**利用Animation View制作动态的GUI**

Posted on 2013年04月15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 218 次

[**Unity3D教程**](http://www.unitymanual.com/category/manual)：利用Animation View制作动态的GUI。如果要使 GUI 动态产生位移、缩放等动态变化，我们只要依时间改变去计算它们的值就可以做到，这使人想到又要宣告变数并写程式，但是如果善用 Unity 的 Animation View，可能不需要多写任何程式码就能完成更棒的效果。如果想使一个按钮从左向右移，程式如下：

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

|  |  |
| --- | --- |
| 01 | *//啟始 x 座標* |
| 02 | **public** **float** **from**; |
| 03 | *//目標 x 座標* |
| 04 | **public** **float** to; |
| 05 | *//速度* |
| 06 | **public** **float** speed = 1; |
| 07 |  |
| 08 | *//大小位置* |
| 09 | **public** Rect position = new Rect(0,0,50,25); |
| 10 |  |
| 11 | **void** Start(){ |
| 12 |  |
| 13 | position.x = **from**; |
| 14 | } |
| 15 |  |
| 16 | **void** Update(){ |
| 17 |  |
| 18 | **if**(position.x < to) position.x += Time.deltaTime \* speed; |
| 19 | } |
| 20 |  |
| 21 | **void** OnGUI(){ |
| 22 |  |
| 23 | GUI.Button(position , "test"); |
| 24 | } |

这样虽然达到目的，但无法控制几秒内完成这个动作，那么可以利用 Unity 的 Mathf Struct 改成这样：

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

|  |  |
| --- | --- |
| 01 | *//啟始 x 座標* |
| 02 | **public** **float** **from**; |
| 03 | *//目標 x 座標* |
| 04 | **public** **float** to; |
| 05 | *//減緩* |
| 06 | **public** **float** delay = 1; |
| 07 |  |
| 08 | *//大小位置* |
| 09 | **public** Rect position = new Rect(0,0,50,25); |
| 10 |  |
| 11 | **void** Update(){ |
| 12 |  |
| 13 | **if**(position.x < to){ |
| 14 | position.x = Mathf.Lerp(**from** , to , t); |
| 15 | t += Time.deltaTime \* delay; |
| 16 | } |
| 17 | } |
| 18 |  |
| 19 | **void** OnGUI(){ |
| 20 |  |
| 21 | GUI.Button(position , "test"); |
| 22 | } |

这样改变数值的位移是属於线性变化，移动起来好像有点死版，所以也可以把 Mathf.Lerp 改成使用 Mathf.SmoothDamp，那么数值变化会变得更圆滑一些。

如果会使用 iTween 的人可能会觉得 iTween 的 EaseType 呈现的动态效果曲线更好，所以可能会写这个样子：

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

|  |  |
| --- | --- |
| 01 | *//啟始 x 座標* |
| 02 | **public** **float** **from**; |
| 03 | *//目標 x 座標* |
| 04 | **public** **float** to; |
| 05 | *//完成動作秒數* |
| 06 | **public** **float** sec = 1; |
| 07 | *//大小位置* |
| 08 | **public** Rect position = new Rect(0,0,50,25); |
| 09 |  |
| 10 | **void** OnGUI(){ |
| 11 |  |
| 12 | **if**(GUI.Button(position , "test")){ |
| 13 | iTween.ValueTo(gameObject, |
| 14 | iTween.Hash( |
| 15 | "from" , **from** , |
| 16 | "to" , to , |
| 17 | "onupdate" , "UpdateX" , |
| 18 | "easetype" , "easeinoutback", |
| 19 | "time" , speed |
| 20 | ) |
| 21 | ); |
| 22 | } |
| 23 | } |
| 24 |  |
| 25 | **private** **void** UpdateX(**float** x){ |
| 26 |  |
| 27 | position.x = x; |
| 28 | } |

以上，只要点击按钮就可以依指定的 EaseType 效果位移。

但是如果善用 Unity 的 Animation View 的话，这些计算都可以省下来，只要在需要动作的时候让 Animation View 执行 Play 即可，而数值调整只要在 Animation View 中调整就好了，那麽程式可能改成这样：

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

|  |  |
| --- | --- |
| 1 | *//大小位置* |
| 2 | **public** Rect position = new Rect(0,0,50,25); |
| 3 |  |
| 4 | **void** OnGUI(){ |
| 5 |  |
| 6 | **if**(GUI.Button(position , "start")){ |
| 7 | animation.Play(); |
| 8 | } |
| 9 | } |

而 Animation View 中只是对这个 GUI script 的 position.x 设定两个 key 所连接的直线。

如果想要像 iTween 的 EaseType 的曲线那样变化值的话，只要为 Animation View 的直线添加几个 key 调整成类似的曲线，那麽就能获得类似的动态；而且也可以依照自己的喜好拉出特别的动作曲线，而不用受限于他人的定义。