点积的用法-角色控制trick

1.角色控制爬坡

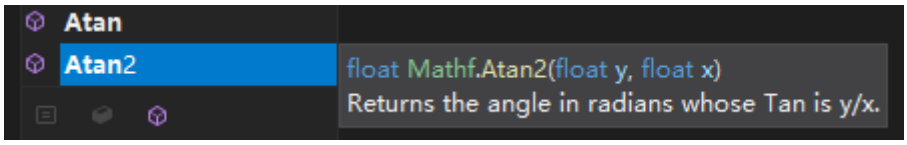
最近看到一段角色爬坡的控制代码,很简短，写的很好。

“talk is cheap,show me your code”，所以直接上代码和原理图了。

```
    public static Vector3 GetDirectionAlongNormal(Vector3 moveDirection, Vector3
normal)
    {
        //flip coordinate
        Vector3 direction = new Vector3(normal.y, -normal.x);
        float dirDot = Vector3.Dot(direction, moveDirection);
        if (dirDot < 0)
        {
            direction *= -1.0f;
        }
        return direction.normalized;
    }
```

2.角色在斜面上旋转-----Atan2(float y,float x);

当角色在斜面上时，你可能需要将角色旋转至垂直于斜面，这样的作法有很多**atan,acos,asin,dot,**都可以实现，这里我想用Atan2，因为以前用基本是笔试，面试的时侯强行用.....,这次可算找到了一个我觉得比较适合的用法。



就像图中说的,Atan2就是tan(y/x),但是精度比tan(y/x)高，将**斜面法线**带入进去**Mathf.Atan2(normal.y,normal.x)*Mathf.Rad2Deg**,便可以**计算出墙于地面的夹角**，注意这里我写的不是旋转角度，而只是夹角，实际上当你去看GitHub上的库的时候，你会发现是这样写的

-Mathf.Atan2(up.x, up.y) * Mathf.Rad2Deg;

其实这是坐标系选择的问题，atan2是默认的坐标系，是我们上数学课的坐标系x轴正向朝右，y轴正向朝上，旋转以x轴为起点逆时针旋转至y轴，而当我们将y和x，颠倒代入时，相当于翻转了坐标系，但仍然是从x轴朝y轴旋转

坐标系选择不同引入的问题

上面的例子中我想强调的是，**在做任何公式推导前，很重要的一点是坐标系确认的问题**，不知道大家有没有这样的经历，每次过亿段时间，去推导旋转矩阵的时候总是发现，和别人推到的结果不一样，但是你自己的理解又是正确的，用起来也没什么**大问题**，那是因为**别人的旋转可能是从y轴正半轴顺时针旋转，而你的不是！**同理，cos，sin，还有其他的函数也会有异曲同工之处(有点抽象了...)

小结

sin，cos，atan2，dot，cross，这几个函数真的是奥妙无穷，字面意思看起来很容易理解，但实际运用起来是有非常多的细节在里面的，渺沧海之一粟矣。