--------------------------------------------------------------------------------------------------------------

벡터 수학 : Vector3로 위치 ,방향,회전,속도,크기 등등을 계산함

2d는 (X,Y) 3D는 (x,y,z)를 사용

벡터를 사용할때는 상대좌표와 절대좌표가 존재

상대좌표 : 현재 좌표기준에서 이동

절대좌표 : 게임내 좌표로 이동

벡터에서의 크기는 피타고라스의 정리를 사용하여 구함(음수가 될수없음), 방향과 크기는 별개

벡터에서의 크기는 속력을 의미

속도와 속력의 차이점

속도는 크기뿐만 아니라 방향까지 의미, 속력은 벡터에서 스칼라값으로 의미되는 크기값

스칼라 : 벡터에 숫자를 곱하면 그 숫자만큼 속도가 증가

벡터로 방향과 속도를 구하면 속도가 정확히 몇인지 표기 하기 쉽지 않기 때문에

1로 정규화된 벡터값을 사용후 스칼라값을 곱하여 속도를 표현하곤함

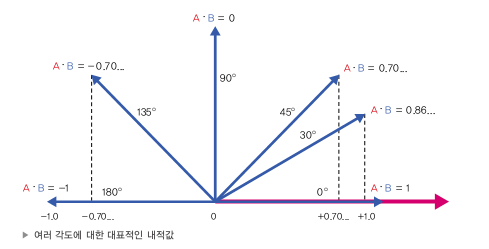
이때 1로 정규화된 벡터값을 방향백터라고 표현(한변에 0.71,0.71인 삼각형)

백터간에 덧셈과 뺄셈이 가능 :

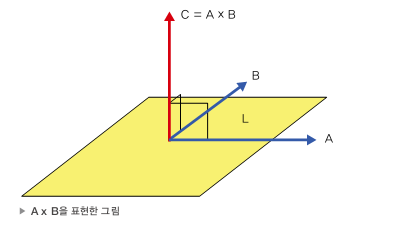
덧셈은 이동한다음 다시 이동하는것을 나타냄.

뺄셈은 두 벡터간의 간격을 의미(즉 현재 위치에서 목적지까지의 방향과 거리)

백터 내적 서로 투영되는 값



벡터 외적 : 다른값에 수직인값



연산 결과가 숫자인 내적과 달리 외적은 연산 결과가 백터이다.

사용이유 : 내적 : 물체가읜의 간격이 얼마나 벌어졌는지, 플에이어의 시선방향과 플레이어가 실제로 이동한 방향사아의 각도가 얼마나 벌여졌는지 등을 파악가능

외적 : 어떤 표면에 수직인 방향을 알수 있음

Vector3연산

단순히 위치를 구한 것이 아니라 방향,속력또한 표현되기에 물체들의 포지션 값도 vector로 선언

벡터는 클래스가 아니라 구조체를 가지기 때문에 참조가 아닌 값타입

스칼라 곱 벡터에 배수를 취함

Vector3 a = new Vector3( 3,6,9);

a = a \* 10; // 10배의 속도를 가짐

덧셈 뺄셈 vector 값끼리 더하거나 빼면됨

정규화

Vector3 a = new Vector3(3,3,3);

Vector3 b = a.normalized;

벡터의 크기

Vector3.magnitude;

Vector3 a = new Vector3(3,3,3);

float b = a.magnitude;

벡터의 내적

Vector3.Dot(a,b);

벡터의 외적

Vector3.Cross(a,b);

물체 사이의 거리 구하는법

Vector3 a = new Vector3(1,1,1);

Vector3 b = new Vector3(3,3,3);

Vector3 c = b-a;

float distance = c.magnitude;

or

float distance = Vector3.Distance(b,a)

현재 위치에서 목적지로 향하는 방향을 원하는 만큼 이동

방향을 구하는방법은 목적지에서 현재위치를 빼는것

Vector3 direction = (목적지-현재위치).normalized;

이동

Vector3 newPos = 현재위치 + direction\* 스칼라값

쿼터니언 :벡터가 속도와 방향을 나타낸다면 쿼터니언은 회전을 나타내는 타입

유니티에서 transform.rotation에 값을 주입할때 transform.postion과 transform,localScale과는다르게 벡터값을 넣을 수 없고 Quaternion.Euler(new Vector()) 형태로 넣어야한다.

벡터를 사용하지 않고 쿼터니언을 사용하는이유:

물체를 회전할때 오릴러각 체계에서는 회전을 순서대로 나누어 실행하는데 실행순서에 따라 앞의 회전에 다음 회전이 영향을받아

자유도가 침범당하는 현상이 일어나고, 이런 현상을 짐벌락 형상이라고 함.

짐벌락을 예방하기 위해 벡터대신 짐벌락을 사용

쿼터니언은 한번에 회전하는 값이므로 짐벌락 현상이 없음

유니티에서는 쿼터니언의 직접적인 생성을 막아놓고, 벡터를 사용하여 쿼터니언을 생성하는 메소드 제공

쿼터니언은 원래 4가지의 변수를 가지고 있으나 벡터와 마찬가지로 3개의 변수만을 사용

Quaternion rotation = Quaternion.Euler(new Vector(0 , 60, 30)); 과 값은 모양으로 사용

쿼터니언 타입을 오일러각 Vector값으로 변환해주는 메소드

Quaternion rotation = Quaternion.Euler(new Vector(0 , 60, 30)); 과 값은 모양으로 사용

Vector3 eulerRoation = rotation.eulerAngles;

쿼터니언에서는 더 회전하는 값을 구할때 더하기가 아니라 곱을 사용, 행렬을 이용하여 이동하기때문에