**Unity3D的Waypoint(路点)**

Posted on 2013年04月08日 by U3d / [Unity3D 基础教程](http://www.unitymanual.com/category/manual/unity3d-%e5%9f%ba%e7%a1%80%e6%95%99%e7%a8%8b)/被围观 199 次

**Unity3D**的Waypoint(路点) 。[**Unity3D**](http://www.unitymanual.com)官方网站上有个非常好的例子，能够“完美”解决路点的需求。以下是没有使用Quaternion （四元数）这个高深的类做的算法简化，使用的是更加方便的Vector3.

实现分为两步。1：路点间的直线行走。2：解决突然转向的问题，让转向平滑。

解决方案1：

每个Waypoint是一个trigger。角色仅仅向着自己的target方向前进，就是z轴的正方向。运动的方法如下：

var fMoveSpeed : float = 4.0; // 角色的移动速度

function Update()

{

transform.Translate( 0, 0, fMoveSpeed );

}

多个Waypoint之间的连线形成一条路，存储所有的Waypoint需要用到数组

var aWaypoint : Transform[]; // 路点数组

将JS代码作为组件给角色。用gameObject的Cube作为路点。先在Scene面板内创建所有路点。然后依序拖入角色的 aWaypoint 变量内，这时此变量的长度会自增。拖动的顺序必须是路点的顺序，所以可以使用更方便的方法，在变量下面有个Unity提供的变量Size，手动填入路点数量。这时Unity会在下面直接创建填入数量的列表。点击每行最后面的圆圈。会弹出一个面板“Select Transform”，这里有本场景内的所有物件，如果路点的名字比较给力那么和数组元素一一对应添加即可。为了让路点数量统一，在代码里需要有个数组长度变量，以便和这个长度相同，放置造成数组越界。

var nWaypointNum : int = 0; // 路点的数量

每个路点设置为Trigger。物件添加Rigibody属性。然后扩充刚才的代码

var nWaypointNum : int;

var aWaypoint : Transform[];

var fMoveSpeed : float;

private var nTargetWaypoint : int; //下一个路点的索引

function Start()

{

nTargetWaypoint = 0;

}

function Update()

{

// 这里使用了一个技巧，就是通过向量减法，算出朝向下一个路点的方向向量。

transform.forward = aWaypoint[nTargetWaypoint].position - transform.position; // 朝自己的z方向移动。 transform.Translate( 0, 0, fMoveSpeed \* Time.deltaTime );

}

function OnTriggerEnter( go : Collider )

{

nTargetWaypoint++;

Debug.Log(nTargetWaypoint);

if( nTargetWaypoint >= 4 ) { nTargetWaypoint = 0; }

}

解决方案2：

通过使用Vector3类提供的函数来产生平滑的差值。

static function Slerp (from : Vector3, to : Vector3, t : float) : Vector3

from------起始向量

to------目标向量

t------取值在0到1之间，如果为0计算出来的向量就和from向量重合，如果为1则和to向量重合。所以这个参数说的直白些就是一个偏向值。（根据测试的实际情况来看，如果t值大于1，则为1，所以当累计便宜大于1也不会出现s型运动）

为了让角色有个转向动画，那么必须让最后的t值每帧都发生改变。 Time.deltaTime 能够获得两帧之间的毫秒时间差，单位为秒。

代码如下：

function Update()

{

// 注意第一个参数是 transform.forward ，这就是为什么前面用z的正方向做朝向原因

transform.forward = Vector3.Slerp( transform.forward, aWaypoint[nTargetWaypoint].position - transform.position, Time.deltaTime \* 10 ); // 朝自己的z方向移动。

transform.Translate( 0, 0, fMoveSpeed \* Time.deltaTime );

}