Unity Lecture 03

Vector 클래스 다루기

VECTOR 란?

수학/물리에서 사용되는 벡터와 흡사하며 내부적으로는 그냥 x, y, z와 같은 필드(멤버 변수)를 가진 클래스입니다. Vector2/3/4 가 있으며 각각 (x, y) / (x, y, z) / (x, y, z, w) 를 가지고 있습니다.

유니티에서는 이 벡터를 좌표나 방향을 나타내는 **거의 모든 곳**에서 사용하고 있으며, 그래서 필수적으로 알아야 하는 개념이기도 합니다.

VECTOR 생성하기

Vector의 생성자는 Vector3(float, float, float) 입니다. 그래서 그냥 다음과 같이 선언하면 됩니다.

```
Vector3 pos = new Vector3(1.5f, 5.5f, 8.5f);
```

여기서, 숫자 뒤에 f를 붙이지 않으면 오류가 발생하는데 그 이유는 그냥 소수점이 있는 숫자는 double 타입 상수이기에 숫자 뒤에 f를 붙여 float임을 나타내 주어야 합니다.

대부분 Vector3을 주로 사용하기 때문에 앞으로 편의상 Vector3으로 예제를 만들겠습니다. 물론 나머지 벡터 사용법도 모두 동일합니다.

VECTOR의 사칙 연산

v1 + v2

(v1.x + v2.x, v1.y + v2.y, v1.z + v2.z)

서로의 x,y,z를 따로 더한 벡터가 반환됩니다.

v1 - v2

위와 같지만 뺄셈이 됩니다.

v * s

(v.x * s, v.y * s, v.z * s)

벡터끼리 곱셈은 되지 않으며, 대신 스칼라(크기가 1인 벡터, 즉 float 값) 와 곱해서 모든 원소에 s를 곱해준 값이 반환됩니다.

v / s

위와 같지만 나눗셈이 됩니다.

속성들

- magnitude

: 벡터의 크기를 구해줍니다. 피타고라스 법칙을 이용하여 벡터의 모든 원소들의 제곱의 합의 제곱근이 크기입니다.

- sqrMagnitude

: 위와 비슷하지만 제곱근을 씌우지 않은 결과를 반환합니다. 제곱근 연산이 컴퓨터가 하기 힘든 연산이기 때문인데, 대소비교만 필요할 때 주로 사용됩니다.

- normalized

: 정규화를 한 값을 반환합니다. 자신을 자신의 크기로 나누면 구해지며, 크기가 1인 벡터가 되며 방향만 나타내고 싶을 때 사용됩니다.

- Vector3.right/up/forward/zero/one

: (1,0,0) / (0,1,0) / (0,0,1) / (0,0,0) / (1,1,1) 의 벡터들입니다. 가끔 사용됩니다.

내적DOT PRODUCT

Vector3.Dot(Vector, Vector)
메서드를 사용해서 내적을 할 수 있습니다. 양수면 같은 방향으로 향하는 벡터고, 0이면 서로 수직인 벡터, 음수면 다른 방향으로 향하는 벡터입니다. 단위 벡터의 경우, 1이면 완전히 같은 방향이며 -1이면 완전히 다른 벡터 입니다.

$$\vec{a} \cdot \vec{b} = |\vec{a}|\,|\vec{b}| \cos\theta$$

내적의 식은 위와 같고, 여러 곳에 응용할 수 있는데, 예를들어 플레이어가 보고있는 방향의 벡터와 몬스터로부터 플레이어로 향하는 벡터의 내적으로 플레이어가 보고있지 않을때만 움직이는 몬스터를 만들수도 있습니다.

또, arccos(내적의 결과 / 두 벡터 크기의 곱) 으로 사이각을 알아내서 몬스터의 시야 범위 안에 플레이어가 들어와있는지를 확인할 수도 있습니다.

$$\theta = \cos^{-1}\left(\frac{v_1 \cdot v_2}{\|v_1\|\,\|v_2\|}\right)$$

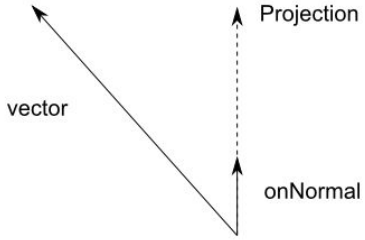
외적CROSS PRODUCT

Vector3.Cross(Vector3, Vector3)
메서드를 사용해서 외적을 할 수 있습니다. 이는 크게 사용되지는 않지만 평면의 법선 벡터(Normal 벡터)를 구할때 등에 사용될 수 있습니다. 벡터 외적의 공식은 아래와 같으며 두 벡터에 수직인 벡터가 반환됩니다.

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

STATIC 메서드들

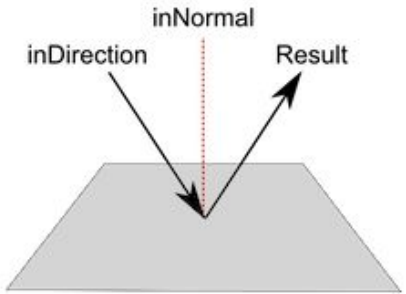
- Vector3.Distance(Vector3, Vector3)
: 두 벡터의 거리를 구합니다.



- Vector3.Project(Vector3 vector, Vector3 onNormal)
: vector 벡터를 onNormal 벡터에 투영한 벡터를 구합니다.

- Vector3.ProjectOnPlane(Vector3 vector, Vector3 planeNormal)
: vector 벡터를 planeNormal 평면에 투영한 벡터를 구합니다. 특정 축을 무시한 방향을 구할 때도 사용됩니다.

- Vector3.Reflect(Vector3 inDirection, Vector3 inNormal)
: inNormal이 특정 평면의 법선 벡터(normal 벡터) 라고 할때, 해당 평면에 반사된 벡터를 구합니다.



TRANSFORM

Vector3.up/right/forward 처럼 Transform에도 transform.up/right/forward가 존재합니다. Vector3와는 다르게, 해당 Transform 기준의 위/오른쪽/앞쪽 방향 벡터를 반환합니다.

이를 이용해서 앞쪽으로 계속 이동하는 오브젝트를 만들 수 있습니다. 현재 회전값을 기준으로 이동하게 됩니다.

```
void Update()
{
    transform.Translate(transform.forward * 8f * Time.deltaTime);
}
```

위 코드는 앞쪽으로 초당 8만큼 움직입니다. Transform의 메서드들과 Time.deltaTime에 대해서는 마이크로 게임 프로젝트인 5장에서 설명해 드리겠습니다.

마치며

궁금한 점은 꼭 질문해주세요!

카카오톡 | 디스코드

sigening | Sigening#6088