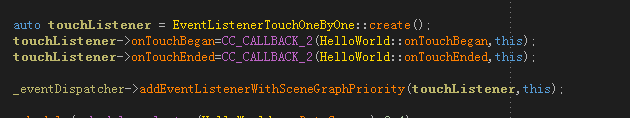
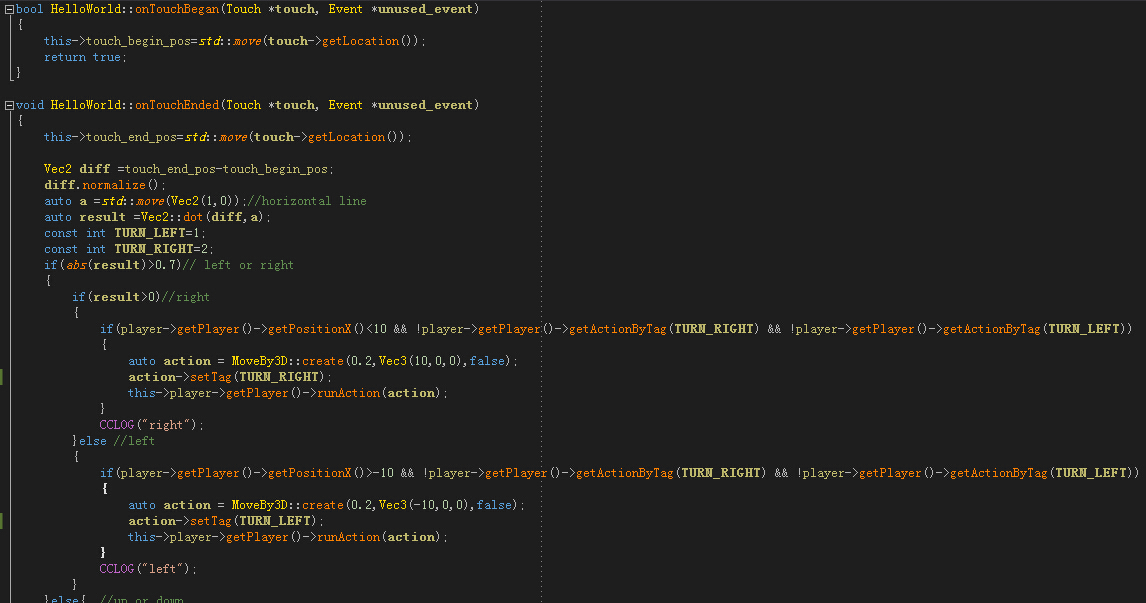
在这张教程中，我们主要是添加一些地面和角色的移动

在这个demo中，我们对跑酷主角的移动进行了一些简化，只允许它向右和向左移动。实现的方式非常简单，首先我们对当前的Layer的touch事件进行监听：

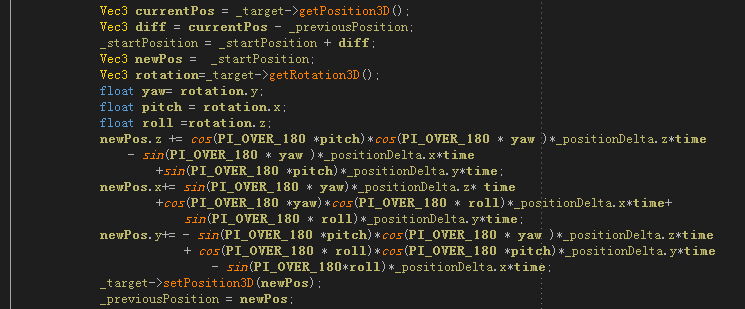


这里，我们主要关心的是触摸开始和结束的时候，触摸的位置。

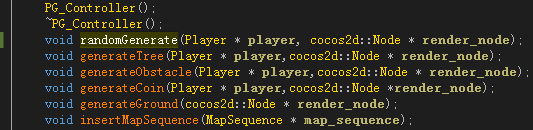


这个方法用开始触摸和结束位置的向量，与水平线的向量的点积（即两线夹角的cos值），来判断用户是往左滑还是往右滑，这个方式要用两个坐标值来比对要精准一些，因为用户拉出来的线条通常都并不是横平竖直的。

当确定了方向之后，我们就要给主角一个action，让它能向左或者向右移动，在run这个Action之前，我们要首先判断，主角是否应在执行action了，在同一时刻，我们只允许用户执行一个运动，这是为了，不让用户移除我们设定的边界，以及不让用户连续滑动屏幕时速度，主角的移动的速度也突然加快。在这里我们使用了一个自己定义的Action——MoveBy3D，这个Action提供了针对Sprite3D以全局坐标系或以本地坐标系为基准的运动。如果有兴趣的话，可以看看这个Action在Update处的实现，在这里就不统一描述了



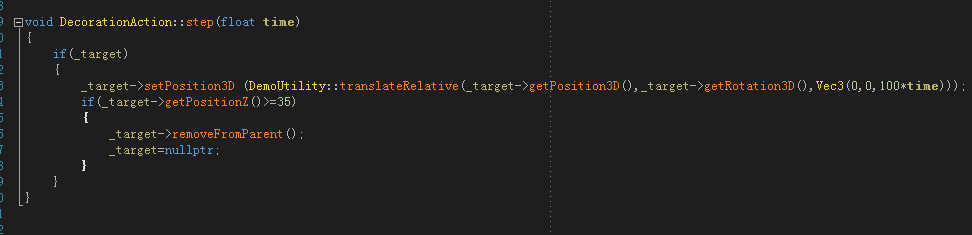
现在，我们来看看，如何显示地面以及一些树，在这个demo里，树和地面，以及之后会讲到的金币还有障碍物等等，都是过程生成的（procedure generate），也就是说是在程序内部自动的生成出整个场景来，而不用预先指定，当然,不可能所有的东西都过程生成，过程生成最小的单位通常被称为sequence，或者pattern这些是不可分割的，在我们demo目前的情况来说，还不需要考虑这些问题，因为树和地面是一直出现的，也就是生成他们的pattern非常的简单，我们来看看过程生成的控制器部分。



现在我们只看其中的randomGenerate、generateGround以及generateTree,

randomGenerate是一个总控函数，它用来控制何时应该生成什么，generateGround以及generateTree分别生成地面和树。

树和主角一样是一个Sprite3D，只不过不同的是，在它被销毁之前就会一直运行一个action，这个action让它不断地靠近相机，当照相机背面一定距离的时候，就将其删除



类似的方法，我们可以用来实现地面效果，目前，地面是一张2D贴图，我们将其按X轴旋转90度，就可以得到地面的效果了，注意，请不要忘记也将在添加完地面之后设置一下CameraMask。最后我们就得到了最终的效果图：

