[NGUI研究院之三种方式监听NGUI的事件方法（七）](http://www.xuanyusong.com/archives/2390)

**NGUI事件的种类很多，比如点击、双击、拖动、滑动等等，他们处理事件的原理几乎万全一样，本文只用按钮来举例。**

**1.直接监听事件**

**把下面脚本直接绑定在按钮上，当按钮点击时就可以监听到，这种方法不太好很不灵活。**

|  |  |  |
| --- | --- | --- |
| 1 | void OnClick() | |
| 2 | { |

|  |  |  |
| --- | --- | --- |
| 3 | Debug.Log("Button is Click!!!"); | |
| 4 | } |

**2.使用SendMessage**

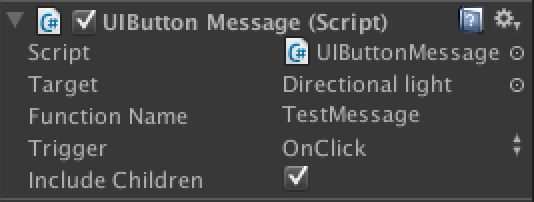
**选择按钮后，在Unity导航菜单栏中选择Component->Interaction->Button Message 组件。**

**Target：接收按钮消息的游戏对象。**

**Function Name：接收按钮消息的方法，拥有这个方法的脚本必须绑定在上面Target对象身上。**

**Trigger：触发的事件，OnClick显然是一次点击。**

**Include Children ：是否让该对象的所有子对象也发送这个点击事件。**

[](http://www.xuanyusong.com/wp-content/uploads/2013/06/%E5%B1%8F%E5%B9%95%E5%BF%AB%E7%85%A7-2013-06-24-%E4%B8%8B%E5%8D%885.28.53.png)

**到UIButtonMessage.cs这个脚本中看看，其实很简单就是调用Unity自身的SendMessage而已。**

|  |  |  |
| --- | --- | --- |
| 01 | void Send () | |
| 02 | { |

|  |  |  |
| --- | --- | --- |
| 03 | if (string.IsNullOrEmpty(functionName)) return; | |
| 04 | if (target == null) target = gameObject; |

|  |  |
| --- | --- |
| 05 |  |
| 06 | if (includeChildren) | |

|  |  |
| --- | --- |
| 07 | { |
| 08 | Transform[] transforms = target.GetComponentsInChildren<Transform>(); | |

|  |  |
| --- | --- |
| 09 |  |
| 10 | for (int i = 0, imax = transforms.Length; i < imax; ++i) | |

|  |  |
| --- | --- |
| 11 | { |
| 12 | Transform t = transforms[i]; | |

|  |  |  |
| --- | --- | --- |
| 13 | t.gameObject.SendMessage(functionName, gameObject, SendMessageOptions.DontRequireReceiver); | |
| 14 | } |

|  |  |
| --- | --- |
| 15 | } |
| 16 | else | |

|  |  |
| --- | --- |
| 17 | { |
| 18 | target.SendMessage(functionName, gameObject, SendMessageOptions.DontRequireReceiver); | |

|  |  |  |
| --- | --- | --- |
| 19 | } | |
| 20 | } |

**3.使用UIListener**

**这个也是推荐大家使用的一种方法，选择按钮后在Unity导航菜单栏中选择Component->NGUI->Internal ->Event Listener 。 挂在按钮上就可以，它没有任何参数。。**

[](http://www.xuanyusong.com/wp-content/uploads/2013/06/%E5%B1%8F%E5%B9%95%E5%BF%AB%E7%85%A7-2013-06-24-%E4%B8%8B%E5%8D%885.38.30.png)

**在任何一个脚本或者类中即可得到按钮的点击事件、把如下代码放在任意类中或者脚本中。**

|  |  |  |
| --- | --- | --- |
| 01 | void Awake () | |
| 02 | { |

|  |  |
| --- | --- |
| 03 | //获取需要监听的按钮对象 |
| 04 | GameObject button = GameObject.Find("UI Root (2D)/Camera/Anchor/Panel/LoadUI/MainCommon/Button"); | |

|  |  |  |
| --- | --- | --- |
| 05 | //设置这个按钮的监听，指向本类的ButtonClick方法中。 | |
| 06 | UIEventListener.Get(button).onClick = ButtonClick; |

|  |  |
| --- | --- |
| 07 | } |
| 08 |  | |

|  |  |
| --- | --- |
| 09 | //计算按钮的点击事件 |
| 10 | void ButtonClick(GameObject button) | |

|  |  |
| --- | --- |
| 11 | { |
| 12 | Debug.Log("GameObject " + button.name); | |

|  |  |  |
| --- | --- | --- |
| 13 |  | |
| 14 | } |

**怎么样是不是很灵活？再看看它的源码，使用的C#的代理，将UI的场景事件通过代理传递出去了。**

|  |  |  |
| --- | --- | --- |
| 01 | public class UIEventListener : MonoBehaviour | |
| 02 | { |

|  |  |
| --- | --- |
| 03 | public delegate void VoidDelegate (GameObject go); |
| 04 | public delegate void BoolDelegate (GameObject go, bool state); | |

|  |  |
| --- | --- |
| 05 | public delegate void FloatDelegate (GameObject go, float delta); |
| 06 | public delegate void VectorDelegate (GameObject go, Vector2 delta); |

|  |  |
| --- | --- |
| 07 | public delegate void StringDelegate (GameObject go, string text); |
| 08 | public delegate void ObjectDelegate (GameObject go, GameObject draggedObject); |

|  |  |  |
| --- | --- | --- |
| 09 | public delegate void KeyCodeDelegate (GameObject go, KeyCode key); | |
| 10 |  |

|  |  |  |
| --- | --- | --- |
| 11 | public object parameter; | |
| 12 |  |

|  |  |  |
| --- | --- | --- |
| 13 | public VoidDelegate onSubmit; | |
| 14 | public VoidDelegate onClick; |

|  |  |  |
| --- | --- | --- |
| 15 | public VoidDelegate onDoubleClick; | |
| 16 | public BoolDelegate onHover; |

|  |  |
| --- | --- |
| 17 | public BoolDelegate onPress; |
| 18 | public BoolDelegate onSelect; | |

|  |  |  |
| --- | --- | --- |
| 19 | public FloatDelegate onScroll; | |
| 20 | public VectorDelegate onDrag; |

|  |  |
| --- | --- |
| 21 | public ObjectDelegate onDrop; |
| 22 | public StringDelegate onInput; | |

|  |  |  |
| --- | --- | --- |
| 23 | public KeyCodeDelegate onKey; | |
| 24 |  |

|  |  |
| --- | --- |
| 25 | void OnSubmit ()                { if (onSubmit != null) onSubmit(gameObject); } |
| 26 | void OnClick ()                 { if (onClick != null) onClick(gameObject); } |

|  |  |
| --- | --- |
| 27 | void OnDoubleClick ()           { if (onDoubleClick != null) onDoubleClick(gameObject); } |
| 28 | void OnHover (bool isOver)      { if (onHover != null) onHover(gameObject, isOver); } |

|  |  |
| --- | --- |
| 29 | void OnPress (bool isPressed)   { if (onPress != null) onPress(gameObject, isPressed); } |
| 30 | void OnSelect (bool selected)   { if (onSelect != null) onSelect(gameObject, selected); } |

|  |  |
| --- | --- |
| 31 | void OnScroll (float delta)     { if (onScroll != null) onScroll(gameObject, delta); } |
| 32 | void OnDrag (Vector2 delta)     { if (onDrag != null) onDrag(gameObject, delta); } |

|  |  |
| --- | --- |
| 33 | void OnDrop (GameObject go)     { if (onDrop != null) onDrop(gameObject, go); } |
| 34 | void OnInput (string text)      { if (onInput != null) onInput(gameObject, text); } |

|  |  |  |
| --- | --- | --- |
| 35 | void OnKey (KeyCode key)        { if (onKey != null) onKey(gameObject, key); } | |
| 36 |  |

|  |  |
| --- | --- |
| 37 | /// <summary> |
| 38 | /// Get or add an event listener to the specified game object. | |

|  |  |  |
| --- | --- | --- |
| 39 | /// </summary> | |
| 40 |  |

|  |  |  |
| --- | --- | --- |
| 41 | static public UIEventListener Get (GameObject go) | |
| 42 | { |

|  |  |
| --- | --- |
| 43 | UIEventListener listener = go.GetComponent<UIEventListener>(); |
| 44 | if (listener == null) listener = go.AddComponent<UIEventListener>(); |

|  |  |  |
| --- | --- | --- |
| 45 | return listener; | |
| 46 | } |

|  |  |
| --- | --- |
| 47 | } |

**但是有时候我们项目中需要监听UI的东西可能不止这些，我们也可以拓展一下C#的事件方法。或者也可以使用**[**Unity3D研究院之通过C#使用Advanced CSharp Messenger（五十）**](http://www.xuanyusong.com/archives/2165)

http://www.xuanyusong.com/archives/2390