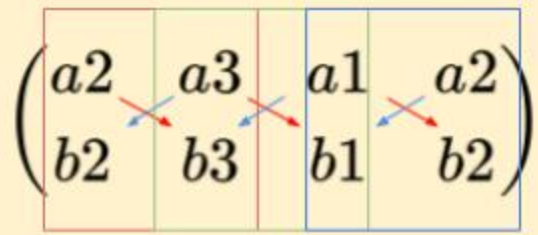
■ 벡터의 외적 (only 3차원 공간벡터)

$a = (a_1 , a_2 , a_3)$
$b = (b_1 , b_2 , b_3)$

$a \times b = (a_2b_3 - a_3b_2 , a_3b_1 - a_1b_3 , a_1b_2 - a_2b_1)$



빨: 더하기
파: 빼기

x y z

외적(cross product)

길고 긴 내적의 과정을 지나 또 다른 벡터의 곱셈인 외적에 대해서 알아볼 차례입니다.
벡터의 내적을 배웠을 때 생각보다 어렵지 않았었죠?
외적도 알고 보면 별거 아닙니다.
일단 부딪쳐 봐야 하니 벡터의 외적에 대한 사전적 정의를 살펴봅시다.

> 벡터의 외적은 두 벡터 a,b 사이의 각을 θ라 하면 a·b sin θ라는 크기, 즉 a,b를 두 변으로 하는 평행사변형의 넓이와 같은 크기를 가지고 a,b를 포함하는 평면에 수직이고...

(출처 : http://terms.naver.com/entry.nhn?docId=1186169&cid=40942&categoryId=32225)

백과사전에서 검색한 외적의 설명입니다.
사실 저 의미를 이해하려면 다음 챕터에 나오는 삼각함수에 대해 알고 있어야 합니다.
따라서 이번 외적 파트를 다 읽고도 뭔가 찝찝함이 남아 있는 분들이라면 다음 챕터에 나오는 삼각함수에 대한 내용을 이해한 뒤에 다시 돌아와 한번 더 학습하시길 권해드립니다.
일단 누구나 알 수 있도록 쉬운 부분부터 설명해 나가도록 하겠습니다.

내적의 결과값은 벡터가 아닌 스칼라값이 나온다는 사실은 다들 아시겠죠?
외적의 결과값은 또다른 벡터 하나가 생기는데 이 벡터는 두 벡터에 모두 수직인 벡터가 됩니다.
(한가지 주의해야 할 점은 벡터의 외적은 3차원 벡터에서만 적용 된다는 것입니다.)

내적을 설명할 때 법선벡터 라는 말을 잠깐 언급했는데 이 법선벡터를 구할 때 벡터의 외적을 사용하게 됩니다.
법선벡터란 어떤 평면에 수직인 벡터를 의미하므로 위에서 말씀드린 외적의 정의와 완벽히 일치하게 되는 것이죠.

어떤 두 벡터의 외적을 구하는 수식은 다음과 같습니다.

$$a \times b = [(a_y \cdot b_z - a_z \cdot b_y), (a_z \cdot b_x - a_x \cdot b_z), (a_x \cdot b_y - a_y \cdot b_x)]$$

우리는 프로그래밍 언어가 더 친숙하므로 코드를 통해 외적을 계산해 보겠습니다.

미리보기 | 소스복사

```
1. Vector3 vector_cross(Vector3 a, Vector3 b)
2. {
3.     return new Vector3(
4.         a.y * b.z - a.z * b.y,
5.         a.z * b.x - a.x * b.z,
6.         a.x * b.y - a.y * b.x);
7. }
```

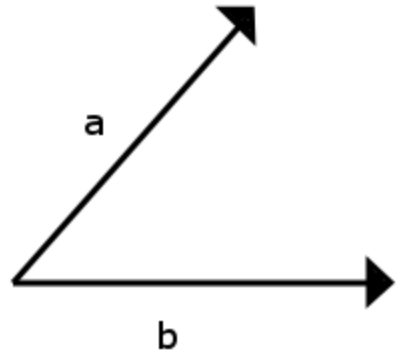
위 수식을 자세히 보면 새로운 벡터의 x성분은 두 벡터의 y,z값을 각각 교차해서 곱한 뒤 서로를 뺀 값이라는 것을 알 수 있습니다.
새로운 벡터의 y,z성분도 동일한 규칙으로 계산을 한 것이죠.
수식이 중요한건 아니니 그냥 한번 읽고 넘어가시면 될 듯 합니다.
중요한건 저런 계산을 통해 나온 새로운 벡터는 두 벡터의 수직이 되는 벡터라는 것입니다.
그 벡터는 두 벡터를 통해 만들어진 평면에 수직인 벡터라고 할 수 있으며
그것을 평면의 법선벡터 또는 노말벡터 라고 부르죠.
보통 3D 제작툴에서 모델을 익스포트할 때 법선벡터도 같이 뽑아져 나오기 때문에 프로그램에서 직접 계산해줄 일은 많지 않을겁니다.
하지만 어떤 원리를 통해서 법선벡터가 생기는지 알고 있다면 활용범위가 더 넓어질 수 있겠죠.

평행사변형의 넓이

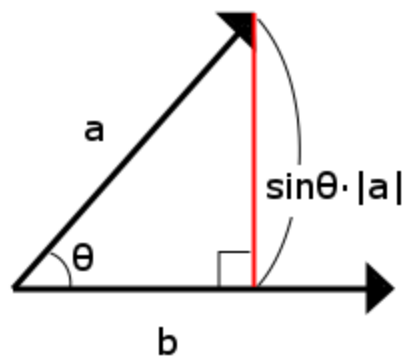
벡터의 외적을 게임 프로그래밍에서가 아닌 수학적으로 설명한 책이나 자료를 보면
아래와 같은 공식을 발견할 수 있습니다.

$$\mathbf{a} \times \mathbf{b} = |\mathbf{a}| \cdot |\mathbf{b}| \cdot \sin\theta \cdot \mathbf{n}$$

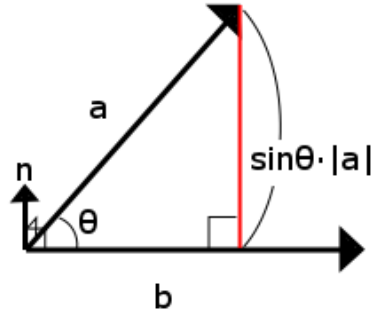
저 공식을 풀어서 설명해 드리자면,
두 벡터 **a**와 **b**의 외적은 두 벡터가 이루는 각도의 **사인(sin)**값에 두 벡터의 **길이**를 곱해서
두 벡터의 수직인 단위벡터 **n**을 곱한 값이라는 뜻입니다.
(**|a|** 이 기호는 벡터 **a**의 길이를 나타내는 표시입니다.)
쉽게 이해할 수 있도록 그림을 통해서 보겠습니다.



두 벡터 **a**와 **b**가 있습니다.

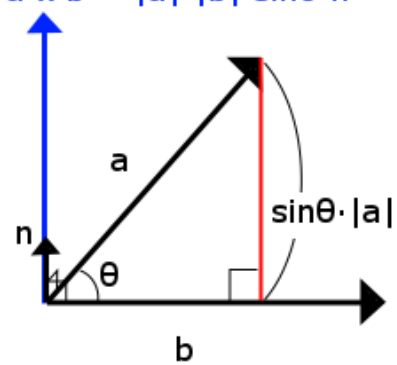


두 벡터 **a**와 **b**가 이루는 각도를 θ(세타)라고 했을 때
그림에서 보이는 빨간색 선(**b**에 수직인 선)의 길이는 sinθ에 **a**의 길이를 곱한 값이 됩니다.



a와 **b**에 모두 수직이면서 길이가 1인 벡터 **n**이 있다고 할 때
(벡터 **n**은 화면 위쪽이 아닌 모니터의 앞쪽으로 나온다고 생각하면 됩니다. 그래야 **a**, **b**에 모두 수직인 벡터가 되니까요)
sinθ에 **a**의 길이와 **b**의 길이를 곱한 값을 **n**에 곱해주면 최종적으로 벡터 **n**을 길게 늘려준 꼴이 됩니다.
왜냐하면 sinθ에 **a**의 길이, **b**의 길이를 곱하면 하나의 스칼라값이 나오는데
여기에 벡터 **n**을 곱하면 벡터와 스칼라를 곱하는 것이기 때문에 결국
벡터 **n**이 확장된 모습이 되기 때문이죠.

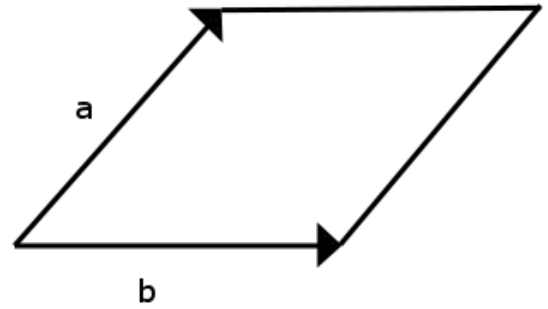
$$\mathbf{a} \times \mathbf{b} = |\mathbf{a}| \cdot |\mathbf{b}| \cdot \sin\theta \cdot \mathbf{n}$$



결국 **a**와 **b**의 외적은 $|\mathbf{a}| \cdot |\mathbf{b}| \cdot \sin\theta \cdot \mathbf{n}$ 으로도 구할 수 있습니다.

하지만 이 방법은 두 벡터 사이의 각도와 수직인 벡터 **n**을 알아야 하며 삼각함수까지 들어가 있기 때문에 계산 과정이 복잡해집니다.
따라서 실무에서는 벡터의 외적을 구할 때 이 방법 보다는 첫 번째로 설명 드린 방법(벡터의 각 성분들을 곱해서 빼는 방법)을 사용하게 되죠.

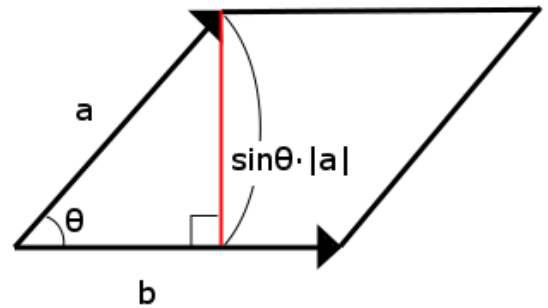
이러한 계산 방법을 통해서 한 가지 재미있는 사실을 알아낼 수 있는데요,
벡터의 외적을 이용하면 두 벡터가 만드는 평행사변형의 넓이를 구할 수 있다는 것입니다.
벡터 **a**와 **b**가 만드는 평행사변형은 다음과 같이 나타낼 수 있습니다.



평행사변형의 넓이를 구하는 공식을 다시 떠올려 봅시다.

```
평행사변형의 넓이 = 밑변 x 높이
```

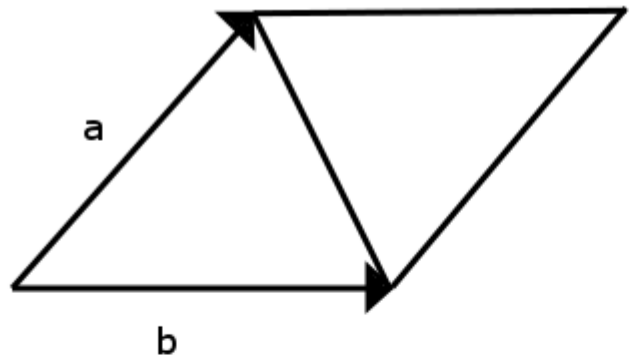
위 그림에서 밑변은 **b**의 길이, 높이는 **a**에서 **b**로 수직으로 내려오는 선이라고 할 수 있죠.



높이를 구하려면 벡터 **a**와 **b**가 이루는 각도를 삼각함수 사인(sin)으로 계산한 값에 **a**의 길이를 곱한 것과 같습니다.
(이 부분은 삼각함수 부분을 알아야 이해할 수 있습니다.)
결국 높이 = **|a|**·sinθ 가 되며, 밑변 = **b**의 길이 이므로
밑변(**|b|**) × 높이(**|a|**·sinθ)를 통해서 두 벡터 **a**와 **b**가 만드는 평행사변형의 넓이를 구할 수 있는 것입니다.

이 공식은 벡터의 외적을 구하는 공식 **|a|**·**|b|**·sinθ·**n**에서 벡터 **n**만 제외한 것과 같죠.
결국 외적으로 구해진 벡터의 길이는 평행사변형의 넓이와 같다는 것을 알 수 있습니다.

또한 평행사변형을 둘로 나누면 삼각형 두 개가 생기기 때문에 외적을 통해 생긴 벡터의 길이를 2로 나누면 두 벡터가 만드는 삼각형의 넓이와 같게 됩니다.



두 벡터 **a**와 **b**가 만드는 삼각형의 넓이는 평행사변형의 넓이의 절반과 같다.

외적을 끝으로 벡터에 대한 기초를 학습해 봤습니다.
벡터를 처음 접했을 때 막연했던 느낌이 많이 사라지셨는지요?
다음 챕터는 벡터에 이어 자주 사용되는 삼각함수에 대한 부분입니다.
이 부분도 모르면 어렵고 알면 쉽습니다.

참고로 이번에 배운 벡터의 개념은 3D 게임 프로그래밍에서도 똑같이 적용 됩니다.
만약 유니티 엔진을 사용할 줄 안다면 게임 오브젝트의 transform.position값을 벡터로 생각하고 지금까지 배웠던 내용들을 코딩하여 구현해 보세요.
수학적으로 계산하던 결과를 직접 컴퓨터를 통해서 확인할 수 있기 때문에 이해가 더욱 잘될 것이라 생각합니다.

다음 강좌는 삼각함수에 대한 내용입니다.
게임 프로그래밍에서 자주 사용되는 삼각함수의 기초적인 의미와 쓰임에 대해 알아볼 것입니다.
감사합니다.