|  |
| --- |
| 03 3D 게임 프로그래밍을 위한 기초 수학 - 3. 벡터의 뺄셈, 벡터와 스칼라의 곱셈  3D 게임 프로그래밍을 위한 기초 수학에 대한 연재 강좌를 시작합니다.  저도 수학에 대해서 잘 모르지만 공부하면서 알게된 지식을 공유하기 위한 목적으로 올립니다.  벡터와 삼각함수 부분을 연재해 나갈 계획이며,  수학에 기초가 약한 분들을 대상으로 쉽게 쓰려고 합니다.  따라서 이미 베테랑이신 분들은 다 아는 내용일지도 모릅니다.  혹시 내용중 잘못된 부분에 대한 지적이나 의견을 주시면 너무 감사하겠습니다.  적극적인 피드백 부탁드립니다.^^  지난 강좌  [3D 게임 프로그래밍을 위한 기초 수학 - 1. 시작](http://lab.gamecodi.com/board/zboard.php?id=GAMECODILAB_Lecture_series&no=121&z=)  [3D 게임 프로그래밍을 위한 기초 수학 - 2. 벡터](http://lab.gamecodi.com/board/zboard.php?id=GAMECODILAB_Lecture_series&no=122&z=) |

**벡터의 뺄셈**

이번에는 벡터의 뺄셈에 대해서 알아보도록 하겠습니다. 뺄셈은 덧셈으로도 표현할 수 있다는 사실을 아시나요?

1 – 1 = 0

1 + (-1) = 0

장난 같지만 위 수식처럼 뺄셈은 음수값을 더하는 형태로 나타낼 수 있죠.

이런 형태는 벡터의 뺄셈에도 똑같이 적용됩니다.

뺄셈을 덧셈처럼 생각한다면 아까 배웠던 벡터의 덧셈을 그대로 사용할 수 있으니 계산하는데 더 편할 겁니다.

a = [5,0]

b = [6,5]

b – a = b + [-a]

= [6,5] + [-[5,0] ]

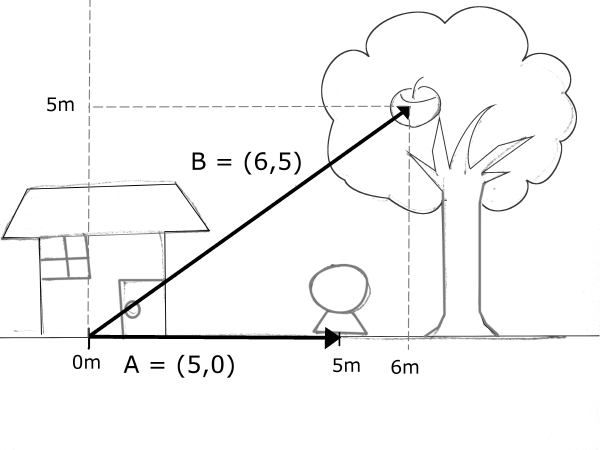
= [6,5] + [-5,0]

= [6 + [-5], 5 + 0]

= [1, 5]

벡터의 뺄셈은 간단히 각 자릿수에 대해서 연산을 해주면 끝입니다만

이것만 봐서는 잘 이해가 안 되기 때문에 그림으로 벡터의 뺄셈을 알아보도록 하죠.

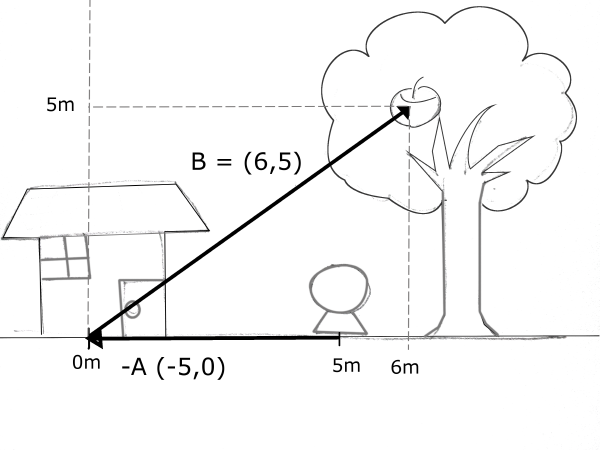


벡터b에서 벡터a를 뺀다는 것은 b – a라는 식으로 나타낼 수 있습니다.

위에서 뺄셈은 덧셈으로 표현할 수 있다고 말씀드렸었죠.

그렇다면 b – a = b + (-a) 로 표현할 수 있습니다.

벡터에 –를 붙인다는 것은 크기는 그대로 두면서 방향만 반대로 바꾸는 것입니다.

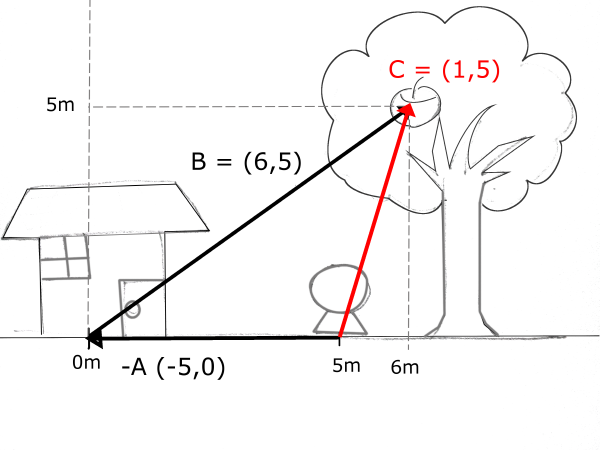


벡터a에 –부호를 붙이면 위 그림과 같이 길이는 그대로 유지하면서 방향만 반대로 되는 벡터가 됩니다.

-a벡터를 구했으므로 아까 배운 벡터의 덧셈을 이용하여 두 벡터를 더해봅시다.

간단하겠죠? 벡터의 덧셈은 꼬리에서 머리로 이어주면 된다고 했습니다.

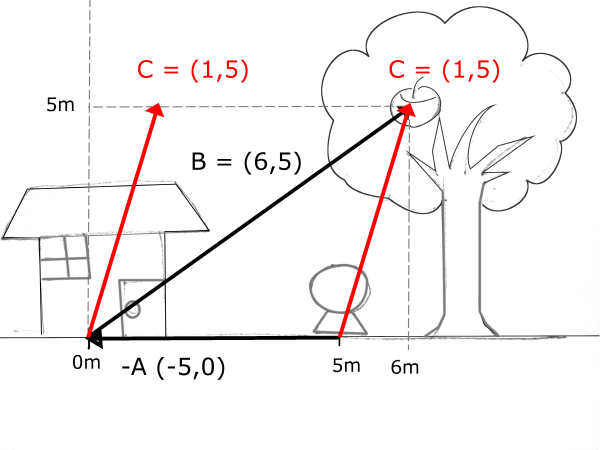
a의 꼬리에서 b의 머리까지 이어주면 아래와 같은 벡터가 생깁니다.



벡터의 뺄셈도 덧셈처럼 생각한다면 특별히 어려운 개념은 없을 것입니다.

뺄셈을 통해 생겨난 벡터도 다른 벡터들과 완전히 똑같은 특성을 갖게 되죠.

아래 그림을 봅시다.



위 그림에서 빨간색으로 나타낸 두 개의 벡터는 완전히 동일한 벡터입니다.

왜냐하면 크기와 방향이 같기 때문이죠.

이처럼 벡터를 볼 때는 **크기(길이)와 방향**이 어떻게 생겼는지를 봐야 헷갈리지 않습니다.

그럼 벡터의 뺄셈은 어디에 활용될 수 있을까요?

위 그림에서 벡터c는 사람에서 사과를 향하는 벡터라는 것을 알 수 있죠.

**이처럼 특정 위치에서 다른 위치를 향하는 벡터를 구할 때** 벡터의 뺄셈을 사용한다면

아주 쉽게 해당 벡터를 구할 수 있습니다.

게임 프로그래밍에서는 캐릭터를 특정 장소로 움직이도록 만들 때 사용합니다.

예를 들어 플레이어를 추적하는 몬스터 인공지능을 구현한다고 합시다.

아주 단순하게 생각하면 몬스터가 현재 플레이어가 서 있는 위치를 향해 매 프레임 이동하는 로직을 구상할 수 있습니다.

그렇다면 몬스터의 x,y 좌표에 어떤 계산을 적용해야 플레이어를 향하여 이동하게 만들 수 있을까요?

이 때 몬스터의 이동 방향을 구하는데 벡터의 뺄셈을 활용할 수 있을 것입니다.

플레이어를 향하는 방향 = (플레이어의 위치 - 몬스터의 위치)

몬스터의 위치와 플레이어의 위치를 벡터로 간주한다면 앞서 배운 벡터의 뺄셈을 그대로 적용시킬 수 있습니다.

계산된 결과는 몬스터에서 플레이어를 잇는 벡터가 되는데, 이 벡터를 따라서 몬스터를 움직이게 하면 됩니다.

Tip.

실제 코드를 구현할 때에는 이 벡터를 단위벡터로 만든 뒤

몬스터의 이동속력을 곱하여 최종적으로 속도 벡터를 얻는 과정이 필요합니다.

이렇게 얻은 속도 벡터에 몬스터의 현재 위치를 가리키는 벡터를 매 프레임마다 더해주면

몬스터가 플레이어의 위치로 움직이는 모습이 구현되죠.

게임 내에서는 매 프레임마다 몬스터의 위치를 갱신해 주는 코드가 들어갈 테니

프레임 사이의 시간간격을 구해서 속도 벡터에 곱해준 뒤

몬스터의 위치를 가리키는 벡터에 더해야 정상적인 결과가 나오게 됩니다.

**방향 벡터 = 플레이어의 현재 위치 – 몬스터의 현재 위치**

**방향 벡터를 단위 벡터로 만들고 이동 속력을 곱해 속도 벡터를 계산**

**몬스터의 새로운 위치 = 몬스터의 현재 위치 + 시간 \* 속도 벡터**

유니티 엔진을 사용한다면 게임 오브젝트의 위치를 나타내는 값으로 transform.position 변수가 벡터로 정의되어 있으므로 위 수식에 그대로 적용시킬 수 있습니다.

·미리보기 | 소스복사·

1. Vector3 velocity = (player.transform.position – transform.position).normalized;
2. // 몬스터의 이동 속력은 10m/s 로 계산한다. 물론 단위는 정하기 나름이다. 여기서는 미터로 정했다.
3. // 이 값을 크게 주면 몬스터가 더 빠르게 이동하며
4. // 작게 주면 더 천천히 이동하게 된다.
5. velocity  = velocity \* 10.0f;
6. transform.position = transform.position + Time.deltaTime \* velocity;

위 코드를 매 프레임마다 수행한다면 플레이어를 추적하는 몬스터의 모습을 구현할 수 있게 됩니다.

여기에서 몬스터의 이동 속력을 나타내는 값으로 10을 사용했습니다.

이 예제에서는 10m/s라고 정의했는데 10픽셀이라고 단정 지어서 생각하지 않기를 바랍니다.

3D 게임 프로그래밍에서는 카메라의 설정에 따라 화면에 보여지는 물체의 위치가 달라지므로 픽셀단위로 이동 한다는 말은 정확하지 않습니다.

픽셀 보다는 미터 같은 실생활의 단위를 사용하는 것이 더 좋습니다.

**벡터의 뺄셈 정리**

1. 뒤에 있는 벡터의 방향을 반대로 바꾼다.

    b – a 라면 a벡터의 방향을 반대로 바꾼다.

    방향을 바꾸는 것은 길이는 그대로 두고 화살표가 가리키는 곳만 반대로 만들어 주는 것을 의미한다.

2. 두 벡터를 덧셈 규칙에 맞게 꼬리에서 머리로 이어준다.

3. 벡터의 뺄셈은 한 위치에서 다른 위치로 향하는 벡터를 만들 때 사용할 수 있다.

**벡터와 스칼라의 곱셈**

일반적으로 더하기 빼기 다음에는 곱하기를 배웁니다.

벡터도 마찬가지로 곱하기가 존재합니다만 흔히 알고 있는 곱하기와는 약간 다릅니다.

심지어 곱하기의 종류는 두 가지나 있으며 계산 방법마저도 각각 다릅니다.

여기서는 일단 벡터끼리의 곱셈 이전에 벡터와 스칼라의 곱셈에 대해서 알아보겠습니다.

스칼라란 그냥 1,2,3,99,100,...과 같이 익히 알고 있는 수입니다.

벡터와 스칼라의 곱셈은 일반적인 수의 곱셈과 같은 개념입니다.

a = [5,0]

d = 2

a x d = [5,0] x 2 = [10,0]

위 수식에서 a는 벡터이며 d는 스칼라값(그냥 우리가 알고 있는 평범한 숫자라고 생각하면 됩니다) 입니다.

벡터 a [5,0]에 스칼라값 2를 곱하면 [10,0]으로 원래의 벡터보다 두배 커진 벡터가 됩니다.

5 x 2 = 10

0 x 2 = 0 이 되어서 [10,0]이라는 벡터가 됩니다.

그렇다면 음수를 곱하면 어떻게 될까요?

예상하신 대로입니다.

벡터a에 –2를 곱하면 [-10,0] 이 됩니다.

벡터와 스칼라의 곱셈은 정말 간단하며 게임 프로그래밍을 할 때도 적용하기 쉽습니다.

예를 들면 방향 벡터에 속력을 곱해서 속도 벡터로 만드는데 사용할 수 있습니다.

앞서 설명했던 플레이어를 따라가는 몬스터의 인공지능 구현 예를 보면,

몬스터에서 플레이어로 향하는 벡터를 구하고 이 벡터에 속력을 곱해서 속도를 구한다고 했습니다.

이 때 사용하는 연산이 바로 벡터와 스칼라의 곱셈입니다.

별로 특별할 것 없는 단순한 연산입니다.

여기까지 벡터의 덧셈과 뺄셈, 벡터와 스칼라의 곱셈 까지 알아봤습니다.

다음 강좌에서는 벡터끼리의 곱셈인 내적과 외적에 대해서 학습하도록 하겠습니다.

감사합니다.