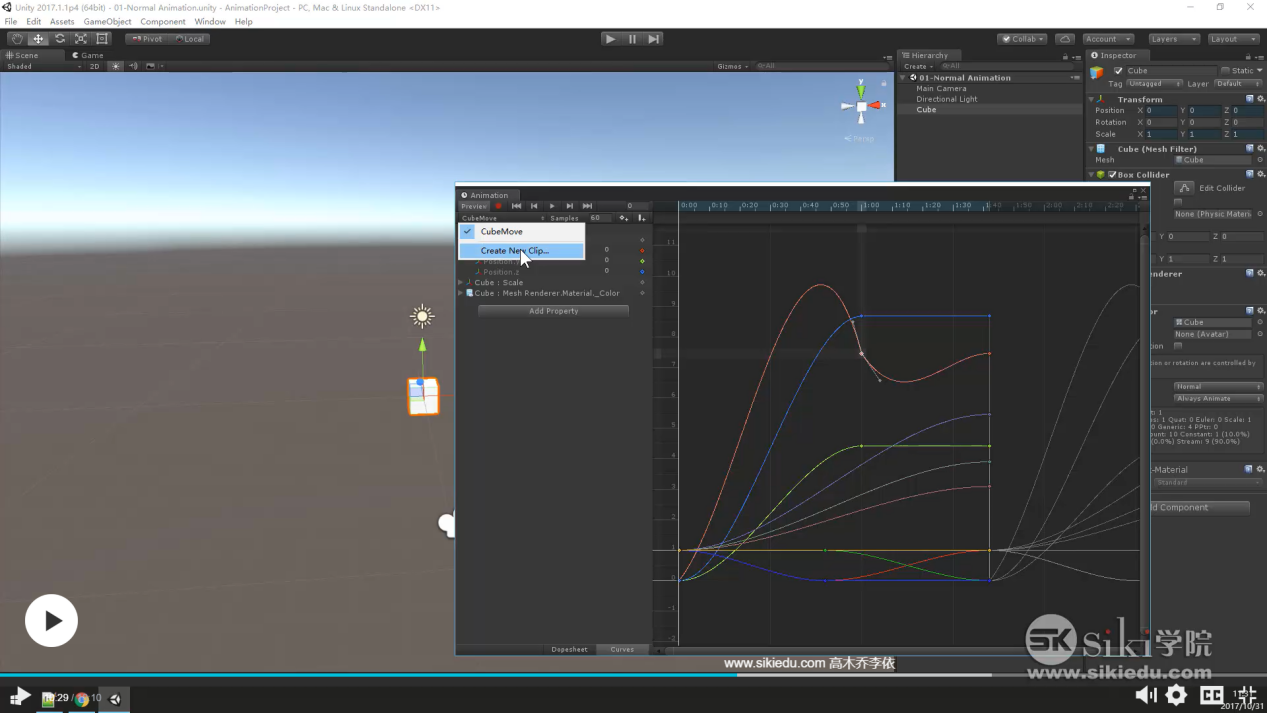
Unity动画 （入门案例学习）

1. windows-animation-选中物体
2. samples 改变每秒帧数 改变运动快慢
3. Create new Clip同一个物体添加多个动画



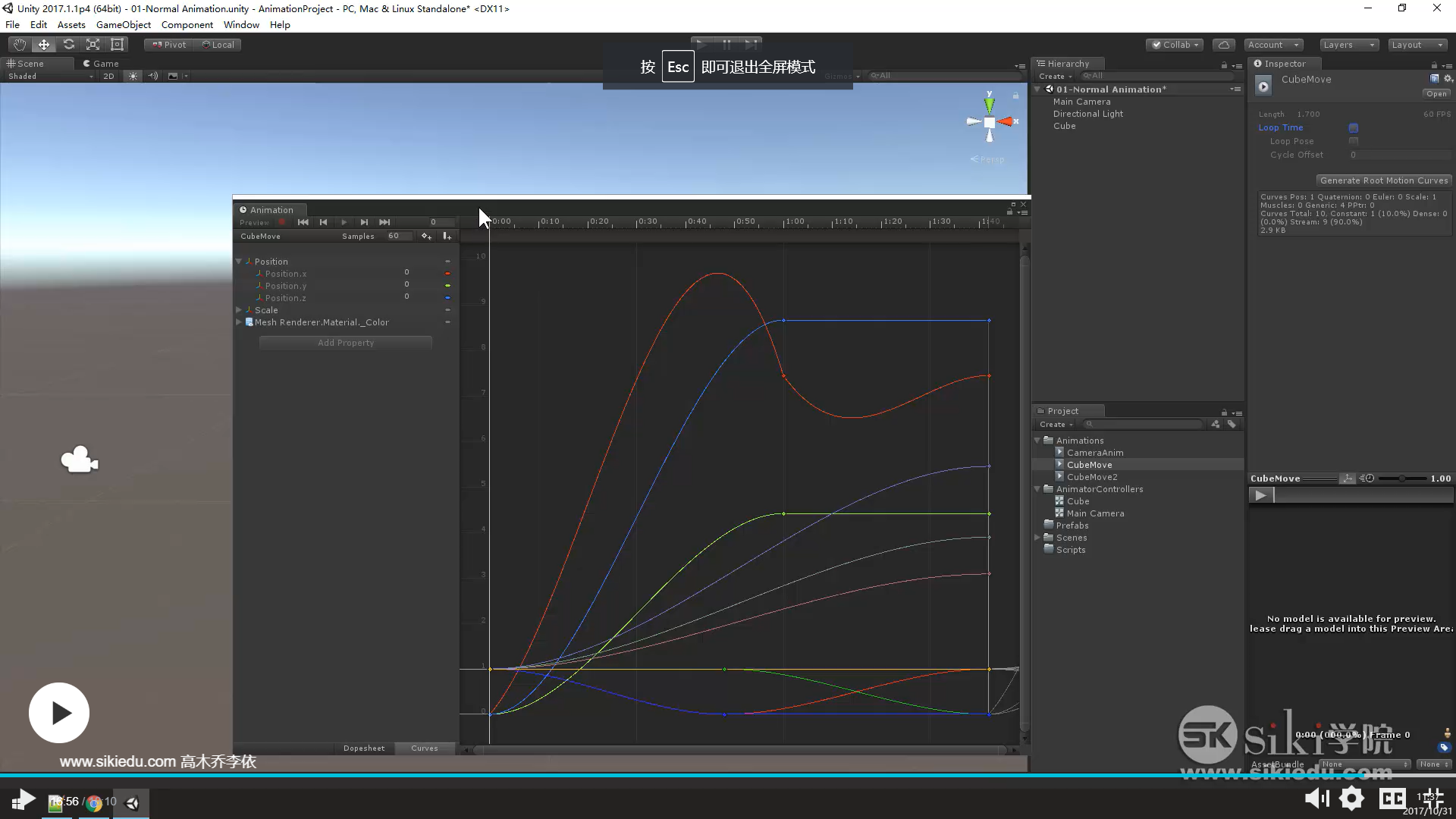
1. 动画方式 1.Transform改变物体位置
2. 添加关键帧如果使用add key会使所有地方都加一个关键帧 如要单独一个属性增加关键帧 直接选中时间点然后在面板界面修改
3. Preview旁边的小红点按完后会根据你当前的改变生成关键帧
4. Dopesheet 关键帧模式

Curves 曲线模式 可以修改属性的曲线

按住shift加鼠标滚轮放大坐标轴

曲线模式下选择分割点右键鼠标Left tangent可以修改成linea和free（consant weight）

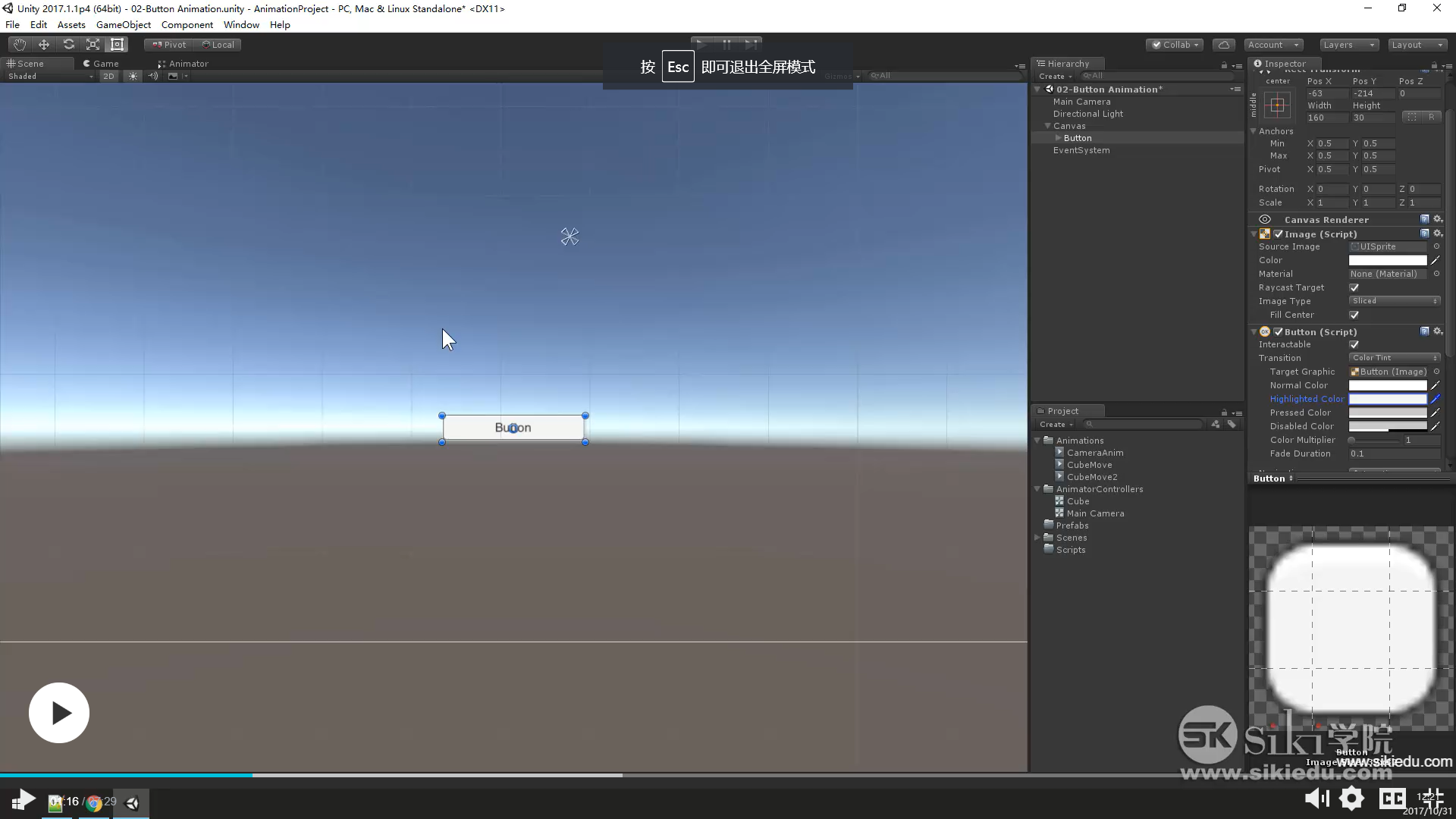
1. 在project里面点击动画inspector面板里面可以选择循环或者不循环 loop time/lose



9. ugui的按钮动画（四个状态）

1.button ->Transition ->animation（动画）/sprite swap（图片）

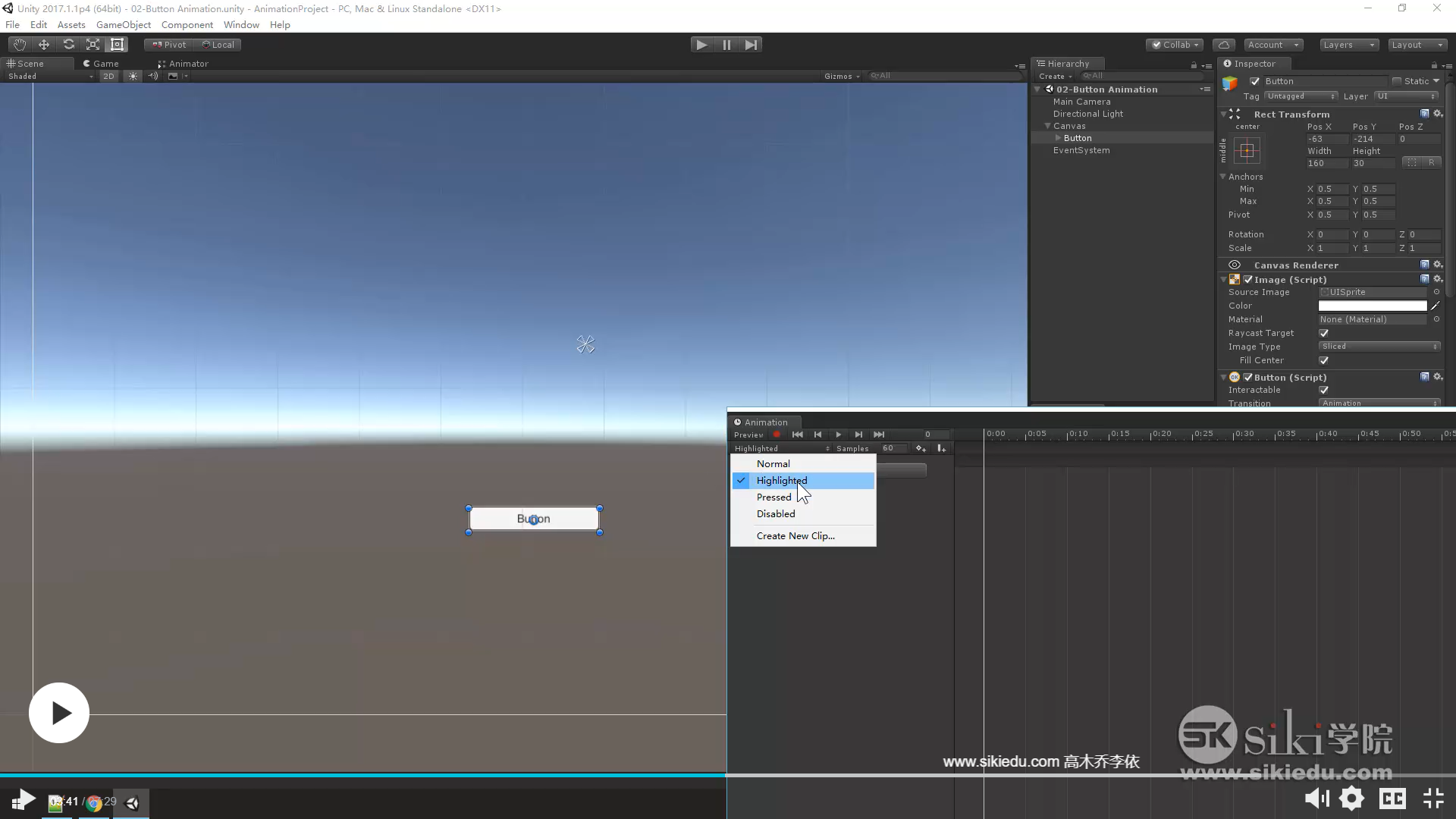
按钮的状态->inspector->button(script)->Interactable 勾选按钮有效反之无效

 如图所示 Interactable下面就是Transition

在选择Transition-animation后

在window-animation里面就可以编辑按钮不同状态的动画

通过红点录制最方便



10. 2d游戏的精灵动画

通过图片播放变成动画

生成方式一：直接将两张图片拖入场景 系统会自动生成动画 修改依旧通过window-animation

生成方式二：手动创建 选择物体->inspector->添加组件animator->将自己创建的animator controller赋值给controller->打开window中的animation 使用sprite格式的动画 直接拖拽图片进行关键帧赋值

10.3d模型

皮肤渲染-skinned Mesh Render

Inspector面板里面（选择模型本身的inspector而不是场景里面的模型）可以修改模型

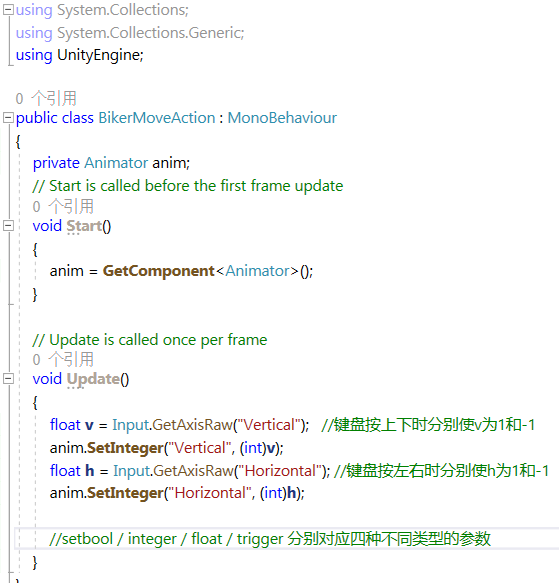
Model可以修改模型 rig修改动画的导入方式->humanoid专门用于人型动画 generic可以用于怪物且两者都支持状态机且不能用animation 旧版本的legacy不能使用状态机，只能使用animation自动播放

11.Animator可以直接通过拖拽动画来使用

首先拖拽进入的是默认动画 后面拖拽的都是灰色的 可以通过make transition来创造连线

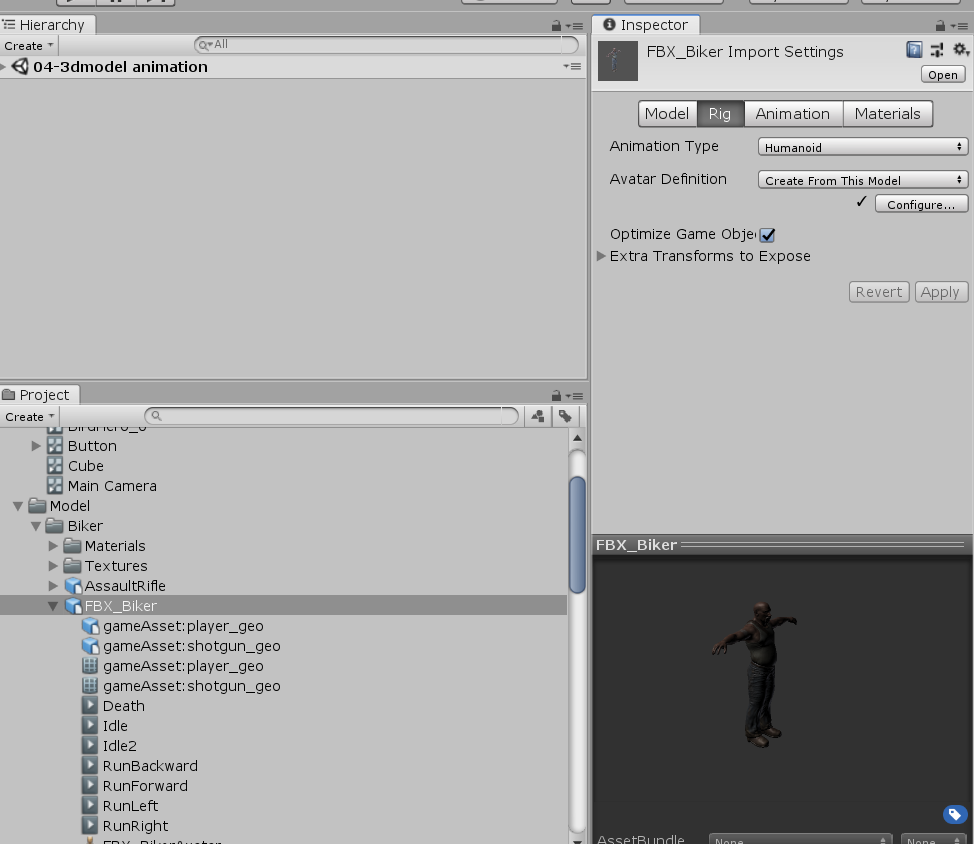
Animator通过创立参数来判断改变物体运动的条件 需通过代码来完善

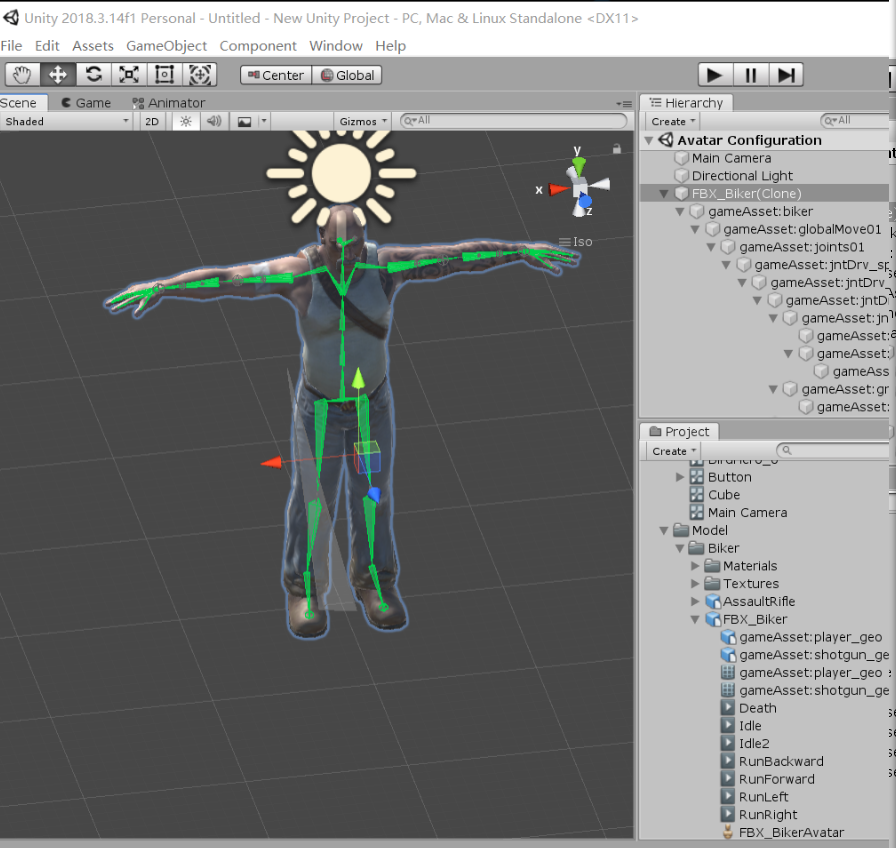
Animator 本身也是一个组件可以通过getcomponent来调用

用到的代码以及解释如下图

12.generic和humanoid的区别 关键点：Humanoid可以直接通过运动控制人物的运动

Generic不能使用别人的动画 把animator给他也不能使其得到别人的动画

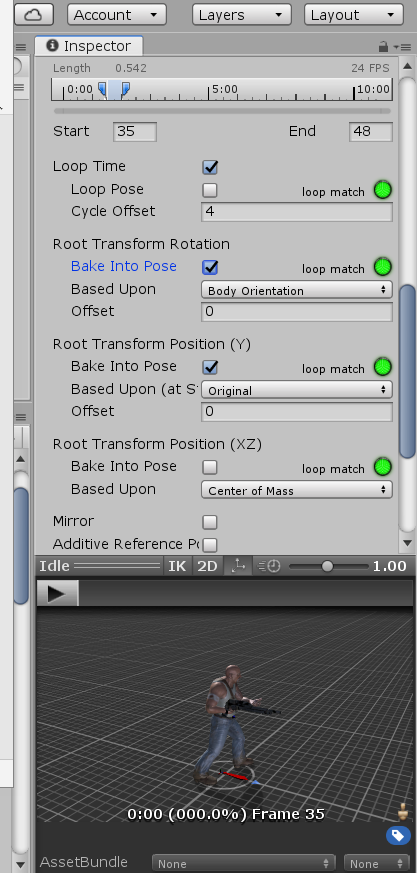


点击右上角configure可以进入下面的界面

在该处可以调节骨骼映射并且测试骨骼的各种姿势

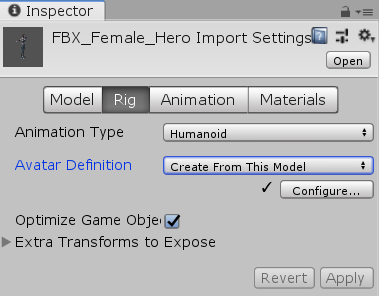
将模型设定为humanoid后 在模型的属性界面选择animation可以查看动画

且其中的bake into pose 选项可以在该动画发生时固定人物的location rotation等

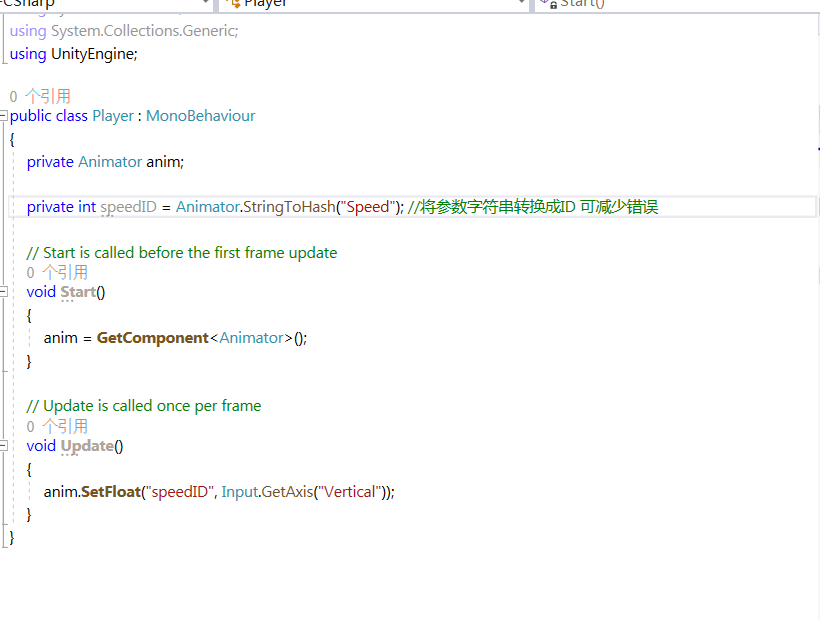


Offset 偏移

Animation Type为Humanoid时使用其他模型的动画，如果骨骼不一样，在inspector的rig界面->Avatar Defintion需选择Create From This Model



13.StringToHash将参数字符串转换成ID

在使用animator的参数的时候要尽量使用这种方式

简单来说

当使用Animator时，如果要引用某个状态或者参数，有两种方法。

1：通过字符串名称

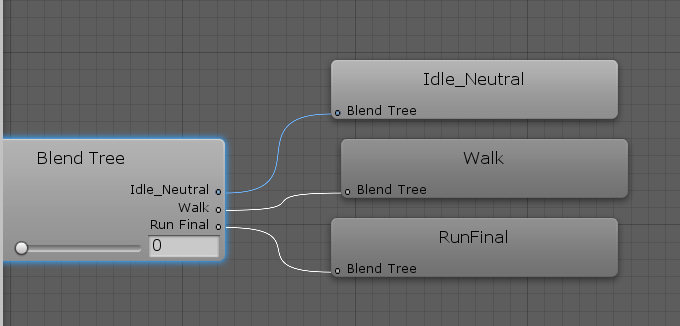
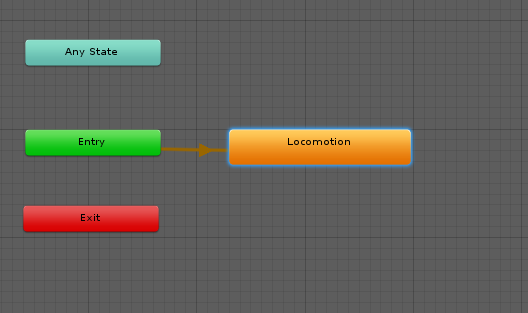
2：通过整数“Hash ID”

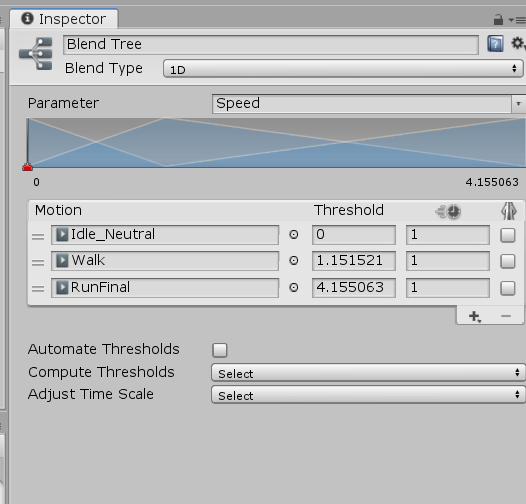
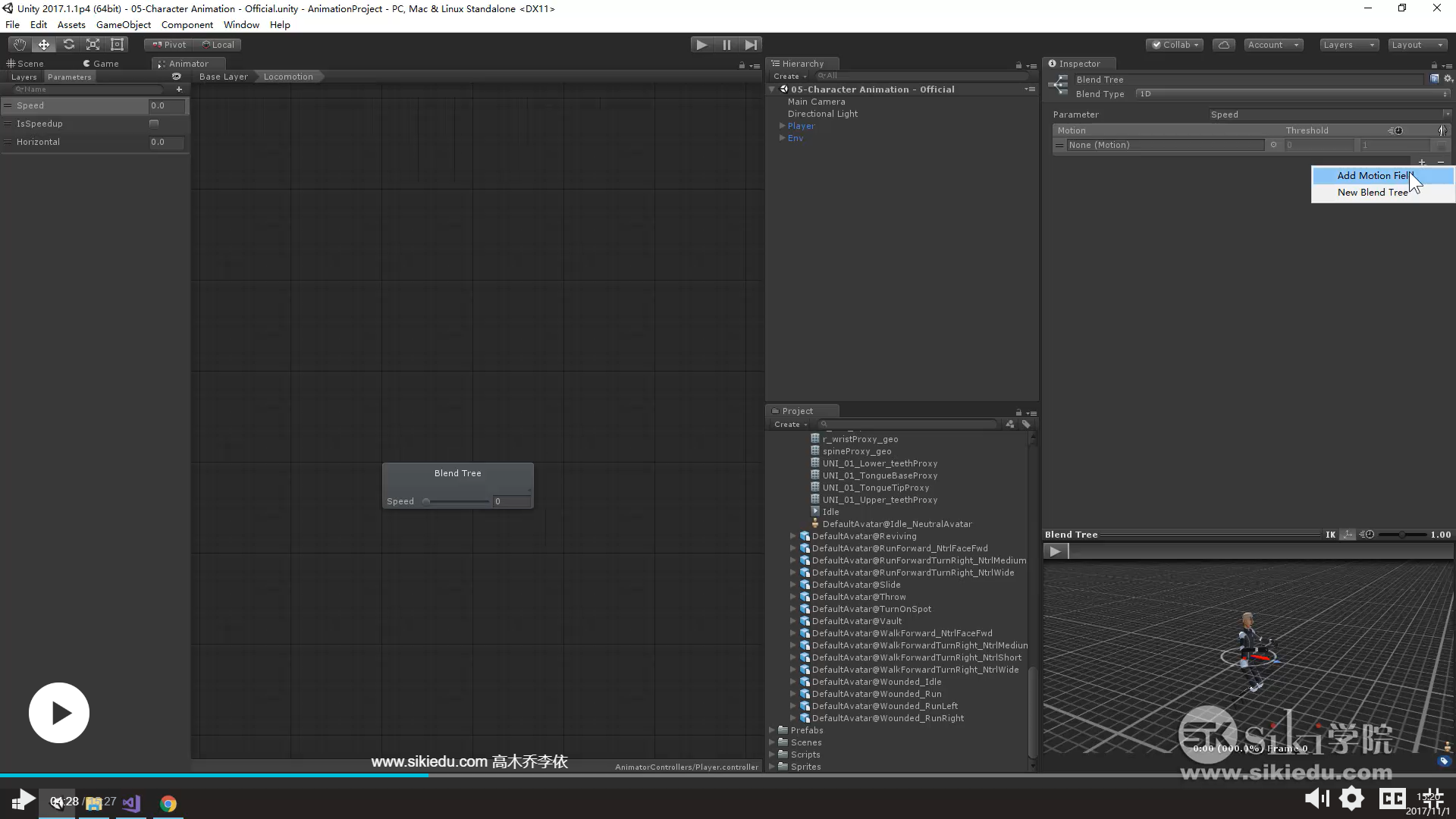
通过“Hash ID”更有优势，因为不容易出错，且更加高效。

两种的使用方式不同。animation状态或者parameters的字符串(Tag)需要通过对象使用；但是“Hash ID”不需要，他是Animator的静态方法。

详见：<https://blog.csdn.net/googtiki/article/details/78636997>

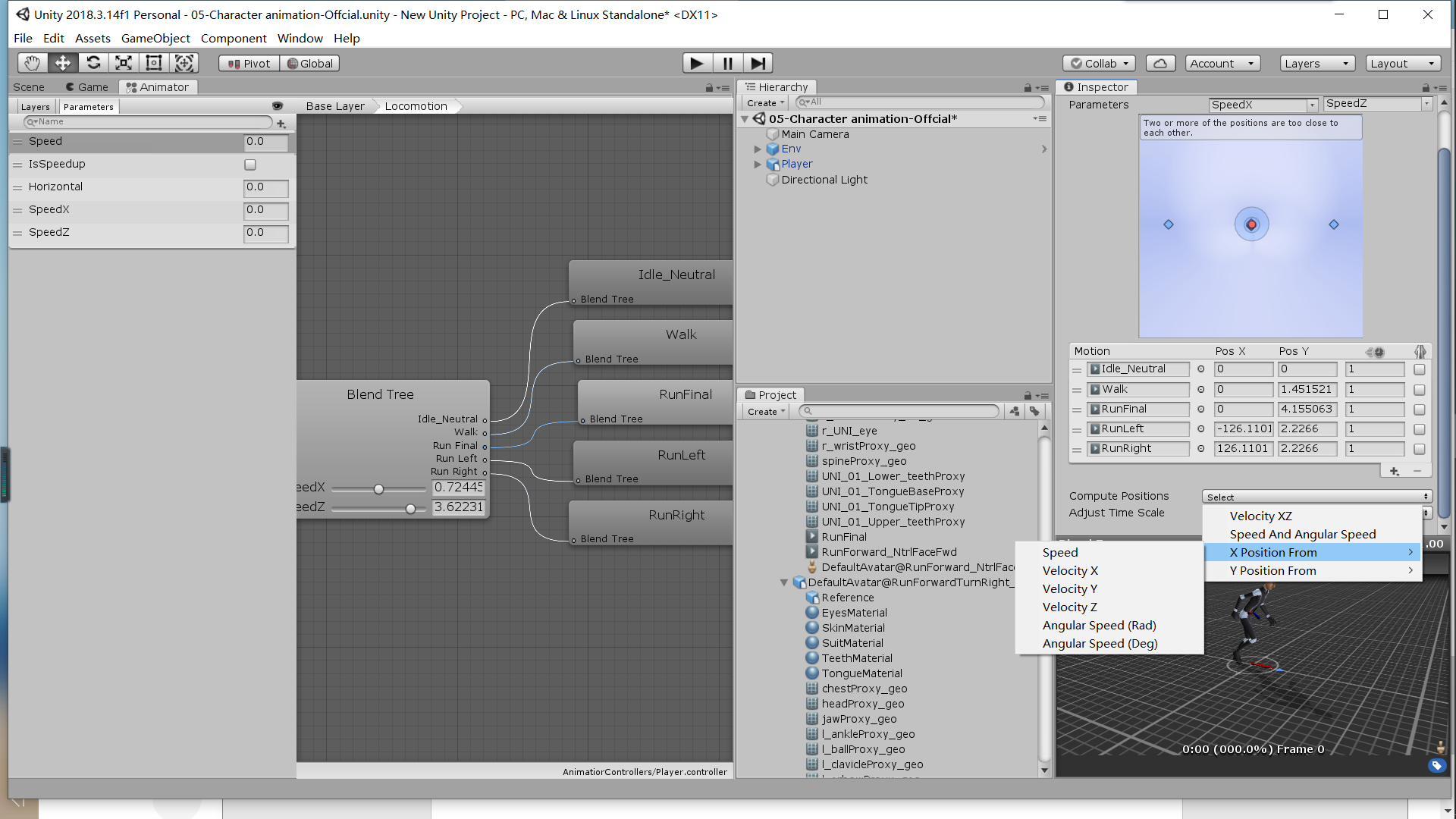
14.混合树 Blend Tree （入门）

在animator中右键创造混合树，利用Add Montion Filed可以在混合树中加入多个动作



上图中的Compute Tresholds意思为计算阈值可以通过不同的参数来计算 只需要一个阈值选择1d 否则2d 3d

2d类型的混合树 见案例动画



阈值的标准可以根据动画本身来随机应变

**案例的转向动画x轴转向较慢但是角速度快 便可以通过角速度来调整**

**反正必须要找出容易区分的标准**

并且阈值本身就是参数 在代码中需要把阈值可能达到的最大值考虑进去使数值能变化到最大范围。5

比如角度最大为126时

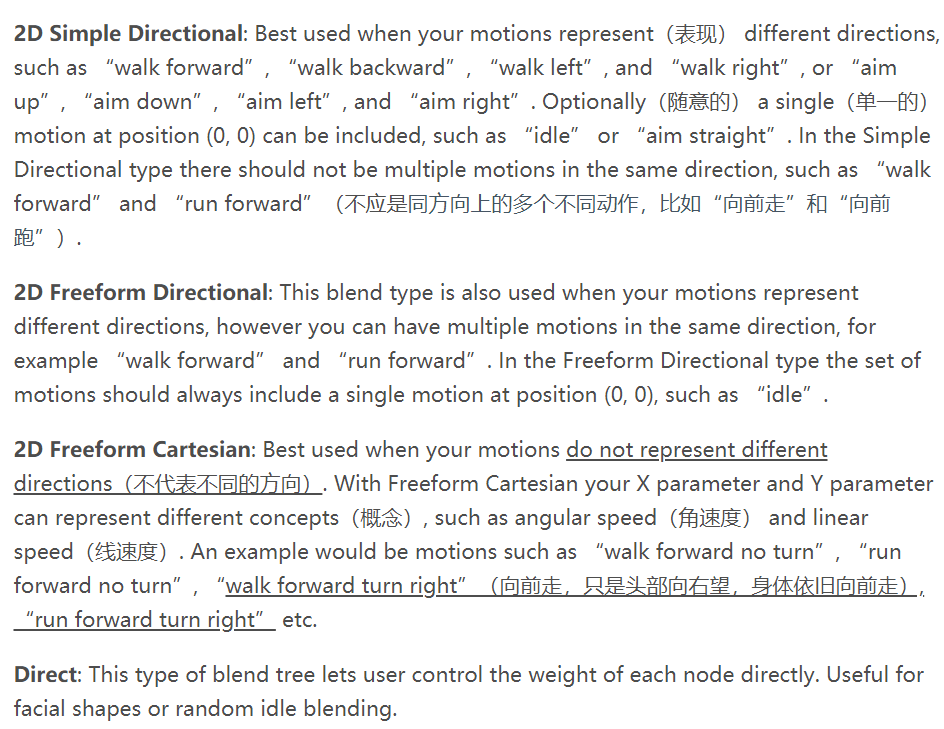
anim.SetFloat(SpeedRotateID, Input.GetAxis("Horizontal") \* 126);

14.关于如何选择树的种类

简单来说 2D Simple Directional 不能容纳同一方向不同运动

2D Freedom Directional 可以容纳但需要有一个一直待在原点的动作

2D Freedom Cartesian 不能表示不同方向的运动

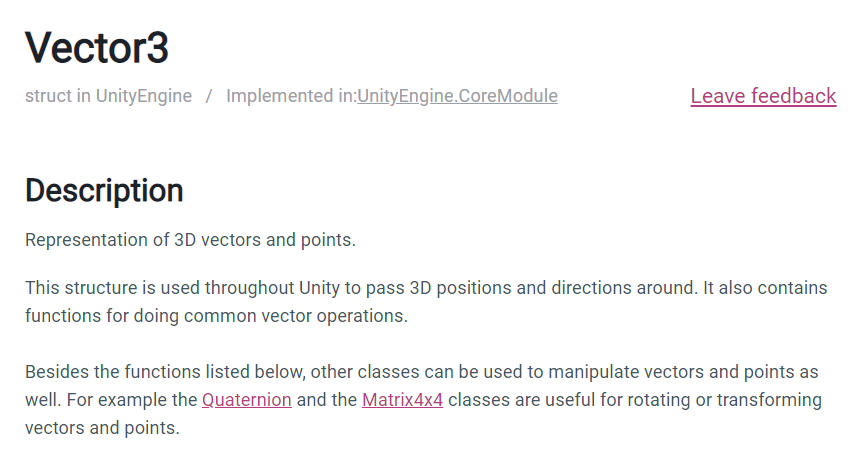


15. 摄像机的跟随

1.关于vector3;

表示三维向量和点。

这种结构被用于整个Unity的三维位置和方向的传递。它还包含执行公共向量运算的函数。

除了下面列出的函数外，还可以使用其他类来操作向量和点。例如，四元数和矩阵4x4类用于旋转或转换向量和点。

2. TransformDirection

# [Transform.TransformDirection 变换方向](https://www.cnblogs.com/living481/p/6285869.html)

官方描述：

JavaScript ⇒ TransformDirection(direction: Vector3): Vector3;   
C# ⇒ Vector3 TransformDirection(Vector3 direction);

#### Description 描述

Transforms direction from local space to world space.

变换方向从局部坐标转换到世界坐标。

This operation is not affected by scale or position of the transform. The returned vector has the same length as direction.

这个操作不会受到变换的缩放和位置的影响。返回的向量与direction有同样的长度。

个人理解：

如自身Z轴方向跟世界Z轴方向一致，目前我的位置是（0，0，0），我向前（相对自身）移动一个单位，我的位置变成（0，0，1），

即为：transform.postion += vector3(0,0,1);

当自身Z轴方向跟世界X轴方向一致时，我的位置还是（0，0，0），我向前（相对自身）移动一个单位，我的位置变成（1，0，0），

即为：transform.postion += transform.TransformDirection(vector3(0,0,1))的结果。



上述代码分析 1.vector3是一个类 既可以用来表示位置也可以表示方向

2.lerp public static Vector3 Lerp(Vector3 a, Vector3 b, float t);

其中a就是图中的from向量，b就是to向量。

t是夹在 [0…1]之间，当t = 0时，返回from，当t = 1时，返回to。当t = 0.5 返回from和to的平均数

<https://blog.csdn.net/Egret_or_Unity/article/details/79485232> //关于lerp函数的理解

16.关于matchtarget

<https://blog.csdn.net/qinyuanpei/article/details/38275323>

1. 定义
2. void MatchTarget (Vector3 matchPosition, Quaternion matchRotation, AvatarTarget targetBodyPart, MatchTargetWeightMask weightMask,

指定匹配的位置、角度、身体的具体部分、权重、匹配开始及匹配结束的时间(先了解)

Eg：**anim.MatchTarget(matchTarget, Quaternion.identity, AvatarTarget.LeftHand, new MatchTargetWeightMask(Vector3.one, 0), 0.32f, 0.4f);**

四元数 quaternion见其他Word(待学习)

17.关于AnimatorStateInfo（先了解 待熟悉）

## Returns

**AnimatorStateInfo** An [AnimatorStateInfo](https://docs.unity3d.com/ScriptReference/AnimatorStateInfo.html) with the information on the current state.

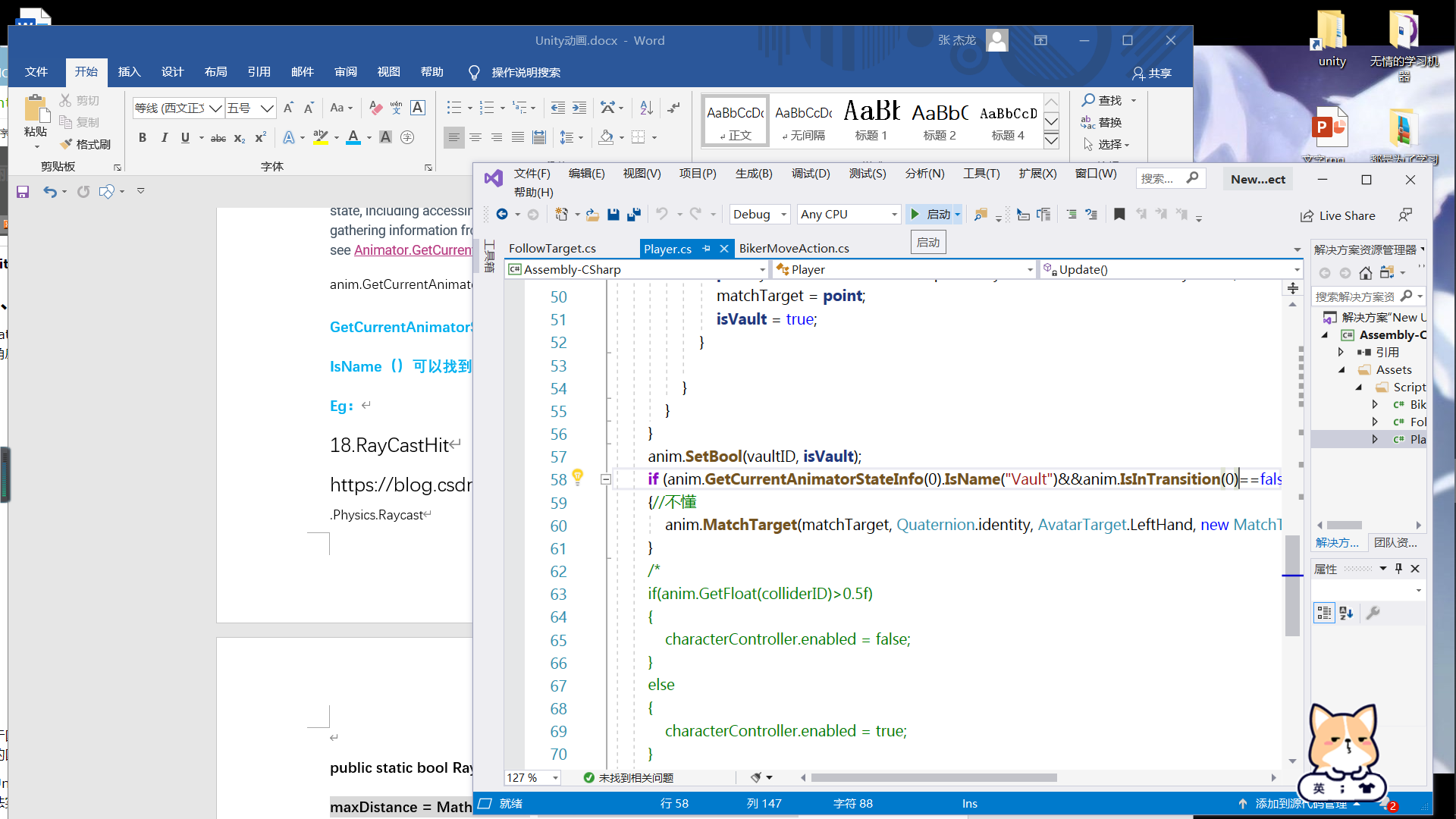
## Description

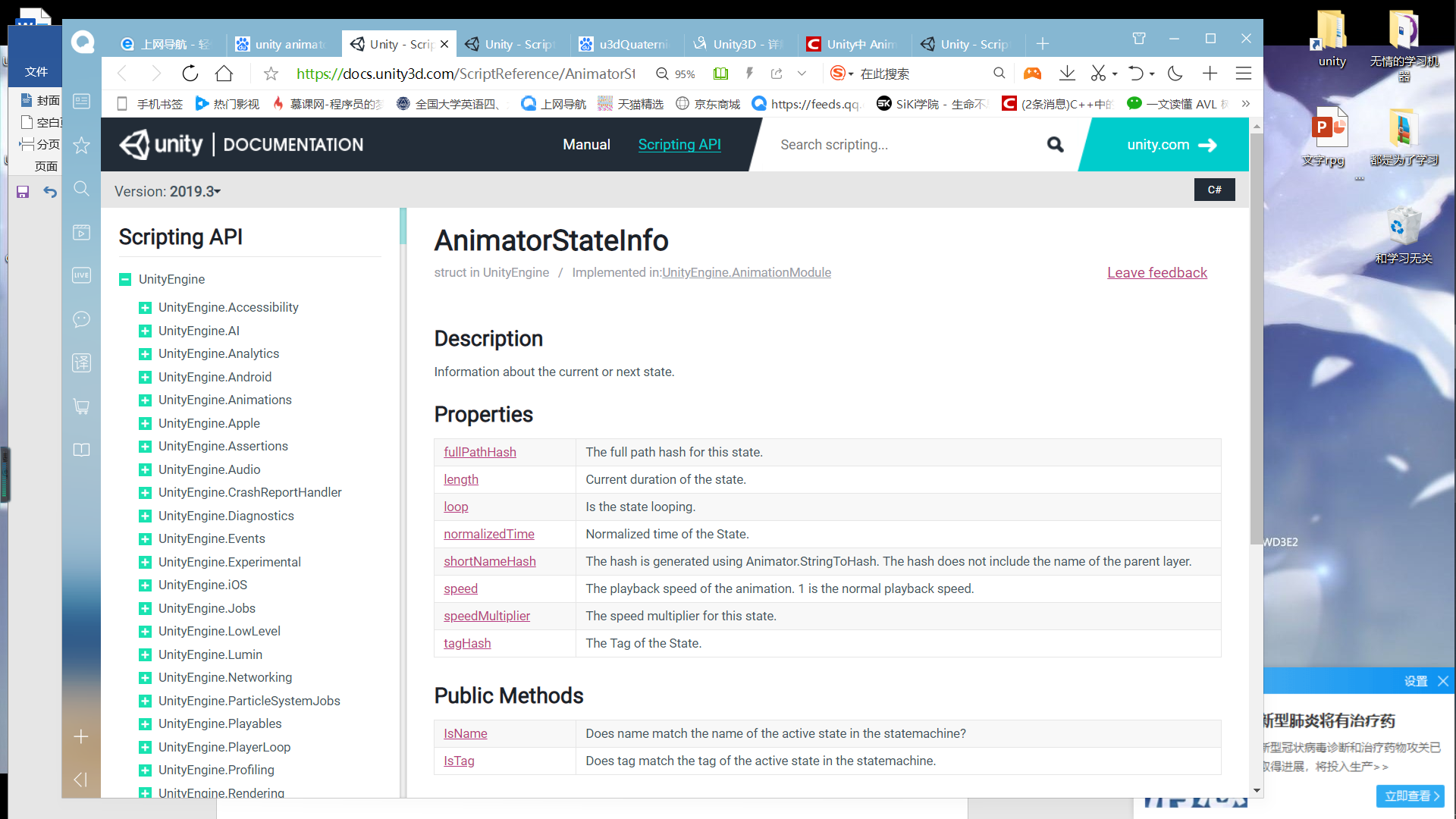
Returns an [AnimatorStateInfo](https://docs.unity3d.com/ScriptReference/AnimatorStateInfo.html) with the information on the current state.

Fetches the data from the current state in the Animator. Use this to get details from the state, including accessing the state’s speed, length, name and other variables. For gathering information from the clips that the states hold, see [Animator.GetCurrentAnimatorClipInfo](https://docs.unity3d.com/ScriptReference/Animator.GetCurrentAnimatorClipInfo.html).

anim.GetCurrentAnimatorStateInfo(0).IsName("Locomotion")

**GetCurrentAnimatorStateInfo先获得一个AnimatorStateInfo对象，后面IsName（）可以找到括号内的动画或者是混合树**

**Eg：**



**18.关于RayCast**

https://blog.csdn.net/L1ght/article/details/76150084

https://blog.csdn.net/qq\_30454411/article/details/79140318

.Physics.Raycast

API文档中只给出这几个方法：

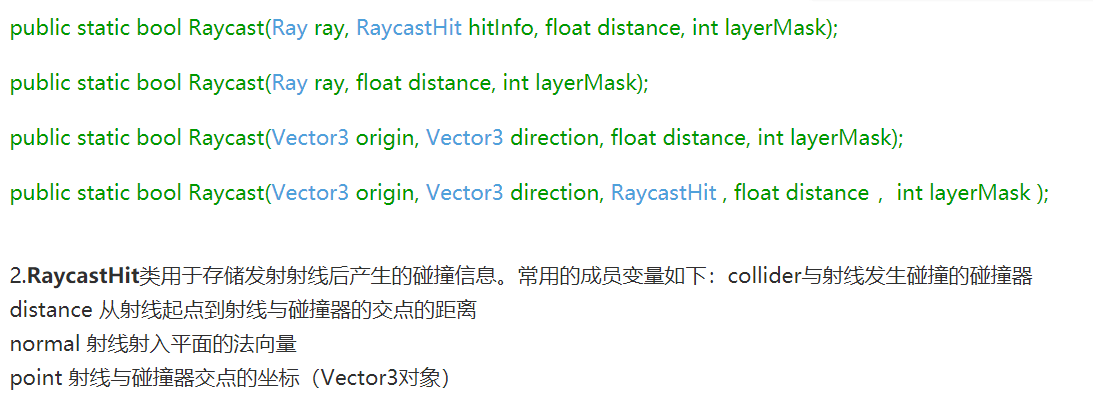
public static bool Raycast([Ray](http://www.unitymanual.com/file:/F:/Soft/unity4.6/Editor/Data/Documentation/html/en/ScriptReference/Ray.html)ray, [RaycastHit](http://www.unitymanual.com/file:/F:/Soft/unity4.6/Editor/Data/Documentation/html/en/ScriptReference/RaycastHit.html)hitInfo, float distance, int layerMask);

public static bool Raycast([Ray](http://www.unitymanual.com/file:/F:/Soft/unity4.6/Editor/Data/Documentation/html/en/ScriptReference/Ray.html)ray, float distance, int layerMask);

public static bool Raycast([Vector3](http://www.unitymanual.com/file:/F:/Soft/unity4.6/Editor/Data/Documentation/html/en/ScriptReference/Vector3.html)origin, [Vector3](http://www.unitymanual.com/file:/F:/Soft/unity4.6/Editor/Data/Documentation/html/en/ScriptReference/Vector3.html)direction, float distance, int layerMask);

public static bool Raycast([Vector3](http://www.unitymanual.com/file:/F:/Soft/unity4.6/Editor/Data/Documentation/html/en/ScriptReference/Vector3.html)origin, [Vector3](http://www.unitymanual.com/file:/F:/Soft/unity4.6/Editor/Data/Documentation/html/en/ScriptReference/Vector3.html)direction, [RaycastHit](http://www.unitymanual.com/file:/F:/Soft/unity4.6/Editor/Data/Documentation/html/en/ScriptReference/RaycastHit.html), float distance ，int layerMask );

分别对着，射线的发射点**Vector3 origin,**，射线的方向**Vector3 direction**，射线的最大距离**float maxDistance = Mathf.Infinity**，检测带不带碰撞器**int layerMask = DefaultRaycastLayers,**，无碰撞器打不着另一个物体

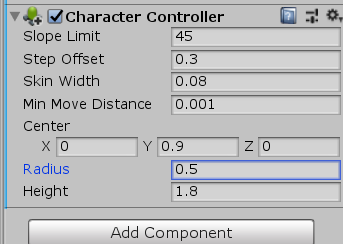
这个函数的意思就是：以射线ray经过的maxDistance长度之内，与第一个对象进行的物理碰撞的信息，存储在hitInfo中；如果有碰撞物体，返回true, 反之false;

19.角色控制器

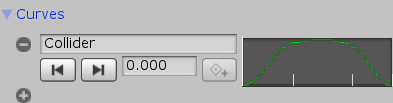
https://www.cnblogs.com/alps/p/7718966.html

**角色控制器允许你在受制于碰撞的情况下很容易的进行运动，而不用处理刚体。角色控制器不受力的影响，仅仅当你调用Move函数时才运动。然后它将执行运动，但是受制于碰撞。**

拥有多个属性 说白了就是专门帮助控制角色运动等的



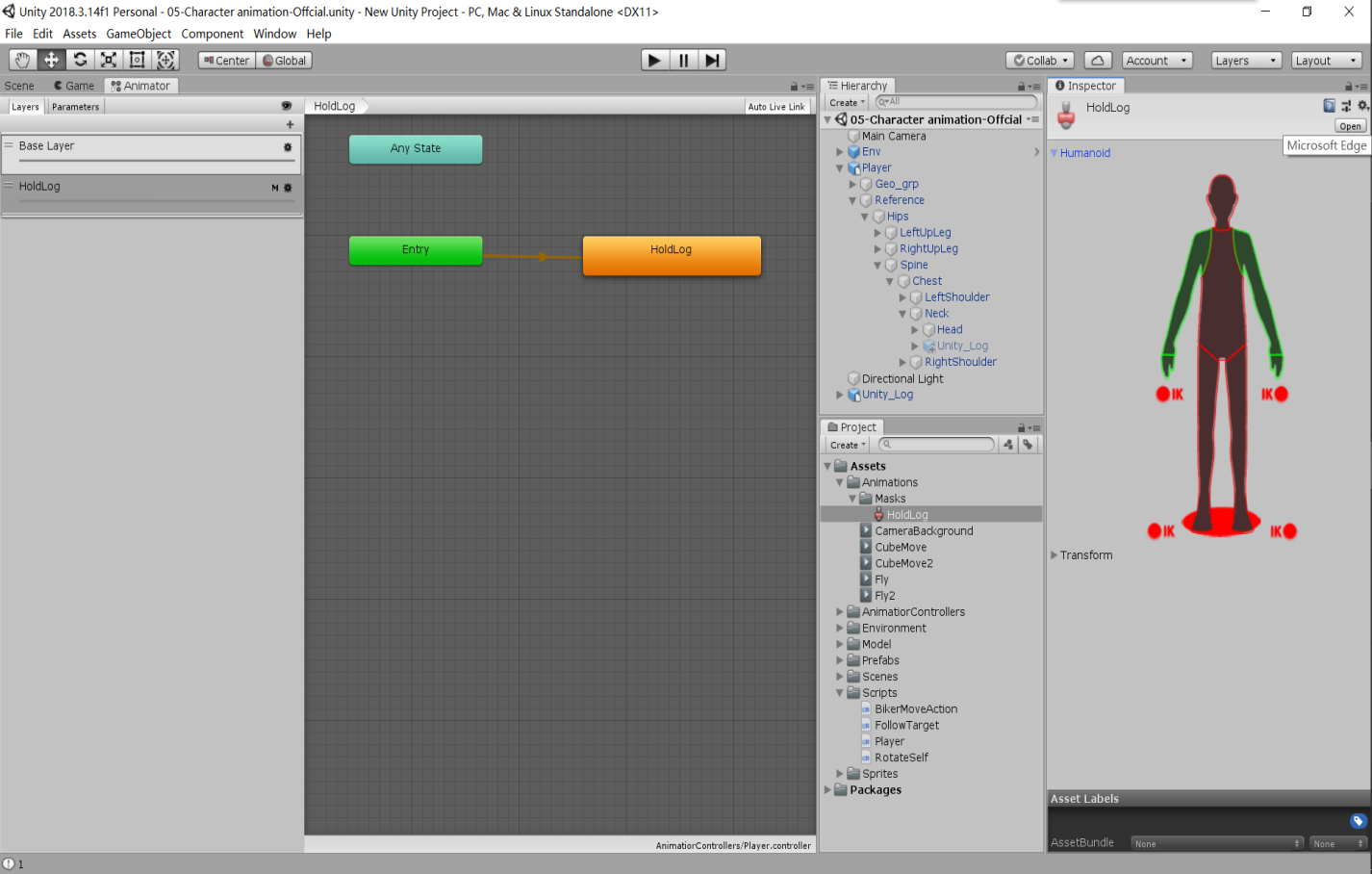
由于在进行特殊运动时角色控制器可能导致碰撞使运动失效 所以经常需要开启关闭角色控制器 这时候就需要在animator界面设置参数并且设置Curves来改变参数 通过代码来控制其开启和关闭

20.IK动画

**IK是Inverse Kinematic的缩写，也就是反向动力学。是根据骨骼的终节点来推算其他父节点的位置的一种方法。比如通过手的位置推算手腕、胳膊肘的骨骼的位置。**”  
“没错，那你能想到一些适用的场景么？”  
“嗯。。。我想想：**比如角色需要拿各种不同的东西，让角色的手能符合各种不同的东西的握持位置，这样就不用针对每种不同的东西单独制作动画了！**”  
“非常棒！这是IK最常见的一种用途。**其他的用途其实还有比如：角色的头的旋转，这样可以和你视角的方向一致。角色的脚的位置，这样可以让角色踩在地面跟贴合。**”  
“对对对，我只想到手了。那还有其他的么？”  
“**Unity中IK能设置的部位就是5个，分别是：头、左右手、左右脚。**所以没有其他部位的IK了，我们常见的其实也都是这些。”

3.不同层次的layers有两种影响的模式 override additive 顾名思义 如果有冲突 前一种方式会把前面的layer覆盖 而后一种是附加，比如前面的动画没有涉及手，就可以附加手的动画，但如果涉及到手，便不会产生作用

为了防止override将前面的layer覆盖 可以使用AvatarMask 针对人性的骨骼，可以选择动画影响的地方，影响的地方设定成绿色

4.类似于mashtarget mashtarget是通过差值 Ik动画可以直接指定目标结点，使指定的东西和目标结点的相对位置，相对角度不变



1.需说明权重 2.在AvatarIKGoal中只能选择可动的绿色位置类似于上图