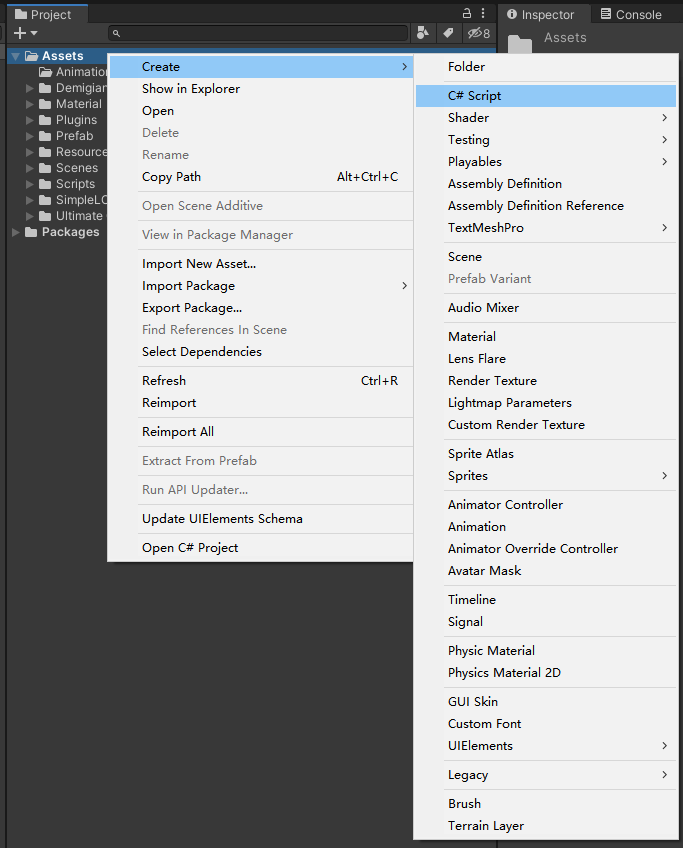
## 运动仿真

前面已经提到Unity中脚本的存在形式也是组件(Component)，因此运动仿真的一个**主要思路**就是先编辑脚本，再将脚本作为组件挂在某个物体上。

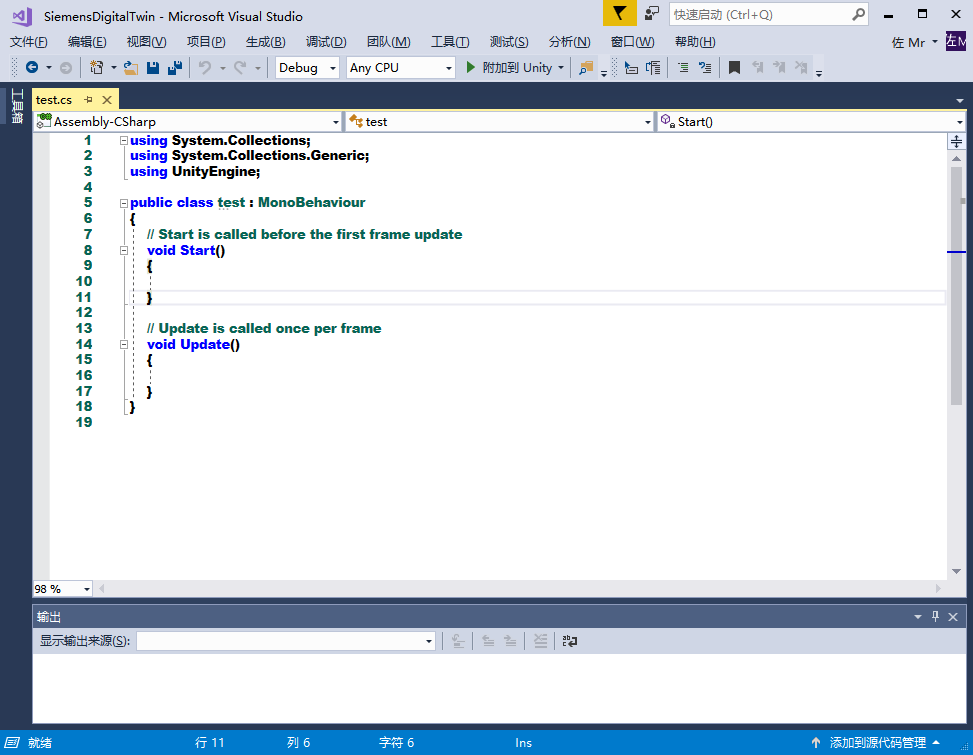
### 操作流程

右键单击【Assets】--【Create】--【C# Script】创建C#脚本。



**图1.1 创建从c#脚本步骤**

双击创建的C#脚本，进入Visual Studio软件如下图1.2所示：



**图1.2 脚本编辑窗口**

### Unity生命周期（☆☆☆）

可以看到新建脚本包括Start方法、Update方法，这涉及到Unity的生命周期，如下图所示：

最先执行的方法是Awake，这是生命周期的开始，用于进行激活时的初始化代码，一般可以在这个地方将当前脚本禁用: this.enable=false，如果这样做了，则会直接跳转到OnDisable方法执行一次，然后其它的任何方法都将不再被执行。

如果当前脚本处于可用状态，则正常的执行顺序是继续向下执行OnEnable，当然我们可以在另外一个脚本中实现这个脚本组件的启动：this.enable =true;

再向下执行，会进行一个判断，如果Start方法还没有被执行，则会被执行一次，如果已经被执行了，则不会再被执行。这是个什么意思呢？我们可以在某个脚本中将组件禁用this.enable=false，再启用时会转到OnEnable处执行，这时继续向下走，发现Start执行过了，将不再被执行。比如说：第一次启用时，将怪物的初始位置定在了(0,0,0)点，然后怪物可能会发生了位置的变换，后来被禁用了，再次启用时，不会让怪物又回到初始的(0,0,0)位置。

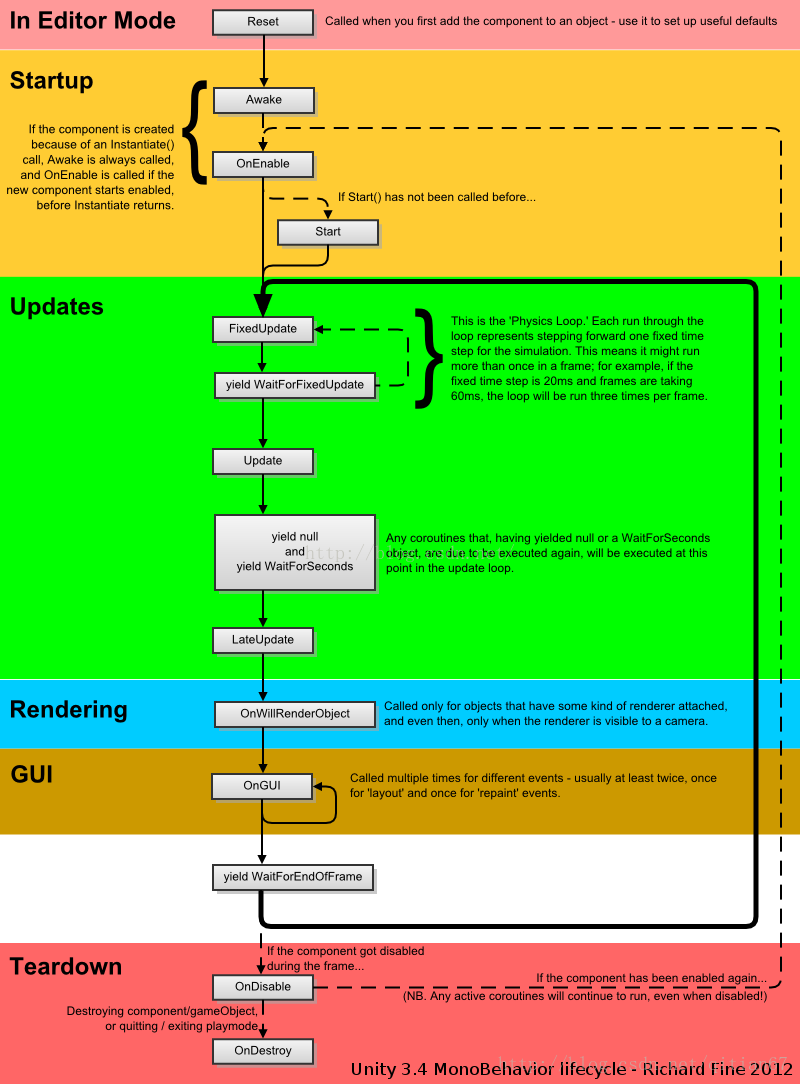
继续向后执行，就是FixedUpdate了，然后是Update，再然后是LateUpdate，如果后面写了Reset，则会又回到Update，在这4个事件间可以进行循环流动。

再向后执行，就进入了渲染模块（Rendering），非常重要的一个方法就是OnGUI，用于绘制图形界面。当然，如果你使用了NGUI，这个生命周期的事情你就不用考虑了。

再向后，就是卸载模块（TearDown），这里主要有两个方法OnDisable与OnDestroy。当被禁用(enable=false)时，会执行OnDisable方法，但是这个时候，脚本并不会被销毁，在这个状态下，可以重新回到OnEnable状态（enable=true）。当手动销毁或附属的游戏对象被销毁时，OnDestroy才会被执行，当前脚本的生命周期结束。

脚本自带函数执行顺序如下：将下面脚本挂在任意物体运行即可得到

Awake -> OnEnable -> Start -> FixedUpdate -> Update -> LateUpdate -> OnGUI -> Reset -> OnDisable -> OnDestroy



**图1.3 Unity脚本生命周期**

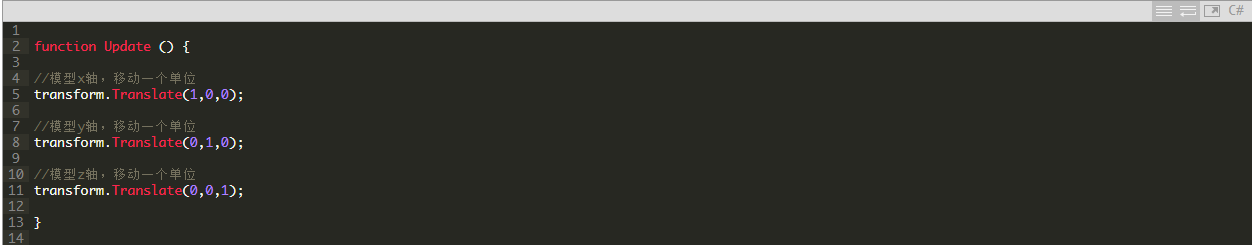
* 1.Awake：用于在游戏开始之前初始化变量或游戏状态。在脚本整个生命周期内它仅被调用一次.Awake在所有对象被初始化之后调用，所以你可以安全的与其他对象对话或用诸如GameObject.FindWithTag()这样的函数搜索它们。每个游戏物体上的Awake以随机的顺序被调用。因此，你应该用Awake来设置脚本间的引用，并用Start来传递信息Awake总是在Start之前被调用。它不能用来执行协同程序。
* 2.Start：仅在Update函数第一次被调用前调用。Start在behaviour的生命周期中只被调用一次。它和Awake的不同是Start只在脚本实例被启用时调用。你可以按需调整延迟初始化代码。Awake总是在Start之前执行。这允许你协调初始化顺序。在所有脚本实例中，Start函数总是在Awake函数之后调用。
* 3.FixedUpdate：固定帧更新，在Unity导航菜单栏中，点击“Edit”–>“Project Setting”–>“Time”菜单项后，右侧的Inspector视图将弹出时间管理器，其中“Fixed Timestep”选项用于设置FixedUpdate()的更新频率，更新频率默认为0.02s。
* 4.Update：正常帧更新，用于更新逻辑。每一帧都执行，处理Rigidbody时，需要用FixedUpdate代替Update。例如:给刚体加一个作用力时，你必须应用作用力在FixedUpdate里的固定帧，而不是Update中的帧。(两者帧长不同)FixedUpdate，每固定帧绘制时执行一次，和update不同的是FixedUpdate是渲染帧执行，如果你的渲染效率低下的时候FixedUpdate调用次数就会跟着下降。FixedUpdate比较适用于物理引擎的计算，因为是跟每帧渲染有关。Update就比较适合做控制。
* 5.LateUpdate：在所有Update函数调用后被调用，和fixedupdate一样都是每一帧都被调用执行，这可用于调整脚本执行顺序。例如:当物体在Update里移动时，跟随物体的相机可以在LateUpdate里实现。LateUpdate,在每帧Update执行完毕调用，他是在所有update结束后才调用，比较适合用于命令脚本的执行。官网上例子是摄像机的跟随，都是在所有update操作完才跟进摄像机，不然就有可能出现摄像机已经推进了，但是视角里还未有角色的空帧出现。
* 6.OnGUI：在渲染和处理GUI事件时调用。比如：你画一个button或label时常常用到它。这意味着OnGUI也是每帧执行一次。
* 7.Reset：在用户点击检视面板的Reset按钮或者首次添加该组件时被调用。此函数只在编辑模式下被调用。Reset最常用于在检视面板中给定一个默认值。
* 8.OnDisable：当物体被销毁时 OnDisable将被调用，并且可用于任意清理代码。脚本被卸载时，OnDisable将被调用，OnEnable在脚本被载入后调用。注意： OnDisable不能用于协同程序。
* 9.OnDestroy：当MonoBehaviour将被销毁时，这个函数被调用。OnDestroy只会在预先已经被激活的游戏物体上被调用。注意：OnDestroy也不能用于协同程序。

### 平移、旋转

接下来介绍西门子设备的托盘运动仿真。首先新建一个C#脚本，命名为“TuopanController”，在Project界面中打开该脚本。

**（1）模型的平移**

模型的移动可以使用**Translate**方法，Translate方法中的三个参数分别表示模型在三维世界中X、Y、Z轴移动的单位距离。

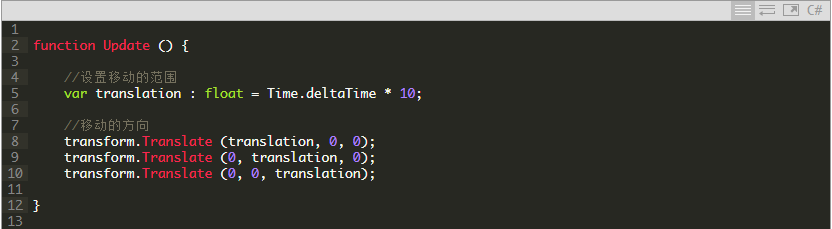


**图1.4 Translate方法一**

执行代码发现托盘移动速度为1m/s，怎么能修改移动的速度呢？

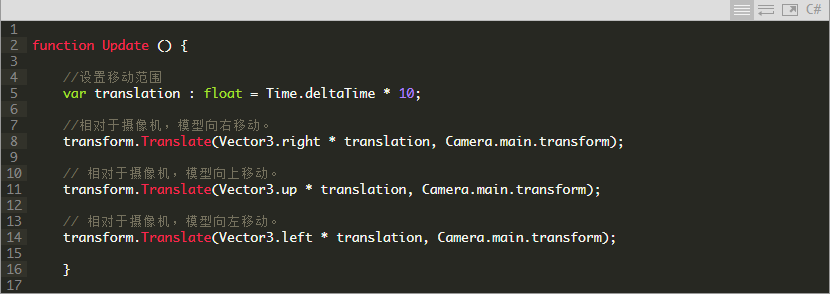
* Time.deltaTime：标示上一次调用Update一秒为标示每帧执行所消耗的时间。

使用该参数，我们就可以根据它修改方向移动的速度了。



**图1.5 Translate方法二**

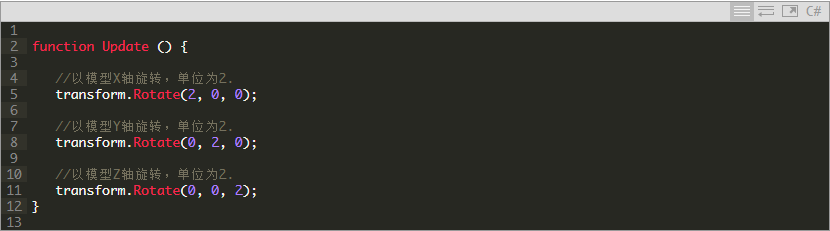
模型的平移可以选择一个参照物，下面代码第二个参数设置模型移动参照物，这里设置成摄像机。那么模型将以相对与摄像机进行移动。



**图1.6 Translate方法三**

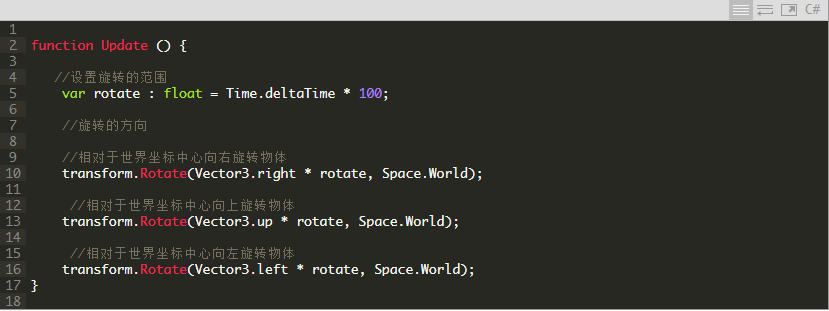
**（2）模型的旋转**

模型的旋转可以使用**Rotate**方法，Rotate方法中的三个参数分别表示模型在三维世界中X、Y、Z轴旋转的单位距离。



**图1.7 Rotate方法一**

模型的旋转可以选择一个参照物，下面代码第二个参数设置模型移动参照物，这里设置成3D世界。那么模型将以相对与整个3D世界进行旋转。



**图1.7 Rotate方法二**

### Dotween插件

DOTween是一款针对Unity的快速、高效、完全类型安全的面向对象的补间动画引擎，并且对于C#用户做出了很多的优化。

接下来以DOTween插件为开发工具，介绍西门子设备的托盘运动仿真。首先新建一个C#脚本，命名为“TuopanController”，在Project界面中打开该脚本。

**（1）模型的平移**

模型的平移可以使用**DOMove**相关方法，脚本内容如下：

using DG.Tweening; //引入命名空间

public class TuopanController : MonoBehaviour

{

  void Start()

    {

        transform.DOMove(new Vector3(4, 3, 0), 3); //脚本物体3秒从当前位置移动 到世界坐标(4,3,0)位置

    }

}

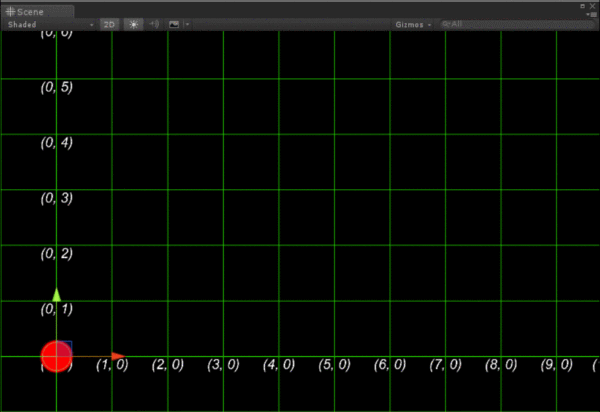


图1.8 [动图.files\slide0001.htm](动图.files/slide0001.htm)

可以看到，在使用DOTween插件后，transform方法也能够实现DOMove方法，这是因为C#的拓展性，使其和Unity的一些类能产生链接。因此，我们想让某个物体移动时，让该物体的transform组件调用DOTWeen方法即可。

常用的DOMove相关方法包括以下几种：

* 世界坐标上移动：transform.DOMove
* 本地坐标上移动：transform.DOLocalMove
* 世界坐标的X轴上移动：transform.DOMoveX
* 本地坐标的X轴上移动：transform.DOLocalMoveX

**（2）模型的旋转**

模型的旋转可以使用**DORotate**相关方法，脚本内容如下：

using DG.Tweening; //引入命名空间

public class TuopanController : MonoBehaviour

{

  void Start()

    {

        transform.DORotate(new Vector3(0, 90, 0), 3); //脚本物体3秒Y轴旋转到 90°,X轴及Z轴旋转到0°

    }

}

常用的DORotate相关方法包括以下几种：

* 世界坐标内旋转：transform.DORotate
* 本地坐标内旋转：transform.DOLocalRotate
* 绕世界坐标的X轴旋转：transform.DORotateX
* 绕本地坐标的X轴旋转：transform.DOLocalRotateX

**（3）动画正放与倒放**

上述制作的移动旋转动画会会自动播放，并且播放完后自动销毁，无法重复使用，因此需要通过一些设置来避免。脚本内容如下：

using DG.Tweening; //引入命名空间

public class DOTWeenTest : MonoBehaviour

{

    Tweener tw; //声明一个Tweener对象

    void Start()

    {

tw= transform.DOMove(new Vector3(3, 4, 0), 2);//将动画保存在

Tweener对象中

        tw.Pause();//暂停,防止自动播放

        tw.SetAutoKill(false);//关闭动画自动销毁

    }

    //创建两个方法事件,控制前放后倒放

    public void Forward()

    {

        tw.PlayForward(); //该动画正放

    }

    public void Back()

    {

        tw.PlayBackwards(); //该动画倒放

    }

}

将Forward和Back两个方法分别挂在“前”、“后”两个按钮上，运行程序，结果如图所示：

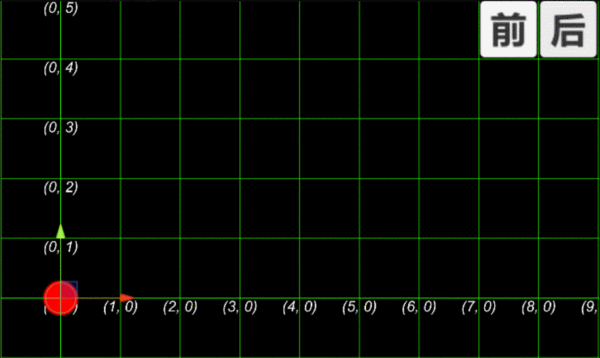


图1.9 [正反播放.files\slide0001.htm](正反播放.files/slide0001.htm)

动画播放相关的其他常用方法包括：

Pause(): 暂停动画

SetLoops(n): 动画循环n次

**（4）设置动画曲线**

在上述展示的动画效果中可以看到，物体在播放动画时速度是由快到慢（倒放除外），这种现象是由动画播放时的动画曲线决定的，我们可以通过修改动画曲线来改变物体动画效果，脚本如下：

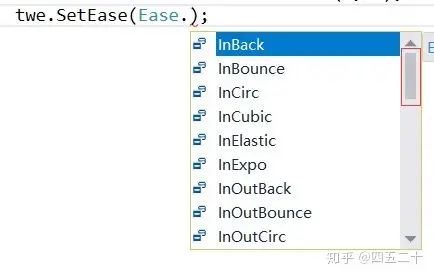
{

    Tweener twe = transform.DOMoveX(5, 2); // 3秒时间在世界坐标中,让X轴 移动到5的位置

    twe.SetEase(Ease.InCubic); //由慢到快

}

动画曲线的种类非常多，读者可以自行查阅文档。



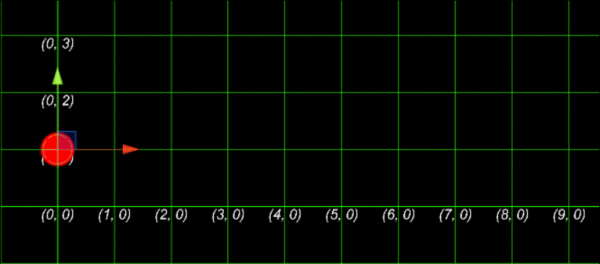


图1.10 [动画曲线.files\slide0001.htm](动画曲线.files/slide0001.htm)

**（5）可视化编辑**

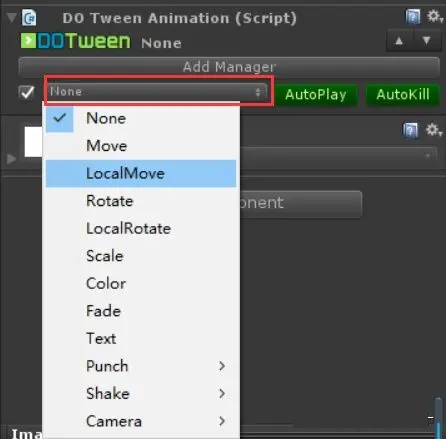
除了使用脚本来播放DOTween动画外，该插件还拥有可视化编辑的功能。

* 1.为需要做动画的物体添加DoTweenAnimation组件：



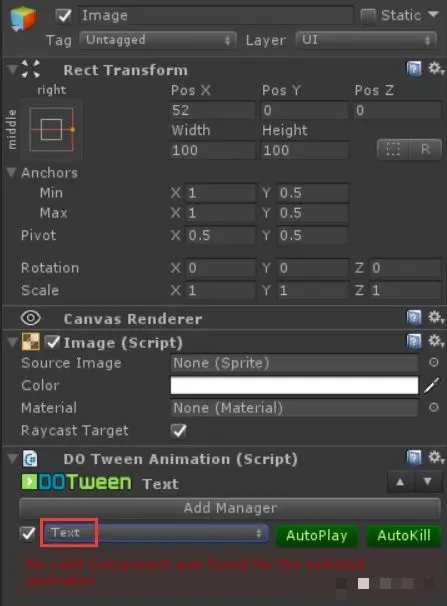
**图1.11 DoTweenAnimation组件**

* 2.选择要做动画的组件：



**图1.12 动画组件**

* 3.当要做动画的组件不存在时则报错，非常方便：



**图1.13 dotween组件报错**

* 4.各种参数：



**图1.14 dotween组件参数**

* 5.当在编辑器关闭了自动播放后，使用代码播放：

public DOTweenAnimation a; //在编辑器界面将挂有该组件的物体拖进来

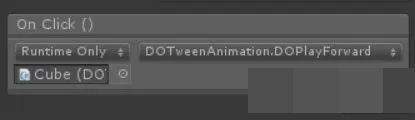
void Start()

{

    a.DOPlayForward();

}

* 6.如果不使用代码，使用点击按钮播放,在按钮的事件窗口将物体拖进去,找到DoTweenAnimation组件,调用播放方法:

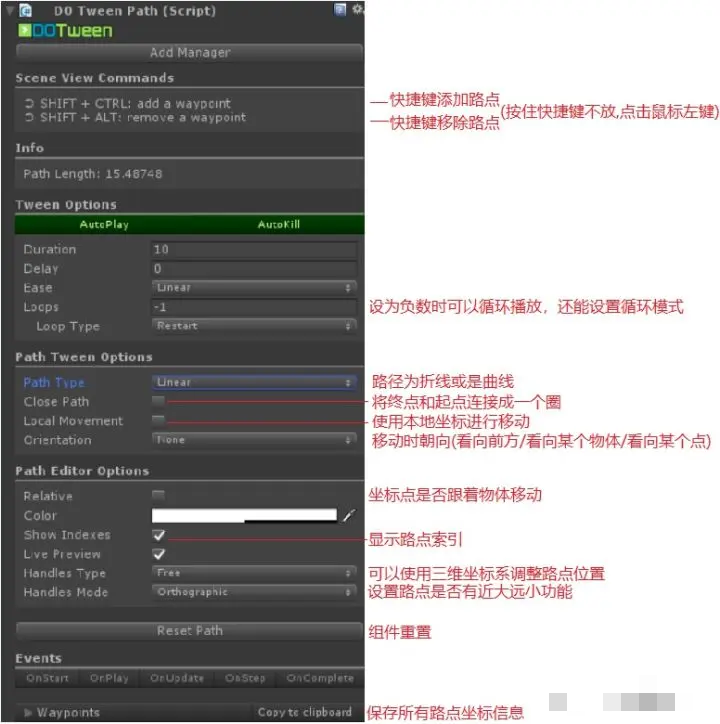


**图1.15 DoTween组件播放按钮**

**（6）路径编辑器**

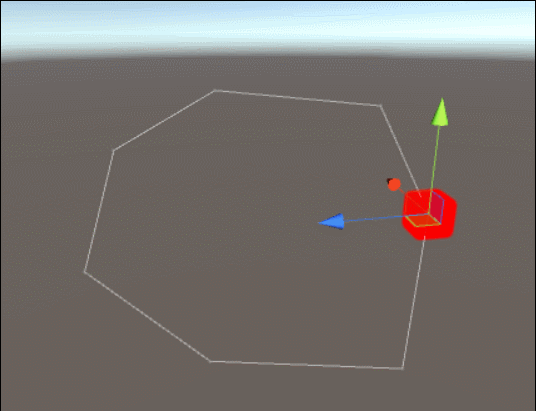
使用该组件可以做一些寻路的效果：

* 1.在场景中创建一个Cube，添加DoTweenPath组件：



**图1.16 DoTweenPath组件**

* 2.运行程序：



**图1.20** [路径编辑器.files\slide0001.htm](路径编辑器.files/slide0001.htm)