# **Unity制作一个好玩且鸡肋的操作回滚功能**

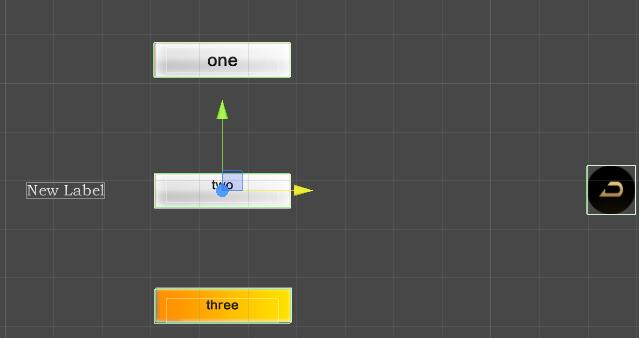
作者：[子奔](http://gad.qq.com/user/index?id=640841)

链接：<http://gad.qq.com/article/detail/286329>

在描述正文之前，必须介绍一下，堆栈后进先出的特点。

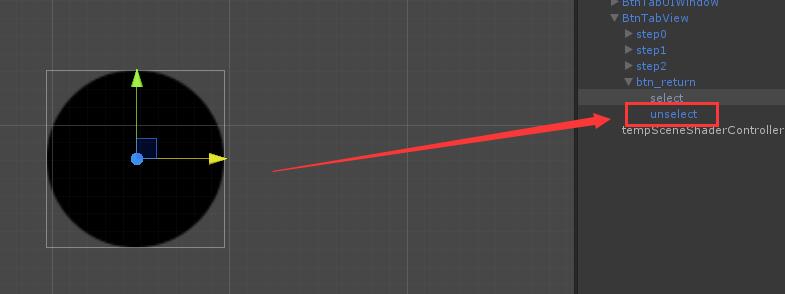
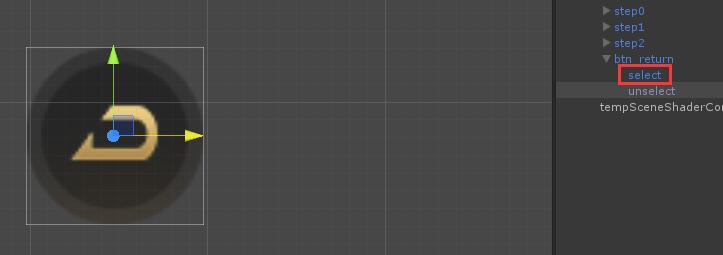
其实就好比，一组人进到一条黑漆漆只容得下一个人的死胡同，第一个人进去了，砰，撞墙了，但是他郁闷啊。前面都是人，他只能等最后进的人先出去，最后他才能出去。

操作回滚主要也是运用堆栈后进先出的特点进行编写的。



这是一个示例，假设这三个页签按键（当然实际上，可能是几个页面，几个Btn，几个文本）代表用户的操作，而返回键对用户操作进行回滚。

返回键：分为两种状态，第一是用户有了用户操作，可点击状态；第二是用户没有用户操作或者已经滚回去的不能再滚了，是不可点击状态。



好啦好啦，避免超字儿，贴上代码，TabButtonGroup是页签属性的类，以及页签本身所涉及的逻辑就不贴了。整体还是比较容易理解的。

public class BtnTabView : MonoBehaviour

｛

//定义的TabBtnGroup类存储TabBTN属性

private List<TabBtnGroup> m\_TabBtnsList = new List<TabBtnGroup>();

//对应每个按钮存在的委托事件(按键索引对应委托传进去的值)

private List<Action <int> > ClickCallBacks = new List<Action<int>> ();

//当前TabBTN的索引

private int m\_CurTabBtnIndex = 0;

//创建Stack存贮以前的点击索引

private Stack<int> m\_PreTabBtn = new Stack<int>();

//是不是第一次点击返回按键

//因为第一次点击要删除两次栈顶的值，之后就不应了

int m\_PreBtnIndex = 0;

private void Start()

｛

Transform root = this.m\_currentTrans;

if (root != null)

｛

AddTabBtn(root);

InitTabBtn();

//一打开设置为不可打开的状态

SetPreCanSel();

//在选中状态默认第一个按钮为第一个状态

m\_PreTabBtn.Push(0);

｝

｝

/// TabBTN点击事件的设置

public void Click(GameObject gb, UIEventData data)

｛

string clickName = gb.name;

//...针对页签的点击(省略)

if( clickName == "select" && data != null )

｛

HandleReturnBtn();

｝

｝

/// 对于返回键的操作

/// </summary>

private void HandleReturnBtn()

｛

//栈顶不为空

//为空设置为不可选状态

m\_PreBtnIndex++;

if ( PreBtnCount(m\_PreTabBtn) > 1 && m\_PreBtnIndex==1 )

｛

//把栈顶的值返回并删除,删除的第二个就是之前的按键

m\_PreTabBtn.Pop();

int preIndex = m\_PreTabBtn.Pop();

HandeleRetBtn(preIndex);

SetPreFirCanSel();

｝

if ( PreBtnCount(m\_PreTabBtn) > 0 && m\_PreBtnIndex != 1 )

｛

int preIndex = m\_PreTabBtn.Pop();

HandeleRetBtn(preIndex);

SetPreFirCanSel();

｝

｝

/// 返回栈顶是否有元素

private bool isPreTabBtnNULL(Stack<int> m\_PreTabBtn)

｛

return m\_PreTabBtn.IsNull();

｝

/// 得到PreTabBtn堆栈内存在的个数

private int PreBtnCount(Stack<int> m\_PreTabBtn)

｛

return m\_PreTabBtn.Count;

｝

/// 设置返回按钮的是否可选中状态

public void SetPreCanSel( )

｛

//当堆栈里数目大于1显示select按键，否则为unselect按键（略）

｝

public void SetPreFirCanSel()

｛

//当堆栈里数目大于1显示select按键，否则为unselect按键（略）

｝

/// 把以前的index存储在Stack里面方便调用

private void SetPreIndex(int index )

｛

m\_PreTabBtn.Push(index);

｝

/// 点击事件的具体调用

private void HandeleTabBtn( int index )

｛

//对应执行点击TabBTN的委托方法，以传进去的按键的索引为判断

GetCurIndex(index);

SetSelORUnSel();

//把点击的索引传进去

SetPreIndex(index);

//有点击传入的第一次，设置为可选状态

SetPreCanSel();

｝

// 点击返回按钮

public void HandeleRetBtn( int preIndex )

｛

//对应执行点击TabBTN的委托方法，以传进去的按键的索引为判断

GetCurIndex( preIndex );

SetSelORUnSel();

｝

/// 把TabBtn和对应的Action委托事件存储在列表里(略)

private void AddTabBtn( Transform rootTrans )

｛

｝

private void InitTabBtn()

｛

//因为暂时没有其他数据所以只能先手工设置第一次为0选中状态

SetSelORUnSel();

｝

/// 得到当前的按键索引为m\_CurTabBtnIndex赋值

private int GetCurIndex( int index )

｛

if (index < m\_TabBtnsList.Count)

｛

m\_CurTabBtnIndex = index;

return m\_CurTabBtnIndex;

｝

else

return 0;

｝

//设置按钮的选中状态或者未选中状态(略)

private void SetSelORUnSel( )

｛

｝

//action委托对应的执行方法

private void ActionToLabel( int index )

｛

｝

｝