**iTween动画库：关于Move移动篇**

Posted on 2013年01月24日 by U3d / [Unity3D脚本/插件](http://www.unitymanual.com/category/script)/被围观 574 次

iTween这个类库的主要功能就是处理模型从起始点到结束点之间运动的轨迹。（移动，旋转，音频，路径，摄像机等）它是一个开源的项目并且完全免费，它们的官网在这里 <http://itween.pixelplacement.com/index.php> 打开网之后点击右上角Get iTween图标即可，或者在AssetStores商店中直接下载。

移动模型时候用到的几个核心方法如下：

iTween.MoveTo()： 让模型移动到一个位置，它的底层函数是通过动态的修改模型每一帧的transform.position完成的，所以它会百分之百到达目标点，不会出现误差。

iTween.MoveFrom():它和上面的一样，iTween.MoveTo()是将模型移动到目标位置，而iTween.MoveFrom()是将模型从目标位置移动到原始位置。

iTween.MoveAdd() 和iTween.MoveBy()底层实现一样，大家可以去看源码。处理移动时采用的是transform.Translate也就是API的平移，这样在处理移动的时候可能会出现一些误差，但是效果好点。

iTween.MoveUpdate():和iTween.MoveTo()差不多，只是它需要放在循环或者Update()中。

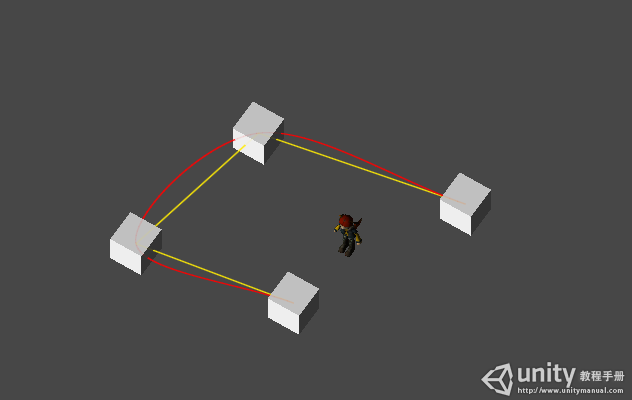
有了核心的移动方法后，我们就来了解iTween强大的核心参数，与事件。移动方法的参数都差不多，所以这里我就以MoveTo来做例子。直接上代码。

Move.cs绑定在需要移动的游戏对象身上。

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |
| --- | --- |
| 01 | **using** UnityEngine; |
| 02 | **using** System.Collections; |
| 03 |  |
| 04 | **public** **class** Move : MonoBehaviour |
| 05 | { |
| 06 |  |
| 07 | **void** Start() |
| 08 | { |
| 09 |  |
| 10 | *//键值对儿的形式保存iTween所用到的参数* |
| 11 | Hashtable args = new Hashtable(); |
| 12 |  |
| 13 | *//这里是设置类型，iTween的类型又很多种，在源码中的枚举EaseType中* |
| 14 | *//例如移动的特效，先震动在移动、先后退在移动、先加速在变速、等等* |
| 15 | args.**Add**("easeType", iTween.EaseType.easeInOutExpo); |
| 16 |  |
| 17 | *//移动的速度，* |
| 18 | args.**Add**("speed",10f); |
| 19 | *//移动的整体时间。如果与speed共存那么优先speed* |
| 20 | args.**Add**("time",1f); |
| 21 | *//这个是处理颜色的。可以看源码的那个枚举。* |
| 22 | args.**Add**("NamedValueColor","\_SpecColor"); |
| 23 | *//延迟执行时间* |
| 24 | args.**Add**("delay", 0.1f); |
| 25 | *//移动的过程中面朝一个点* |
| 26 | args.**Add**("looktarget",Vector3.zero); |
| 27 |  |
| 28 | *//三个循环类型 none loop pingPong (一般 循环 来回)* |
| 29 | *//args.Add("loopType", "none");* |
| 30 | *//args.Add("loopType", "loop");* |
| 31 | args.**Add**("loopType", "pingPong"); |
| 32 |  |
| 33 | *//处理移动过程中的事件。* |
| 34 | *//开始发生移动时调用AnimationStart方法，5.0表示它的参数* |
| 35 | args.**Add**("onstart", "AnimationStart"); |
| 36 | args.**Add**("onstartparams", 5.0f); |
| 37 | *//设置接受方法的对象，默认是自身接受，这里也可以改成别的对象接受，* |
| 38 | *//那么就得在接收对象的脚本中实现AnimationStart方法。* |
| 39 | args.**Add**("onstarttarget", gameObject); |
| 40 |  |
| 41 | *//移动结束时调用，参数和上面类似* |
| 42 | args.**Add**("oncomplete", "AnimationEnd"); |
| 43 | args.**Add**("oncompleteparams", "end"); |
| 44 | args.**Add**("oncompletetarget", gameObject); |
| 45 |  |
| 46 | *//移动中调用，参数和上面类似* |
| 47 | args.**Add**("onupdate", "AnimationUpdate"); |
| 48 | args.**Add**("onupdatetarget", gameObject); |
| 49 | args.**Add**("onupdateparams", **true**); |
| 50 |  |
| 51 | *// x y z 标示移动的位置。* |
| 52 |  |
| 53 | args.**Add**("x",5); |
| 54 | args.**Add**("y",5); |
| 55 | args.**Add**("z",1); |
| 56 |  |
| 57 | *//当然也可以写Vector3* |
| 58 | *//args.Add("position",Vectoe3.zero);* |
| 59 |  |
| 60 | *//最终让改对象开始移动* |
| 61 | iTween.MoveTo(gameObject,args); |
| 62 | } |
| 63 |  |
| 64 | *//对象移动中调用* |
| 65 | **void** AnimationUpdate(**bool** f) |
| 66 | { |
| 67 | Debug.Log("update :" + f); |
| 68 | } |
| 69 | *//对象开始移动时调用* |
| 70 | **void** AnimationStart(**float** f) |
| 71 | { |
| 72 | Debug.Log("start :" + f); |
| 73 | } |
| 74 | *//对象移动时调用* |
| 75 | **void** AnimationEnd(**string** f) |
| 76 | { |
| 77 | Debug.Log("end : " + f); |
| 78 | } |
| 79 |  |
| 80 | } |

在看看iTween中的寻路算法，其实非常非常的简单，我们几乎不用做任何事情。如下图所示，我们能清楚的看到编辑了一个简单的寻路，我们通过iTween 来实现小人跑步到终点。



iTween动画库：关于Move移动篇

Path.cs 绑在主角身上即可。

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |
| --- | --- |
| 01 | **using** UnityEngine; |
| 02 |  |
| 03 | **using** System.Collections; |
| 04 |  |
| 05 | **public** **class** Path : MonoBehaviour { |
| 06 |  |
| 07 | *//路径寻路中的所有点* |
| 08 |  |
| 09 | **public** Transform [] paths; |
| 10 |  |
| 11 | **void** Start () |
| 12 |  |
| 13 | { |
| 14 |  |
| 15 | Hashtable args = new Hashtable(); |
| 16 |  |
| 17 | *//设置路径的点* |
| 18 |  |
| 19 | args.**Add**("path",paths); |
| 20 |  |
| 21 | *//设置类型为线性，线性效果会好一些。* |
| 22 |  |
| 23 | args.**Add**("easeType", iTween.EaseType.linear); |
| 24 |  |
| 25 | *//设置寻路的速度* |
| 26 |  |
| 27 | args.**Add**("speed",10f); |
| 28 |  |
| 29 | *//是否先从原始位置走到路径中第一个点的位置* |
| 30 |  |
| 31 | args.**Add**("movetopath",**true**); |
| 32 |  |
| 33 | *//是否让模型始终面朝当面目标的方向，拐弯的地方会自动旋转模型* |
| 34 |  |
| 35 | *//如果你发现你的模型在寻路的时候始终都是一个方向那么一定要打开这个* |
| 36 |  |
| 37 | args.**Add**("orienttopath",**true**); |
| 38 |  |
| 39 | *//让模型开始寻路* |
| 40 |  |
| 41 | iTween.MoveTo(gameObject,args); |
| 42 |  |
| 43 | } |
| 44 |  |
| 45 | **void** OnDrawGizmos() |
| 46 |  |
| 47 | { |
| 48 |  |
| 49 | *//在scene视图中绘制出路径与线* |
| 50 |  |
| 51 | iTween.DrawLine(paths,Color.yellow); |
| 52 |  |
| 53 | iTween.DrawPath(paths,Color.red); |
| 54 |  |
| 55 | } |
| 56 |  |
| 57 | } |

运行后即可看到主角自动寻路的效果。

如果你仔细阅读到这里你可能会想到，iTween做的东西有点像 Mathf.Lerp() Vector3.Lerp() lookAt()等等这类的方法。假设不使用iTween这个类就用源生的API其实也可以实现上述的所有效果。只有iTween帮我们封装的更好一些，平滑过渡的效果更好一些，而且还能增加一些特效。只是这些特效与动画全都是iTween通过数学的方法计算出来。因为底层它们使用的也是简单的 移动旋转API中的方法。我觉得寻路的话可以使用Unity自带的方法（因为是官方提供的），处理一些简单的动画使用iTween还是挺不错的，因为更加形象。

代码下载地址：<http://vdisk.weibo.com/s/omt5Q>

原文来自：http://www.xuanyusong.com/archives/2052