것이다. 일반적으로 이 벡터는 $\overrightarrow{OA}$로 표현하거나 간단하게 $\overrightarrow{A}$ 또는 A로 표현한다. 점 A에서 점 B까지의 벡터는 $\overrightarrow{AB}$로 표현한다. 방향을 나타내는 벡터는 벡터의 시작점을 사용하지 않고 n과 v같이 소문자를 사용하여 표현한다.

다음은 3-차원 벡터의 순서쌍 (x, y, z)를 표현하는 구조체이다.

```
struct Vector
{
    float x;
    float y;
    float z;
};
```

④ 벡터의 기본 연산(Basic Vector Operation)

벡터에 대한 기본적인 연산은 벡터의 합(덧셈, Addition), 벡터의 스칼라 곱(Scalar Multiplication), 벡터의 차(뺄셈, Subtraction)이다. 벡터의 합(덧셈), 벡터의 차(뺄셈), 벡터의 스칼라 곱 연산의 결과는 벡터가 된다. 즉, 두 개의 벡터에 대하여 벡터의 합(덧셈)과 벡터의 차(뺄셈) 연산을 하면 새로운 벡터가 생성된다. 하나의 벡터에 대하여 스칼라 곱 연산을 하면 새로운 벡터를 생성한다. 벡터의 합(덧셈), 벡터의 차(뺄셈), 벡터의 스칼라 곱 연산은 요소별 연산(Per-component operation)이다. 즉, 요소별 연산이란 벡터의 연산을 할 때 같은 요소(원소)끼리 연산을 한다는 것을 의미한다. 벡터의 합(덧셈)과 벡터의 차(뺄셈) 연산을 하려면 두 벡터의 차원(원소의 개수)이 같아야 한다.

- 벡터의 덧셈, 뺄셈, 스칼라 곱의 대수적(Algebraic) 표현

  $m-$차원 벡터 $\boldsymbol{a}=(a_1,...,a_{m-1},a_m)$와 $\boldsymbol{b}=(b_1,...,b_{m-1},b_m)$의 덧셈의 결과는 $(1 \le i \le m)$ 일 때, 다음을 만족하는 $m-$차원 벡터 $\boldsymbol{v}$이다.

  $$v_i = a_i + b_i$$

  $m-$차원 벡터 $\boldsymbol{a}=(a_1,...,a_{m-1},a_m)$와 $\boldsymbol{b}=(b_1,...,b_{m-1},b_m)$의 뺄셈의 결과는 $(1 \le i \le m)$ 일 때, 다음을 만족하는 $m-$차원 벡터 $\boldsymbol{v}$이다.

  $$v_i = a_i - b_i$$

  $m-$차원 벡터 $\boldsymbol{a}=(a_1,...,a_{m-1},a_m)$와 실수(스칼라) c의 곱 $c\boldsymbol{a}$는 다음을 만족하는 $m-$차원 벡터 $\boldsymbol{v}$이다.

$$v_i = c\,a_i$$

3-차원 벡터 $\boldsymbol{a} = (x_1, y_1, z_1)$와 벡터 $\boldsymbol{b} = (x_2, y_2, z_2)$에 대한 벡터의 덧셈과 뺄셈은 다음과 같다.

$$\boldsymbol{a} + \boldsymbol{b} = (x_1, y_1, z_1) + (x_2, y_2, z_2) = (x_1 + x_2, y_1 + y_2, z_1 + z_2)$$
$$\boldsymbol{a} - \boldsymbol{b} = (x_1, y_1, z_1) - (x_2, y_2, z_2) = (x_1 - x_2, y_1 - y_2, z_1 - z_2)$$

3-차원 벡터 $\boldsymbol{a} = (x, y, z)$와 실수(스칼라) c의 곱 $c\boldsymbol{a}$는 다음과 같다.

$$c\boldsymbol{a} = c(x, y, z) = (cx, cy, cz)$$

$\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$가 벡터일 때 벡터의 덧셈에 대하여 다음의 법칙이 성립한다.

- 교환 법칙(Commutative): $\boldsymbol{a} + \boldsymbol{b} = \boldsymbol{b} + \boldsymbol{a}$
- 결합 법칙(Associative): $(\boldsymbol{a} + \boldsymbol{b}) + \boldsymbol{c} = \boldsymbol{a} + (\boldsymbol{b} + \boldsymbol{c})$
- 덧셈의 항등원: $\boldsymbol{a} + \boldsymbol{0} = \boldsymbol{0} + \boldsymbol{a} = \boldsymbol{a}$

벡터 $\boldsymbol{v}$, $\boldsymbol{w}$가 벡터이고 a, b가 실수일 때, 벡터의 스칼라 곱과 덧셈에 대하여 다음의 법칙이 성립한다.

- 모든 벡터 $\boldsymbol{v}$에 대하여 $\boldsymbol{v} + (-\boldsymbol{v}) = \boldsymbol{0}$인 벡터 $-\boldsymbol{v}$가 존재한다.
- $(ab)\boldsymbol{v} = a(b\boldsymbol{v})$
- $(a+b)\boldsymbol{v} = a\boldsymbol{v} + b\boldsymbol{v}$
- $a(\boldsymbol{v} + \boldsymbol{w}) = a\boldsymbol{v} + a\boldsymbol{w}$
- $1\boldsymbol{v} = \boldsymbol{v}$

다음은 3-차원 벡터의 합(덧셈), 벡터의 차(뺄셈), 스칼라 곱의 예이다.

$\boldsymbol{a} = (1, 2, 3)$ $\boldsymbol{b} = (2, 1, 4)$

$\boldsymbol{a} + \boldsymbol{b} = (1, 2, 3) + (2, 1, 4) = (3, 3, 7)$

$\boldsymbol{a} - \boldsymbol{b} = (1, 2, 3) - (2, 1, 4) = (-1, 1, -1)$

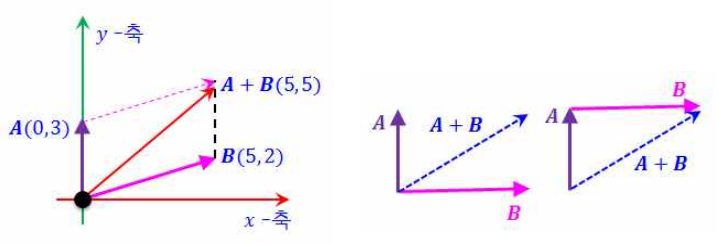
$2\boldsymbol{a} = 2(1, 2, 3) = (2, 4, 6)$

- 벡터의 덧셈, 뺄셈, 스칼라 곱의 기하적(Geometric) 의미

다음은 벡터의 덧셈, 뺄셈, 스칼라 곱의 기하적 의미를 설명하는 예이다.

- 벡터의 합(덧셈)

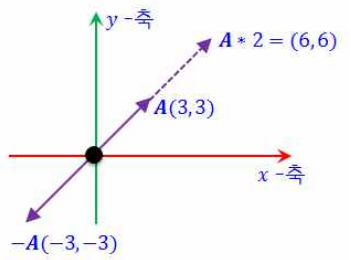
  두 벡터 $\boldsymbol{A} = (0, 3)$와 $\boldsymbol{B} = (5, 2)$의 시작점을 같게 했을 때, 두 벡터의 길이를 두 변으로 하는 평행사변형의 대각선(빨간색)이 두 벡터의 합 $\boldsymbol{A} + \boldsymbol{B} = (5, 5)$이 된다. 벡터는 평행이동에 의해 겹쳐질 수 있으면 같은 벡터임에 유의하라.



- 벡터와 실수(스칼라)의 곱

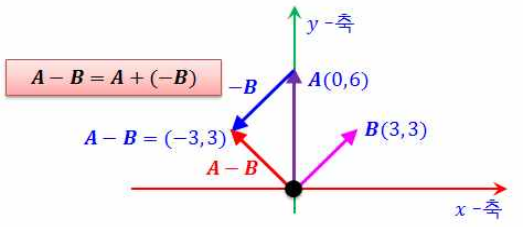
  벡터 $\boldsymbol{A}=(3,3)$와 실수 2의 곱은 $\boldsymbol{A}=2(3,3)=(6,6)$이 된다. 벡터 $\boldsymbol{A}=(3,3)$와 실수 -1의 곱은 $\boldsymbol{A}=-1(3,3)=-(3,3)=(-3,-3)$이 된다.

  벡터 $\boldsymbol{a}$에 양의 실수 c를 곱하면 벡터 $c\boldsymbol{a}$의 길이가 c배가 되고 방향이 바뀌지 않는다. 벡터 $\boldsymbol{a}$에 음의 실수 c를 곱하면 벡터 $c\boldsymbol{a}$의 길이가 $|c|$배가 되고 벡터의 방향이 반대(180°)가 된다.



- 벡터의 차(뺄셈)

  두 벡터 $\boldsymbol{A}=(0,6)$와 $\boldsymbol{B}=(3,3)$의 뺄셈은 $\boldsymbol{A}-\boldsymbol{B}=(-3,3)$이다(빨간색 벡터). 이것은 벡터 $\boldsymbol{A}$와 $\boldsymbol{B}$의 시작점을 원점으로 같게 했을 때, 점 $\boldsymbol{B}$에서 점 $\boldsymbol{A}$로 가는 벡터 $\overrightarrow{\boldsymbol{BA}}$와 같다. 벡터는 평행이동에 의해 겹쳐질 수 있으면 같은 벡터이므로, 벡터 $(\boldsymbol{A}-\boldsymbol{B})$를 평행이동하면 벡터 $\overrightarrow{\boldsymbol{BA}}$이다. 또한 $\boldsymbol{A}-\boldsymbol{B}=\boldsymbol{A}+(-\boldsymbol{B})$임을 보여준다.

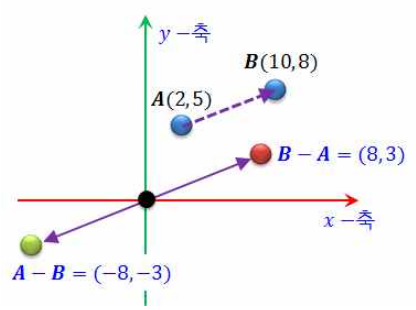


  벡터의 뺄셈은 교환법칙이 성립하지 않는다. 즉 두 벡터 $\boldsymbol{A}$와 $\boldsymbol{B}$에 대하여 $\boldsymbol{A}-\boldsymbol{B}\neq\boldsymbol{B}-\boldsymbol{A}$이다.

  다음은 $(\boldsymbol{A}-\boldsymbol{B})$와 $(\boldsymbol{B}-\boldsymbol{A})$는 같지 않고, 크기는 같지만 방향은 서로 반대임을 보

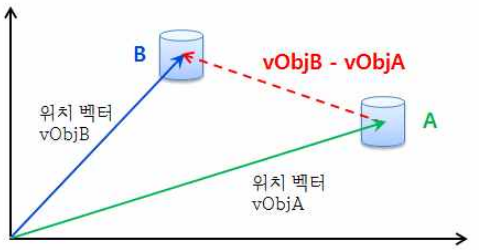
여주는 예이다.

$$A - B = -(B - A)$$



벡터의 뺄셈은 두 점이 주어질 때 두 점사이의 방향 벡터를 구하기 위하여 사용할 수 있다. 객체 A와 B의 위치(벡터) $\boldsymbol{A}$와 $\boldsymbol{B}$가 주어질 때 객체 A에서 B로의 방향은 벡터 $(\boldsymbol{B}-\boldsymbol{A})$이고 객체 B에서 A로의 방향은 벡터 $(\boldsymbol{A}-\boldsymbol{B})$이다.

다음은 점 A에서 점 B로 객체를 이동할 때, 이동 방향은 $(\boldsymbol{B}-\boldsymbol{A})$ 벡터임을 보여준다.



- 3-차원 벡터의 덧셈, 뺄셈, 스칼라 곱의 함수 표현

  3-차원 벡터의 덧셈과 뺄셈을 위한 함수는 다음과 같이 나타낼 수 있다.

```
Vector Vector3Add(Vector A, Vector B)
{
    Vector C;
    C.x = A.x + B.x;
    C.y = A.y + B.y;
    C.z = A.z + B.z;

    return(C);
}

Vector Vector3Subtract(Vector A, Vector B)
{
    Vector C;
    C.x = A.x - B.x;
    C.y = A.y - B.y;
    C.z = A.z - B.z;
```

```
    return(C);
}
```

다음은 3-차원 벡터와 실수(스칼라)의 곱을 표현한 함수이다.

```
Vector Vector3ScalarMultiply(Vector A, float scalar)
{
    Vector C;
    C.x = A.x * scalar;
    C.y = A.y * scalar;
    C.z = A.z * scalar;

    return(C);
}
```

- 벡터의 크기(Magnitude, 길이: Length)

  벡터 $\boldsymbol{a}$의 크기(길이)는 $|\boldsymbol{a}|$로 표기한다. m-차원 벡터 $\boldsymbol{a}=(a_1,\ldots,a_{m-1},a_m)$의 크기 $|\boldsymbol{a}|$는 다음과 같이 정의한다.

$$|\boldsymbol{a}|=\sqrt[2]{a_1^2+a_2^2+\cdots+a_m^2}$$

  3-차원 벡터 $\boldsymbol{a}=(x,y,z)$의 크기 $|\boldsymbol{a}|$는 다음과 같다. 이것은 원점에서 점 $\boldsymbol{A}$까지의 거리를 나타낸다.

$$|\boldsymbol{a}|=\sqrt[2]{x^2+y^2+z^2}$$

  벡터 $\boldsymbol{a}$의 크기에 대하여 다음의 성질이 성립한다.

  - $|\boldsymbol{a}|\geq 0$
  - $(|\boldsymbol{a}|=0)\Leftrightarrow(\boldsymbol{a}=\boldsymbol{0})$
  - $|c\boldsymbol{a}|=|c||\boldsymbol{a}|$
  - $|\boldsymbol{a}+\boldsymbol{b}|\leq|\boldsymbol{a}|+|\boldsymbol{b}|$

  다음은 3-차원 벡터의 크기를 구하는 함수이다.

```
float Vector3Length(Vector A)
{
    return(sqrt(A.x*A.x + A.y*A.y + A.z*A.z));
}
```

  3-차원 벡터 $\boldsymbol{a}=(x,y,z)$와 실수 c의 곱 $c\boldsymbol{a}$의 크기 $|c\boldsymbol{a}|$는 벡터 $\boldsymbol{a}$의 크기 $|\boldsymbol{a}|$의 $|c|$배가 된다.

$$|c\boldsymbol{a}|=\sqrt[2]{(cx)^2+(cy)^2+(cz)^2}=\sqrt[2]{c^2(x^2+y^2+z^2)}=|c||\boldsymbol{a}|$$

- 단위 벡터(Unit Vector)

단위 벡터는 벡터의 크기(길이)가 1인 벡터이다. 단위 벡터가 아닌 벡터를 단위 벡터로 변환하는 것을 벡터의 정규화(Normalization)라고 한다. 즉, 벡터를 정규화하면 벡터의 크기가 1이 된다. 벡터의 정규화는 벡터의 모든 요소를 벡터의 크기로 나누는(벡터의 크기의 역수로 스칼라 곱을 하는) 연산이다.

벡터 $\boldsymbol{a}$를 정규화한 단위 벡터는 $\hat{\boldsymbol{a}}$로 표기한다.

$$\hat{\boldsymbol{a}} = \frac{1}{|\boldsymbol{a}|}\boldsymbol{a}$$

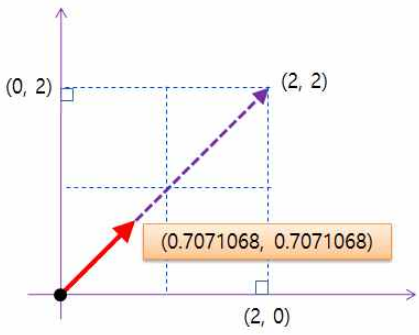
3-차원 벡터 $\boldsymbol{a} = (x, y, z)$의 정규화는 다음과 같이 나타낼 수 있다.

$$\hat{\boldsymbol{a}} = \frac{1}{|\boldsymbol{a}|}(x, y, z) = \left(\frac{x}{|\boldsymbol{a}|}, \frac{y}{|\boldsymbol{a}|}, \frac{z}{|\boldsymbol{a}|}\right) = \left(\frac{x}{\sqrt{x^2+y^2+z^2}}, \frac{y}{\sqrt{x^2+y^2+z^2}}, \frac{z}{\sqrt{x^2+y^2+z^2}}\right)$$

다음은 3-차원 벡터의 정규화를 나타내는 함수이다.

```
Vector Vector3Normalize(Vector A)
{
    float length = Vector3Length(A);
    A.x = A.x / length;
    A.y = A.y / length;
    A.z = A.z / length;

    return(A);
}
```

다음 그림은 벡터의 정규화를 나타낸다. 벡터의 정규화는 벡터와 벡터의 크기의 역수를 스칼라 곱을 하는 것이다. 벡터의 크기는 양수이므로 벡터의 정규화는 벡터의 방향을 바꾸지 않고 벡터의 크기만을 1로 바꾸는 연산이다.



모든 단위 벡터는 크기가 1이므로 방향만 서로 다르다. 즉, 단위 벡터는 벡터의 방향 정보만 가진 벡터라고 할 수 있다. 그러므로 단위 벡터는 방향을 표현하기 위하여 사용한다. 방향을 표현하는 예는 회전축, 다각형(면)의 방향, 조명의 방향, 물체의 이동 방향 등이다. 방향을 표현하는 벡터는 항상 단위 벡터를 사용한다고 생각하자.

- 물체(객체)의 이동(Movement)

  단위 벡터는 물체가 향하고 있는 방향 또는 물체의 이동 방향을 표현하기 위해 사용할 수 있다. 물체의 이동 방향(벡터)과 이동 거리(양의 실수)가 주어지면 물체를 이동할 수 있다(위치를 변경할 수 있다). 물체를 어떤 방향으로 이동할 때, 방향을 나타내는 단위 벡터에 이동 거리를 곱(벡터와 스칼라의 곱)하여 얻어진 벡터는 이동 벡터(Movement vector, Displacement vector)라고 한다. 이동 벡터를 이동하기 전의 물체 위치를 나타내는 벡터에 더하면(벡터의 덧셈) 물체를 이동한 결과 위치를 계산할 수 있다.

  다음은 벡터의 스칼라 곱과 벡터의 덧셈을 응용하는 예이다. 2차원 평면에서 점 $A(22, 4)$에서 점 $D(20.57, 21.72)$까지 이동 과정을 벡터를 사용하여 설명하고 있다.

  먼저, 점 $A(22, 4)$에서 위쪽(y-축) 방향($\boldsymbol{L1}$, 단위 벡터 (0, 1))으로 5만큼의 거리를 이동한다. 벡터 (0, 1)이 단위 벡터이므로 이동 거리는 정확히 5이다.

$$\boldsymbol{B} = \boldsymbol{A} + 5\boldsymbol{L1} = (22, 4) + 5(0, 1) = (22, 9)$$

  다음에 왼쪽 대각선 방향(벡터 (-1, 1))으로 10만큼의 거리를 이동한다. 벡터 (-1, 1)는 단위 벡터가 아님에 주의하라. 벡터 (-1, 1)의 크기는 $\sqrt{2} \approx 1.414$이므로 정규화하면 단위 벡터 $\boldsymbol{L2}$는 다음과 같다.

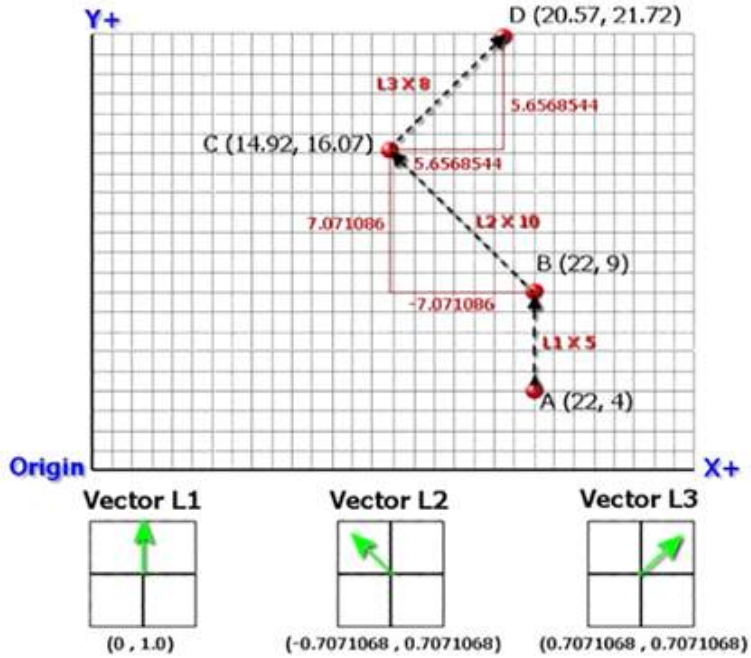
$$\boldsymbol{L2} = \frac{1}{\sqrt{2}}(-1, 1) = (\frac{-1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$$

$$\boldsymbol{C} = \boldsymbol{B} + 10\boldsymbol{L2} = (22, 9) + 10(\frac{-1}{\sqrt{2}}, \frac{1}{\sqrt{2}}) = (14.92, 16.07)$$

  그리고 오른쪽 대각선 방향(벡터 (1, 1))으로 8만큼의 거리를 이동한다. 벡터 (1, 1)는 단위 벡터가 아님에 주의하라. 벡터 (1, 1)를 정규화하면 단위 벡터 $\boldsymbol{L3}$는 다음과 같다.

$$\boldsymbol{L3} = \frac{1}{\sqrt{2}}(1, 1) = (\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$$

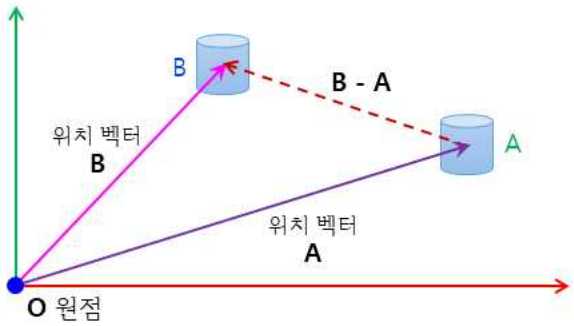
$$\boldsymbol{D} = \boldsymbol{C} + 8\boldsymbol{L3} = (14.92, 16.07) + 8(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}) = (20.57, 21.72)$$



물체의 이동 방향과 이동 거리가 주어질 때 최종 위치를 계산하기 위하여 필요한 벡터의 연산은 벡터의 정규화, 벡터의 스칼라 곱, 벡터의 덧셈이다.

다음은 벡터의 뺄셈을 활용하여 객체 A가 객체 B를 향하도록 fDistance 만큼 이동하여 위치를 변경하는 함수이다. 객체 A가 객체 B를 향하도록 fDistance 만큼 이동했을 때, 객체 A(위치 벡터 $\boldsymbol{A}$)의 새로운 위치 A'(위치 벡터 $\boldsymbol{A}'$)는 수학적으로 다음과 같다. 즉, 객체 A가 객체 B를 향하는 방향을 구하려면, 벡터 $\boldsymbol{B}$에서 벡터 $\boldsymbol{A}$를 뺄셈의 결과 $(\boldsymbol{B}-\boldsymbol{A})$를 정규화한다. 그리고 fDistance 만큼 이동하려면 벡터 $(\boldsymbol{B}-\boldsymbol{A})$의 단위 벡터에 fDistance를 곱하여 벡터 $\boldsymbol{A}$에 더한다.

$$\boldsymbol{A}' = \boldsymbol{A} + fDistance\left(\frac{\boldsymbol{B}-\boldsymbol{A}}{|\boldsymbol{B}-\boldsymbol{A}|}\right)$$




```
Vector MoveObjectFromAToB(Vector A, Vector B, float fDistance)
{
    Vector direction = Vector3Subtract(B, A);
```

```
    Vector normal = Vector3Normalize(direction);
    Vector movement = Vector3ScalarMultiply(normal, fDistance);
    Vector result = Vector3Add(A, movement);

    return(result);
}
```

일반적으로 게임 프로그램에서 객체를 이동할 거리 d가 직접 주어지지 않고 객체의 이동 속력(Speed: ㎧)이 주어진다. 객체의 이동 속력 s가 주어지고 이동 시간 $\triangle t$가 주어지면 이동 거리는 $(s \triangle t)$이다. 게임 프로그램의 경우 $\triangle t$는 프레임 레이트(Frame rate)가 될 수 있다. 즉, 매 프레임 마다 $\triangle t$가 주어지면 한 프레임 시간 동안 객체를 $(s \triangle t)$씩 만큼 천천히 이동하면 된다(그래야 객체가 목표 지점을 이동하는 과정을 화면으로 볼 수 있게 된다).

게임 프로그램에서 게임 객체를 이동하는 상황은 크게 다음과 같다.

- 이동할 객체 A의 위치, 이동할 방향 $\boldsymbol{n}$, 이동 거리 d가 주어지는 경우
  방향 벡터 $\boldsymbol{n}$과 이동할 거리 d를 곱하여 이동할 객체 A의 위치 벡터 $\boldsymbol{A}$에 더한다.
- 두 객체의 위치가 주어지는 경우
  객체 A와 객체 B의 위치가 주어지면, 벡터 $\boldsymbol{B}$에서 벡터 $\boldsymbol{A}$를 뺄셈의 결과 $(\boldsymbol{B}-\boldsymbol{A})$를 정규화하면 이동할 방향 $\boldsymbol{n}$을 구할 수 있다.

정리하면, 벡터의 덧셈 연산은 객체를 이동(위치의 변화)하기 위하여, 벡터의 뺄셈은 두 점 사이의 방향을 구하기 위하여, 벡터의 스칼라 곱 연산은 이동 벡터를 구하기 위하여 사용할 수 있다는 것이다.

⑤ 벡터의 외적(Cross Product) 연산

두 개의 3-차원 벡터에 대한 외적 연산은 두 개의 3-차원 벡터에 모두 수직인 새로운 3-차원 벡터를 구하는 연산이다. 3-차원 벡터 $\boldsymbol{a}=(x_1, y_1, z_1)$와 $\boldsymbol{b}=(x_2, y_2, z_2)$의 외적은 다음과 같이 정의한다.

$$\boldsymbol{a}\times\boldsymbol{b}=(y_1z_2, z_1x_2, x_1y_2)-(z_1y_2, x_1z_2, y_1x_2)=(y_1z_2-z_1y_2, z_1x_2-x_1z_2, x_1y_2-y_1x_2)$$

다음은 3-차원 벡터의 외적의 예이다.

$$\boldsymbol{a}=(0,1,0),\ \boldsymbol{b}=(1,0,0)$$
$$\boldsymbol{a}\times\boldsymbol{b}=(1*0-0*1, 0*1-0*0, 0*0-1*1)=(0,0,-1)$$
$$\boldsymbol{b}\times\boldsymbol{a}=(0*0-0*1, 0*0-1*0, 1*1-0*0)=(0,0,1)$$

3-차원 벡터 $\boldsymbol{a}$와 $\boldsymbol{b}$의 외적은 다음과 같이 표현할 수 있다. θ는 3-차원 벡터 $\boldsymbol{a}$와 $\boldsymbol{b}$가 이루는 각도이고, 3-차원 벡터 $\boldsymbol{n}$은 3-차원 벡터 $\boldsymbol{a}$와 $\boldsymbol{b}$를 포함하는 평면의 법선 벡터(평면

에 수직인 벡터)이다.

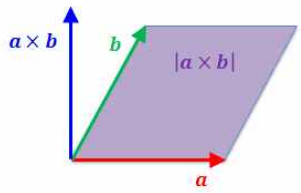
$$\boldsymbol{a} \times \boldsymbol{b} = |\boldsymbol{a}||\boldsymbol{b}| \sin(\theta) \boldsymbol{n}$$

3-차원 벡터의 외적의 정의에 따라 3-차원 벡터의 외적은 교환법칙이 성립하지 않는다. 3-차원 벡터 $\boldsymbol{a}$와 $\boldsymbol{b}$의 외적 $\boldsymbol{a} \times \boldsymbol{b}$는 $\boldsymbol{b} \times \boldsymbol{a}$와 벡터의 크기는 같지만 방향이 반대인 3-차원 벡터이다.

$$\boldsymbol{a} \times \boldsymbol{b} = -(\boldsymbol{b} \times \boldsymbol{a})$$

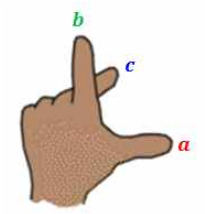
3-차원 벡터 $\boldsymbol{a}$와 $\boldsymbol{b}$의 외적 $\boldsymbol{a} \times \boldsymbol{b}$의 크기는 두 벡터가 변인 평행사변형의 면적과 같다.

$$|\boldsymbol{a} \times \boldsymbol{b}| = |\boldsymbol{a}||\boldsymbol{b}| \sin(\theta)$$

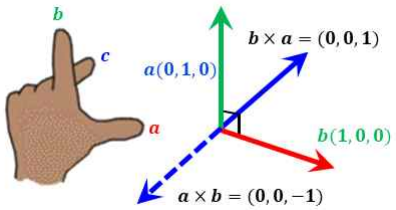


평행한 두 3-차원 벡터의 외적의 결과는 영 벡터(0)이다(방향이 같거나 방향이 반대인 두 3-차원 벡터가 이루는 각도는 0도 또는 180도이므로 $\sin(\theta)$가 0이다).

왼손 좌표계에서 3-차원 벡터의 외적의 결과를 표현(확인)할 수 있는 쉬운 방법 중 하나는 왼손 법칙(Left hand thumb rule)을 사용하는 것이다. 왼손을 다음 그림과 같이 왼손의 엄지, 검지, 그리고 중지를 서로 직각이 되도록 펼쳤을 때, 엄지를 벡터 $\boldsymbol{a}$, 검지가 벡터 $\boldsymbol{b}$라고 하면 중지가 두 벡터의 외적 $\boldsymbol{a} \times \boldsymbol{b} = \boldsymbol{c}$이다. 이때 $\boldsymbol{a}$와 $\boldsymbol{b}$는 서로 직교할 필요는 없지만 두 벡터의 외적인 $\boldsymbol{c}$는 $\boldsymbol{a}$와 $\boldsymbol{b}$에 서로 수직이다. 이것은 3-차원 벡터 $\boldsymbol{a}$와 $\boldsymbol{b}$를 포함하는 평면에 수직인 3-차원 벡터가 $\boldsymbol{c}$임을 나타낸다.



다음 그림은 왼손 좌표계에서 x-축과 y-축의 외적은 z-축임을 보여준다. 그리고 y-축과 x-축의 외적은 -z-축이다. 이것은 앞에서 계산한 외적의 결과와 같다. 외적의 왼손 법칙에 대하여 익숙해지기를 바란다.



다음은 3-차원 벡터의 외적을 구하는 함수의 코드이다.

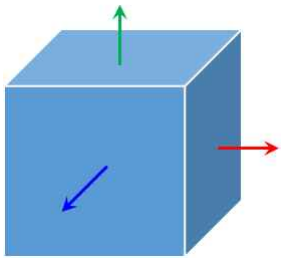
```
Vector Vector3CrossProduct(Vector A, Vector B)
{
    CVector C;
    C.x = (A.y*B.z) - (A.z*B.y);
    C.y = (A.z*B.x) - (A.x*B.z);
    C.z = (A.x*B.y) - (A.y*B.x);

    return(C);
}
```

다음은 $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$가 3-차원 벡터이고 k가 실수일 때 벡터의 외적 연산에 대하여 성립하는 성질이다.

- $\boldsymbol{a}\times(\boldsymbol{b}+\boldsymbol{c})=(\boldsymbol{a}\times\boldsymbol{b})+(\boldsymbol{a}\times\boldsymbol{c})$
- $(\boldsymbol{a}+\boldsymbol{b})\times\boldsymbol{c}=(\boldsymbol{a}\times\boldsymbol{c})+(\boldsymbol{b}\times\boldsymbol{c})$
- $k(\boldsymbol{b}\times\boldsymbol{c})=(k\boldsymbol{b})\times\boldsymbol{c}=\boldsymbol{b}\times(k\boldsymbol{c})$
- $\boldsymbol{a}\times\mathbf{0}=\mathbf{0}\times\boldsymbol{a}=\mathbf{0}$
- $\boldsymbol{a}\times\boldsymbol{b}=\mathbf{0}\Rightarrow(\boldsymbol{a}=\mathbf{0})\vee(\boldsymbol{b}=\mathbf{0})$
- $\boldsymbol{a}\times\boldsymbol{a}=\mathbf{0}$
- $\boldsymbol{a}\times\boldsymbol{b}=-(\boldsymbol{b}\times\boldsymbol{a})$
- $\boldsymbol{a}\times(\boldsymbol{b}\times\boldsymbol{c})+\boldsymbol{b}\times(\boldsymbol{c}\times\boldsymbol{a})+\boldsymbol{c}\times(\boldsymbol{a}\times\boldsymbol{b})=\mathbf{0}$
- $(\boldsymbol{a}\times\boldsymbol{b}=\boldsymbol{a}\times\boldsymbol{c})\not\Rightarrow(\boldsymbol{b}=\boldsymbol{c})$

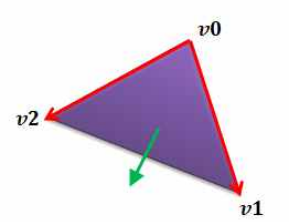
- 다각형의 법선 벡터

  일반적으로 다각형(면: Face)의 방향을 나타내는 3-차원 단위 벡터를 법선 벡터(Normal vector)라고 한다(접선 벡터(Tangent vector)에 수직인 벡터). 다각형의 법선 벡터는 다각형이 향하고 있는 방향을 나타내는 단위 벡터이다. 3-차원 벡터의 외적 연산을 통해 다각형 또는 면의 방향을 구할 수 있다. 면의 방향은 면이 바깥쪽으로 향하고 있는 방향을 의미한다. 어떤 면의 법선 벡터는 그 면과 수직이다(면 위에 있는 모든 선분과 서로 수직이다).



다음 그림과 같이 삼각형의 세 점을 나타내는 벡터 v_0, v_1, v_2가 주어질 때 이 삼각형의 법선 벡터 n은 다음과 같이 계산할 수 있다.

$$n = (v_1 - v_0) \times (v_2 - v_0)$$



메쉬(모델)을 구성하는 다각형(면)들이 주어지면 각 다각형은 삼각형으로 나눌 수 있으므로 삼각형의 법선 벡터를 구하면 면의 법선 벡터를 구할 수 있다.

다음은 다각형의 법선 벡터를 벡터의 외적 연산으로 계산하는 함수의 코드이다.

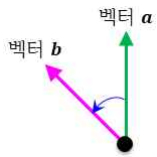
```
Vector GeneratePolygonNormal(Vector v0, Vector v1, Vector v2)
{
    Vector vEdge1 = Vector3Subtract(v1, v0);
    Vector vEdge3 = Vector3Subtract(v2, v0);
    Vector vNormal = Vector3CrossProduct(vEdge1, vEdge3);
    vNormal = Vector3Normalize(vNormal);

    return(vNormal);
}
```

- 회전축(Rotation Axis)

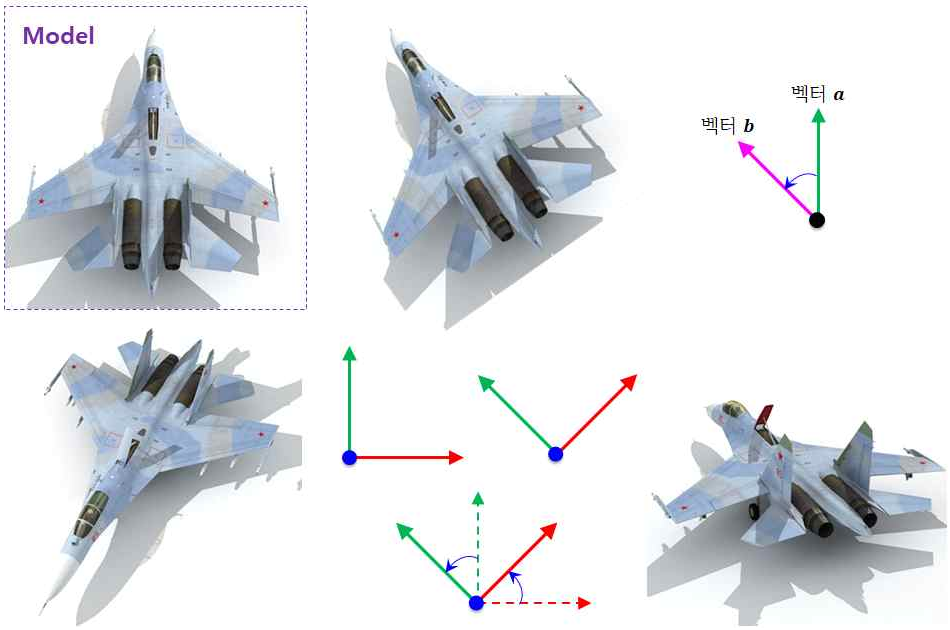
  3-차원 벡터의 외적 연산은 회전축을 구하기 위하여 사용할 수 있다. 다음 그림에서 3-차원 벡터 a를 회전하여 벡터 b와 일치하도록 하려면 회전의 중심(회전축)을 알아야 한다. 3-차원 벡터 a가 어떤 객체가 향하고 있는(예를 들어, 걸어가고 있는) 방향일 때 이 객체를 3-차원 벡터 b의 방향으로 향하도록 하려면 이 객체를 회전시켜야 한다. 이 때 회전의 중심(회전축)을 벡터 a와 벡터 b의 외적 연산으로 구할 수 있다.

  다음 그림에서(왼손 좌표계일 때) 외적 $b \times a$는 여러분을 향하는 벡터이고, $a \times b$는 그 반대 벡터이다. 외적 $b \times a$를 중심으로 벡터 a를 회전하여 벡터 b와 일치하도록 하

려면 반시계 방향으로 회전을 해야 한다. 외적 $\boldsymbol{a}\times\boldsymbol{b}$를 중심으로 벡터 $\boldsymbol{a}$를 회전하여 벡터 $\boldsymbol{b}$와 일치하도록 하려면 시계 방향으로 회전을 해야 한다. 일반적으로 회전의 방향은 회전축 벡터 방향(화살표의 끝)에서 시계 방향이 양의 회전으로 정한다.



다음 그림에서 비행기 객체(모델의 인스턴스) 3개는 진행 방향이 다르다. 비행기 모델의 앞 방향은 벡터 $\boldsymbol{a}$이다. 월드 좌표계에서 각 비행기 객체의 진행 방향을 벡터 $\boldsymbol{b}$라고 하면 모델 비행기를 회전(모델 좌표계에서 자전)시켜야 한다. 회전을 하려면 회전축을 알아야 한다. 회전축은 외적 연산 $\boldsymbol{b}\times\boldsymbol{a}$ 또는 $\boldsymbol{a}\times\boldsymbol{b}$로 구할 수 있다.



정리하면, 벡터의 외적 연산은 면의 방향을 구하기 위하여, 회전축을 구하기 위하여 사용할 수 있다는 것이다.

⑥ 벡터의 내적(Dot Product, Inner Product)

3-차원 벡터 $\boldsymbol{a}=(x_1,y_1,z_1)$와 $\boldsymbol{b}=(x_2,y_2,z_2)$의 내적 $\boldsymbol{a}\bullet\boldsymbol{b}$은 다음과 같이 정의한다. 벡터의 내적의 결과는 벡터가 아닌 실수 값(Scalar: 스칼라)이다.

$$\boldsymbol{a}\bullet\boldsymbol{b}=x_1x_2+y_1y_2+z_1z_2$$

다음은 벡터 $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$와 실수 k, p에 대한 내적 연산에 대하여 성립하는 성질이다.

- $\boldsymbol{a}\bullet\boldsymbol{a}=|\boldsymbol{a}|^2$

- $\boldsymbol{a} \cdot \boldsymbol{a} \geq 0$
- $|\boldsymbol{a}| = \sqrt{\boldsymbol{a} \cdot \boldsymbol{a}}$
- $(\boldsymbol{a} \cdot \boldsymbol{a} = 0) \Leftrightarrow (\boldsymbol{a} = \boldsymbol{0})$
- $\boldsymbol{a} \cdot \boldsymbol{b} = \boldsymbol{b} \cdot \boldsymbol{a}$ 교환 법칙(Commutative)
- $\boldsymbol{a} \cdot (\boldsymbol{b} + \boldsymbol{c}) = \boldsymbol{a} \cdot \boldsymbol{b} + \boldsymbol{a} \cdot \boldsymbol{c}$ 덧셈에 대한 분배 법칙(Distributive)
- $\boldsymbol{a} \cdot (k\boldsymbol{b} + \boldsymbol{c}) = k(\boldsymbol{a} \cdot \boldsymbol{b}) + \boldsymbol{a} \cdot \boldsymbol{c}$ 덧셈과 스칼라 곱에 대한 분배 법칙(Bilinear)
- $(k\boldsymbol{a}) \cdot \boldsymbol{b} = k(\boldsymbol{a} \cdot \boldsymbol{b}) = \boldsymbol{a} \cdot (k\boldsymbol{b})$
- $(k\boldsymbol{a}) \cdot (p\boldsymbol{b}) = kp(\boldsymbol{a} \cdot \boldsymbol{b})$
- $\boldsymbol{a} \cdot (\boldsymbol{b} \cdot \boldsymbol{c}) \neq (\boldsymbol{a} \cdot \boldsymbol{b}) \cdot \boldsymbol{c}$
- $(\boldsymbol{a} \cdot \boldsymbol{b} = \boldsymbol{a} \cdot \boldsymbol{c}) \not\Rightarrow (\boldsymbol{b} = \boldsymbol{c})$

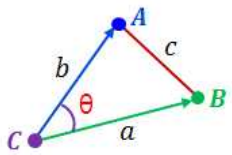
벡터 $\boldsymbol{a} = (x_1, y_1, z_1)$와 $\boldsymbol{b} = (x_2, y_2, z_2)$의 내적 $\boldsymbol{a} \cdot \boldsymbol{b}$ 은 다음과 같이 표현할 수 있다.

$$\boldsymbol{a} \cdot \boldsymbol{b} = x_1x_2 + y_1y_2 + z_1z_2 = |\boldsymbol{a}||\boldsymbol{b}|\cos(\theta)$$

$\boldsymbol{a} \cdot \boldsymbol{b} = |\boldsymbol{a}||\boldsymbol{b}|\cos(\theta)$에 대한 증명은 다음과 같다.

a, b, c가 삼각형의 세 변의 길이이고 θ가 두 변 a, b의 사이각일 때 다음이 성립한다.

$c^2 = a^2 + b^2 - 2ab\cos(\theta)$ ($\because$ 코사인 제2법칙)



위의 삼각형에서 $\boldsymbol{b} = \overrightarrow{CA} = \boldsymbol{A} - \boldsymbol{C}$, $\boldsymbol{a} = \overrightarrow{CB} = \boldsymbol{B} - \boldsymbol{C}$, $\boldsymbol{c} = \overrightarrow{BA} = \boldsymbol{A} - \boldsymbol{B}$라고 하자.

$\boldsymbol{c} = \boldsymbol{A} - \boldsymbol{B} = (\boldsymbol{A} - \boldsymbol{C}) - (\boldsymbol{B} - \boldsymbol{C}) = \boldsymbol{b} - \boldsymbol{a}$이다.

$|\boldsymbol{c}|^2 = \boldsymbol{c} \cdot \boldsymbol{c} = (\boldsymbol{b} - \boldsymbol{a}) \cdot (\boldsymbol{b} - \boldsymbol{a})$ ($\because \boldsymbol{a} \cdot \boldsymbol{a} = |\boldsymbol{a}|^2$)

$|\boldsymbol{c}|^2 = (\boldsymbol{b} - \boldsymbol{a}) \cdot (\boldsymbol{b} - \boldsymbol{a}) = \boldsymbol{b} \cdot \boldsymbol{b} - \boldsymbol{b} \cdot \boldsymbol{a} - \boldsymbol{a} \cdot \boldsymbol{b} + \boldsymbol{a} \cdot \boldsymbol{a}$

$(\boldsymbol{b} - \boldsymbol{a}) \cdot (\boldsymbol{b} - \boldsymbol{a}) = \boldsymbol{b} \cdot \boldsymbol{b} - 2(\boldsymbol{a} \cdot \boldsymbol{b}) + \boldsymbol{a} \cdot \boldsymbol{a}$ ($\because \boldsymbol{a} \cdot \boldsymbol{b} = \boldsymbol{b} \cdot \boldsymbol{a}$)

$(\boldsymbol{b} - \boldsymbol{a}) \cdot (\boldsymbol{b} - \boldsymbol{a}) = |\boldsymbol{a}|^2 - 2(\boldsymbol{a} \cdot \boldsymbol{b}) + |\boldsymbol{b}|^2$

$|\boldsymbol{c}|^2 = |\boldsymbol{b} - \boldsymbol{a}|^2 = |\boldsymbol{b}|^2 - 2(\boldsymbol{a} \cdot \boldsymbol{b}) + |\boldsymbol{a}|^2$

삼각형의 세변의 길이를 $|\boldsymbol{A} - \boldsymbol{B}| = |\boldsymbol{c}| = c$, $|\boldsymbol{A} - \boldsymbol{C}| = |\boldsymbol{a}| = a$, $|\boldsymbol{B} - \boldsymbol{C}| = |\boldsymbol{b}| = b$라고 하자.

$c^2 = b^2 - 2(\boldsymbol{a} \cdot \boldsymbol{b}) + a^2 = a^2 - 2(\boldsymbol{a} \cdot \boldsymbol{b}) + b^2$

$c^2 = a^2 + b^2 - 2ab\cos(\theta)$ ($\because$ 코사인 제2법칙)

$c^2 = a^2 - 2(\boldsymbol{a} \cdot \boldsymbol{b}) + b^2 = a^2 - 2ab\cos(\theta) + b^2$

$\boldsymbol{a} \cdot \boldsymbol{b} = ab\cos(\theta) = |\boldsymbol{a}||\boldsymbol{b}|\cos(\theta)$ (θ는 벡터 $\boldsymbol{a}$와 $\boldsymbol{b}$의 각도)

이 증명이 중요한 것이 아니다. 다음을 꼭 기억하라. 두 벡터 $\boldsymbol{a}$, $\boldsymbol{b}$가 주어지면 두 벡터

의 크기와 내적을 계산할 수 있고, 두 벡터가 이루는 각도 θ의 코사인 값 $\cos(\theta)$를 다음과 같이 계산할 수 있다는 것이다. $\cos(\theta)$를 계산할 수 있으면 θ는 함수 acos()로 구할 수 있다.

$$\boldsymbol{a} = (x_1, y_1, z_1),\ \boldsymbol{b} = (x_2, y_2, z_2)$$

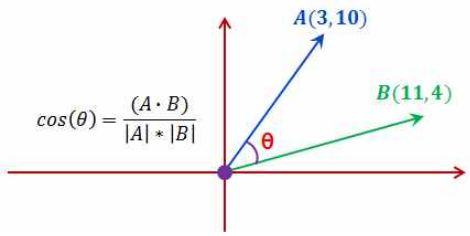
$$\boldsymbol{a} \cdot \boldsymbol{b} = x_1x_2 + y_1y_2 + z_1z_2,\ |\boldsymbol{a}| = \sqrt{x_1^2 + y_1^2 + z_1^2},\ |\boldsymbol{b}| = \sqrt{x_2^2 + y_2^2 + z_2^2}$$

$$\boldsymbol{a} \cdot \boldsymbol{b} = |\boldsymbol{a}||\boldsymbol{b}|\cos(\theta)$$

$$\cos(\theta) = \frac{\boldsymbol{a} \cdot \boldsymbol{b}}{|\boldsymbol{a}||\boldsymbol{b}|} = \frac{x_1x_2 + y_1y_2 + z_1z_2}{\sqrt{x_1^2 + y_1^2 + z_1^2}\sqrt{x_2^2 + y_2^2 + z_2^2}}$$

두 개의 3-차원 벡터에 대한 내적 연산은 두 개의 벡터가 이루는 각도를 구할 수 있는 연산이다. 두 벡터가 이루는 각도 θ는 두 벡터가 같은 시작점을 가질 때의 각도이다.

다음은 이차원 평면에서 두 벡터가 이루는 각도를 내적 연산으로 구하는 예이다.



$$\cos(\theta) = \frac{3*11 + 10*4}{\sqrt{3^2 + 10^2}\sqrt{11^2 + 4^2}} = \frac{73}{122.20063}$$

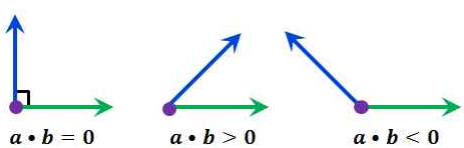
$$\theta = acos(0.5973782) = 0.93056\,rad = 53.31^\circ$$

벡터 $\boldsymbol{a}$와 $\boldsymbol{b}$가 서로 수직하면 $\cos(90) = 0$이므로 $\boldsymbol{a} \cdot \boldsymbol{b} = 0$이다.

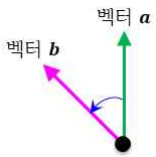
- 내적 연산의 응용: 두 벡터가 이루는 각도를 계산

  다음 그림은 두 벡터의 내적 연산의 결과 값의 부호에 따라 두 벡터가 이루는 각이 90°보다 작은가 또는 큰가를 판단할 수 있음을 보여주고 있다. $(\theta > 90)$이면 $\cos(\theta) < 0$이고, $(0 < \theta < 90)$이면 $\cos(\theta) > 0$이다.

  - 두 벡터의 내적이 0이면 두 벡터는 수직이다.
  - 두 벡터의 내적이 0보다 크면 두 벡터가 이루는 각은 예각(90°보다 작다)이다.
  - 두 벡터의 내적이 0보다 작으면 두 벡터가 이루는 각은 둔각(90°보다 크다)이다.



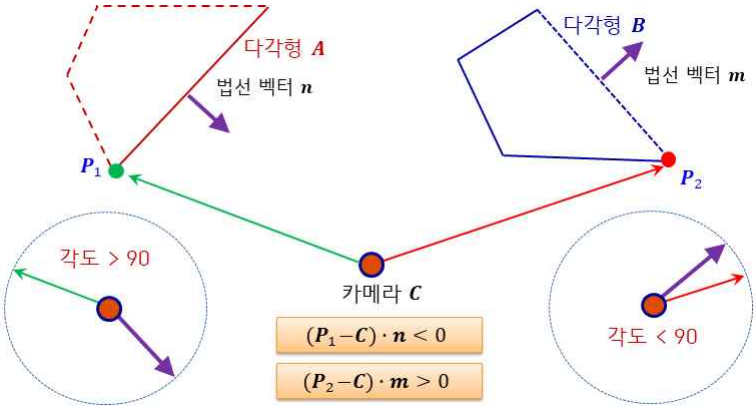
앞에서 다음과 같은 회전의 문제에서 회전축은 두 벡터의 외적 연산으로 계산할 수 있었다. 이제 두 벡터의 회전 각도는 두 벡터의 내적 연산으로 계산할 수 있다.



- 내적 연산의 응용: 은면 제거(Hidden Surface Removal)

다각형의 법선 벡터(면이 향하는 방향)를 알면 그 다각형의 면이 카메라를 향하는 가의 여부를 결정할 수 있다. 이 테스트는 카메라 위치에서 다각형의 평면의 한 점까지의 벡터와 다각형 평면의 법선 벡터의 내적 연산을 하는 것이다. 내적의 결과가 음수이면 다각형이 카메라를 향하고 있는 경우이다. 내적의 결과가 양수이면 다각형은 카메라를 향하지 않는 것을 나타낸다.

다음 그림에서 카메라 C에서 다각형 A의 정점 P_1까지의 벡터 $(P_1 - C)$와 다각형 A의 법선 벡터 n이 이루는 각도는 90도 보다 크므로 내적의 결과는 음수이다. 이 경우 다각형 A는 카메라에 보이게 된다. 다각형 B의 정점 P_2까지의 벡터$(P_2 - C)$와 다각형 B의 법선 벡터 m이 이루는 각도는 90°보다 작으므로 내적의 결과는 양수이다. 이 경우 다각형 B는 카메라 위치에서 보이지 않는다.

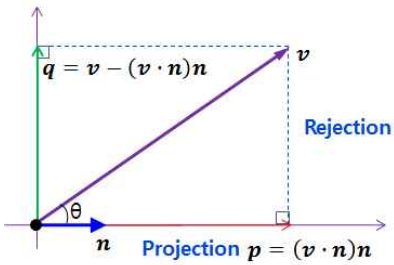


- 내적 연산의 응용(직교 투영: Orthogonal Projection)

내적 연산을 응용하면 어떤 벡터를 서로 수직인 두 개의 벡터의 합으로 표현할 수 있다. 어떤 벡터를 주어진 단위 벡터 n으로 표현하는 것을 직교 투영이라고 한다. 단위 벡터 n에 대한 벡터 v의 직교 투영은 다음과 같이 표현할 수 있다.

$$p = (v \cdot n)n$$

$$p = (|v|\cos(\theta))n = (|v||n|\cos(\theta))n = (v \cdot n)n \qquad (\because |n| = 1)$$



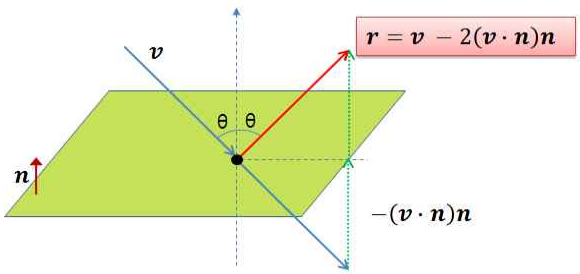
단위 벡터가 아닌 벡터 w에 대한 벡터 v의 직교 투영은 다음과 같이 표현할 수 있다.

$$p = \frac{(v \cdot w)w}{|w|^2}$$

- 내적 연산의 응용(반사 벡터: Reflection Vector)

  내적 연산을 응용하면 법선 벡터 n을 갖는 평면에 대하여 3-차원 벡터 v를 반사시킨 반사 벡터 r을 다음과 같이 표현할 수 있다.

$$r = v - 2(v \cdot n)n$$



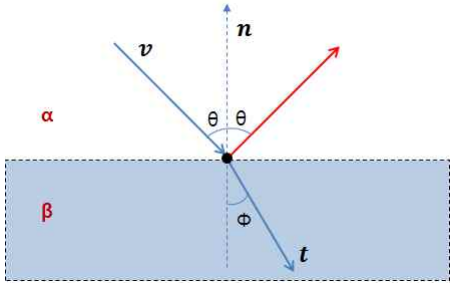
- 내적 연산의 응용(굴절 벡터: Refraction Vector)

  내적 연산을 응용하면 3-차원 벡터 v를 법선 벡터 n을 갖는 평면에 대한 굴절 벡터 t을 다음과 같이 표현할 수 있다. α와 β는 굴절율이다.

$$\lambda = \frac{\alpha}{\beta}$$

$$k = \sqrt{1 - \lambda^2(1 - (-(v \cdot n))^2)}$$

$$t = (\lambda v + (-(v \cdot n)) - k)n$$



⑦ 내적 연산과 외적 연산의 활용

다음은 3-차원 벡터 $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$에 대하여 내적 연산과 외적 연산에 대하여 성립하는 성질이다.

- $\boldsymbol{a} \cdot (\boldsymbol{b} \times \boldsymbol{c}) = \boldsymbol{b} \cdot (\boldsymbol{c} \times \boldsymbol{a}) = \boldsymbol{c} \cdot (\boldsymbol{a} \times \boldsymbol{b})$
- $\boldsymbol{a} \cdot (\boldsymbol{b} \times \boldsymbol{c}) = |\boldsymbol{a}||\boldsymbol{b} \times \boldsymbol{c}|\cos(\theta)$
- $(\boldsymbol{a} \cdot \boldsymbol{b} = \boldsymbol{a} \cdot \boldsymbol{c}) \wedge (\boldsymbol{a} \times \boldsymbol{b} = \boldsymbol{a} \times \boldsymbol{c}) \Rightarrow (\boldsymbol{b} = \boldsymbol{c})$
- $\boldsymbol{a} \times (\boldsymbol{b} \times \boldsymbol{c}) = (\boldsymbol{a} \cdot \boldsymbol{c})\boldsymbol{b} - (\boldsymbol{a} \cdot \boldsymbol{b})\boldsymbol{c}$
- $|\boldsymbol{a} \times \boldsymbol{b}|^2 = |\boldsymbol{a}|^2|\boldsymbol{b}|^2 - (\boldsymbol{a} \cdot \boldsymbol{b})^2$

$|\boldsymbol{a} \times \boldsymbol{b}|^2 = |\boldsymbol{a}|^2|\boldsymbol{b}|^2 - (\boldsymbol{a} \cdot \boldsymbol{b})^2$을 활용하면 $|\boldsymbol{a} \times \boldsymbol{b}| = |\boldsymbol{a}||\boldsymbol{b}|\sin(\theta)$를 다음과 같이 증명할 수 있다.

$$|\boldsymbol{a} \times \boldsymbol{b}|^2 = |\boldsymbol{a}|^2|\boldsymbol{b}|^2 - (\boldsymbol{a} \cdot \boldsymbol{b})^2$$

$$|\boldsymbol{a} \times \boldsymbol{b}|^2 = |\boldsymbol{a}|^2|\boldsymbol{b}|^2 - (|\boldsymbol{a}||\boldsymbol{b}|\cos(\theta))^2 \qquad (\because \boldsymbol{a} \cdot \boldsymbol{b} = |\boldsymbol{a}||\boldsymbol{b}|\cos(\theta))$$

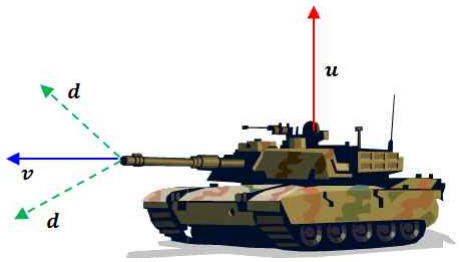
$$|\boldsymbol{a} \times \boldsymbol{b}|^2 = |\boldsymbol{a}|^2|\boldsymbol{b}|^2 - |\boldsymbol{a}|^2|\boldsymbol{b}|^2\cos^2(\theta) = |\boldsymbol{a}|^2|\boldsymbol{b}|^2(1 - \cos^2(\theta)) \equiv |\boldsymbol{a}|^2|\boldsymbol{b}|^2\sin^2(\theta)$$

$$|\boldsymbol{a} \times \boldsymbol{b}| = |\boldsymbol{a}||\boldsymbol{b}|\sin(\theta)$$

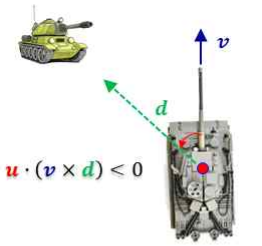
- 스칼라 삼중적(Scalar Triple Products)

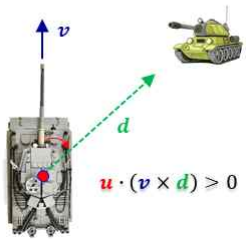
스칼라 삼중적은 3-차원 벡터의 내적과 외적을 함께 사용하여 최단 회전의 문제를 해결할 수 있다. 객체의 방향을 원하는 방향으로 일치시키는 방법은 시계 방향의 회전과 반시계 방향의 회전 2가지가 존재한다.

다음 그림과 같이 게임의 NPC(Non Player Character)가 탱크라고 가정하자. 탱크의 포신이 향하고 있는 현재의 방향 벡터가 $\boldsymbol{v}$이고 적 탱크로 향하는 벡터(포신이 적 탱크를 향하도록 하는 방향 벡터)가 $\boldsymbol{d}$라고 가정하자. 방향 벡터 $\boldsymbol{d}$는 적 탱크의 위치 벡터에서 NPC 탱크의 위치 벡터를 빼고 정규화하면 된다. 그리고 회전할 각도 θ는 벡터 $\boldsymbol{v}$와 $\boldsymbol{d}$를 내적하면 구할 수 있다. 포탑의 회전축 $\boldsymbol{u}$는 이미 주어졌으므로(알고 있으므로) 탱크의 포탑을 회전축 $\boldsymbol{u}$를 중심으로 θ만큼 회전하면 될 것이다(탱크의 포신이 연결된 포탑은 좌우로 회전하고 포신은 위 아래로 회전을 할 수 있고, 탱크 자체는 지형에 따라 기울어져 있을 수 있다).



그러나, 만약에 적 탱크가 포신이 향하고 있는 방향의 왼쪽에 나타나는 경우에는 포탑을 반시계 방향으로 회전을 해야 한다. 적 탱크가 포신이 향하고 있는 방향의 오른쪽에 나타나는 경우에는 포탑을 시계 방향으로 회전을 해야 한다. 즉, 방향 벡터 $\boldsymbol{d}$가 방향 벡터 $\boldsymbol{v}$의 오른쪽이면 포탑을 오른쪽으로 회전하는 것이 최단 회전이 된다. 방향 벡터 $\boldsymbol{d}$가 방향 벡터 $\boldsymbol{v}$의 왼쪽이면 포탑을 왼쪽으로 회전하는 것이 최단 회전이 된다.





회전축이 벡터 $\boldsymbol{u}$일 때 $\boldsymbol{u} \cdot (\boldsymbol{d} \times \boldsymbol{v}) > 0$이면 방향 벡터 $\boldsymbol{d}$가 방향 벡터 $\boldsymbol{v}$의 왼쪽이므로 최단 회전은 반시계 방향이다. $\boldsymbol{u} \cdot (\boldsymbol{d} \times \boldsymbol{v}) < 0$이면 방향 벡터 $\boldsymbol{d}$가 방향 벡터 $\boldsymbol{v}$의 오른쪽이므로 최단 회전은 시계 방향이다. 방향 벡터 $\boldsymbol{d}$가 방향 벡터 $\boldsymbol{v}$의 왼쪽이면, 왼손 좌표계에서 $(\boldsymbol{d} \times \boldsymbol{v})$의 y-좌표는 0보다 크며(위쪽을 향하고 있으며) 벡터 $\boldsymbol{u}$와의 내적은 $\boldsymbol{u} \cdot (\boldsymbol{d} \times \boldsymbol{v}) > 0$이다. 방향 벡터 $\boldsymbol{d}$가 방향 벡터 $\boldsymbol{v}$의 오른쪽이면, 왼손 좌표계에서 $(\boldsymbol{d} \times \boldsymbol{v})$의 y-좌표는 0보다 작으며(아래쪽을 향하고 있으며) 벡터 $\boldsymbol{u}$와의 내적은 $\boldsymbol{u} \cdot (\boldsymbol{d} \times \boldsymbol{v}) < 0$이다.

스칼라 삼중적은 게임에서 캐릭터가 어떤 객체를 따라갈 때 최단 방향의 회전을 하기 위하여 사용할 수 있다.