```
 private void TestDot(Vector3 a, Vector3 b)
    {
        // 计算 a、b 点积结果
        float result = Vector3.Dot(a, b);

        // 通过向量直接获取两个向量的夹角（默认为 角度）， 此方法范围 [0 - 180]
        float angle = Vector3.Angle(a, b);

        // 计算 a、b 单位向量的点积,得到夹角余弦
值,|a.normalized|*|b.normalized|=1;
        result = Vector3.Dot(a.normalized, b.normalized);
        // 通过反余弦函数获取 向量 a、b 夹角（默认为 弧度）
        float radians = Mathf.Acos(result);
        // 将弧度转换为 角度
        angle = radians * Mathf.Rad2Deg;
    }
```

叉乘

```
 private void TestCross(Vector3 a, Vector3 b)
    {
        //计算向量 a、b 的叉积，结果为 向量
        Vector3 c = Vector3.Cross(a, b);

        // 通过反正弦函数获取向量 a、b 夹角（默认为弧度）
        float radians = Mathf.Asin(Vector3.Distance(Vector3.zero,
Vector3.Cross(a.normalized, b.normalized)));
        float angle = radians * Mathf.Rad2Deg;

        // 判断顺时针、逆时针方向，是在 2D 平面内的，所以需指定一个平面，
        //下面以X、Z轴组成的平面为例 ， (Y 轴为纵轴),
        // 在 X、Z 轴平面上，判断 b 在 a 的顺时针或者逆时针方向,
        if (c.y > 0)
```

```
        {
            // b 在 a 的顺时针方向
        }
        else if (c.y == 0)
        {
            // b 和 a 方向相同（平行）
        }
        else
        {
            // b 在 a 的逆时针方向
        }
    }
```

1.根据叉乘得到a，b向量的相对位置，和顺时针或逆时针方位。

简单的说：点乘判断角度，叉乘判断方向。　形象的说：当一个敌人在你身后的时候，叉乘可以判断你是往左转还是往右转更好的转向敌人，点乘得到你当前的面朝向的方向和你到敌人的方向的所成的角度大小。

2.得到a，b夹角的正弦值，计算向量的夹角（0,90），可以配合点乘和Angle方法计算出含正负的方向。

3.根据叉乘大小，得到a，b向量所形成的平行四边形的面积大小，根据面积大小得到向量的相对大小。