Vector3 Dot Cross 常见用法

作者：xu365082218
原文链接：https://ld246.com/article/1515962868867
来源网站：链滴

以下都是个人理解。

Dot计算2个向量的点乘法,一般的几何意义是计算力在方向上做的攻，或者一个向量在另外一个向量的投影，Dot(Va, Vb) = |Va| * |Vb| * cos夹角 = Vax * Vbx + Vay * Vby + Vaz * Vbz

他的一般用法有以下几种

1：求夹角是钝角还是锐角

因为Dot的结果在2个向量均为单位向量时，其结果就为cos夹角，当锐角时，值大于0，钝角时，值与0

再细分一点

2：求敌人是否可见

观察流星.net，在角色正面夹角75度以内且距离140码以内敌方，会成为自动目标，这里的夹角75度就是用Vector3.Dot计算的

设主角的面向单位向量为Vector3 forward

设主角的位置向量为Vector3 playerPos;

设敌方的位置向量为Vector3 enemyPos;

则有

float angler = Vector3.Dot(Vecotr3.Normalize(playerPos - enemy.Pos), forward)

float degree = Mathf.Acos(angler);

这个degree就是对应的弧度反cos得到的角度。由于angler的取值范围受到cos值域的影响(-1, 1)，deree的范围也会受到影响(0, 180)

3：求攻击者在我的何方

判断锐角钝角很简单，但是还有遇见其他情况的时候

受击动作，在P0.pos文件里，每个攻击定义都包含4个方向的受击动作，每个动作都不同，想象把整空间划分为前方，左方，右方，后方，每个占90度那么

前方，（-45，45） 左方-135，-90，右方 45，135，后方 （135，180） （-180，-135）

这个是以受击者为主体，攻击者在受击者的某个方位来算的。

根据前面2计算的角度，只能得到一个0-180度以内的角， 这个夹角，是绕着2个向量形成的平面的法作为轴旋转的夹角，这意味着，绕着这个轴顺逆旋转都可以达到一个角度,

那么一般只能判断夹角是否为钝角，那怎么判断攻击者在我的左方，或者右方,就需要叉乘

叉乘成为向量积，其几何意义是计算结果是一个向量，其结果的模长，为2个向量组成的平行四边形面积

Vector3.Cross计算2个向量形成的面法向量，其计算结果是一个向量,且不满足交换律 cross(a,b) != coss(b, a)这是因为这个法线有2个方向，ab形成的平面把空间划分为2个部分

法线垂直ab形成的平面贯穿2个空间，法线是向量，具有方向，所以这时候2个方向都可以选，选定任一个为法向量n

绕法向量n顺时针旋转得到的角度为正方向，以A向量为起点，绕法向量n顺时针旋转到B向量，那么角从0开始增大，一直到A向量与B向量的夹角

绕法向量n逆时针旋转得到的角度为负方向，以B向量为起点，绕法向量n逆时针旋转到A向量，那么角从0开始减小，一直到B向量与A向量的夹角

Cross(Va, Vb) = new Vector3(VayVbz - Vaz Vby, Vaz * Vbx - Vax * Vbz, Vax * Vby - Vay * Vbx)

|Va| * |Vb| * sin（夹角）

Cross(Vb, Va) = |Va| x| Vb| x sin（-夹角）

这里，夹角在0-180间时，b在a的右侧，叉乘结果为正数，sin夹角 = x分量 / r半径 x分量为正

夹角在-180，0之间时，b在a的左侧，叉乘结果为负数，sin夹角 = x分量 / r半径 x分量为负

在这个求方位的例子里，只要点乘夹角<145度，且 [叉乘>0则在右侧，叉乘<0则在左侧]，就可以判出全部方位

那么知道这些结论后，即可推出游戏中2个角色间的位置关系。