# UI开发规范

|  |  |  |  |
| --- | --- | --- | --- |
| 作者 | 修订内容 | 修订时间 | 版本号 |
| YQ | 创建 | 2018-8-10 | V1.0.0 |

## UI框架背景

名字空间：YUIFramework

在做UI开发前，需要对UI框架有个大致的认知。咱们的UI目前分为两类，普通的UI（UIBase）和系统弹框（UIPopupMessageBox），Hierarchy中分别在不同的UI层下，由不同的管理器管理。

其中系统弹框一般负责项目里公用的系统弹框，如断线重连弹框等，是按优先级来选择弹出的，优先级相同的情况下，优先弹出提前进入队列的。

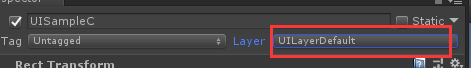
普通的UI又分为两种，状态UI和普通弹窗。状态UI当前最多只会显示一个，并且会入队列，即关闭状态UI的话会继续从队列里选择上个状态UI继续显示。普通弹框当前可以显示多个，关闭不入队列。状态UI面板可以添加附属UI，比如公用的UI顶条。

## 创建新UI

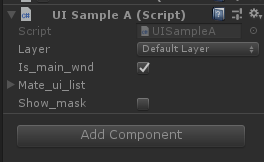
* 名字规范：AssetBundle的名字 == 界面GameObject的名字 == 脚本的名字
* 新开发的UI脚本需要继承UIBase



* 对应的UI预设设置的Layer应为“UILayerDefault”



* 将UI脚本挂在对应的UI预设上



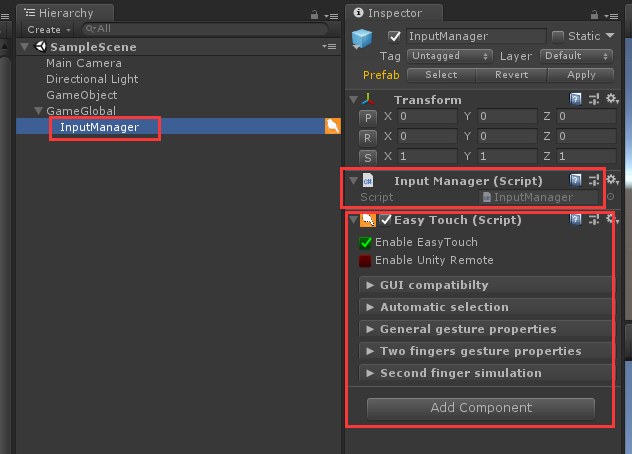
Layer：UI层，一般UI都设置为默认层。

Is\_main\_wnd：表示是否是状态UI。True是状态UI，当前仅有一个,并且关闭可以回退到上一个状态UI；False是普通的弹窗，可以叠加弹出。

Mate\_ui\_list：对状态UI有用，表示显示状态UI时同时附加显示的一些UI界面，这些mate ui一般作为公用的UI，例如顶条。

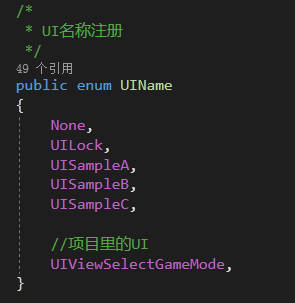
Show\_mask：True表示显示半透明遮罩，适合非全屏普通弹窗。这个同时会使得当前窗口模态显示，即点击不到下面的父UI。

* 运行时保证场景中有InputManager，并且Gameobject上挂载InputManager脚本和EasyTouch脚本



* 新开发的UI要在YUIFramework.UIRegister中注册

1. 注册UI名称



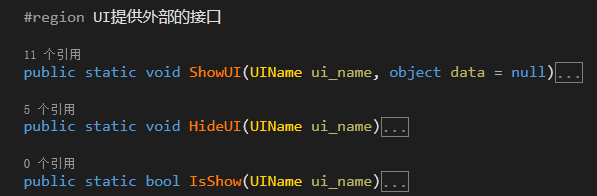
1. 注册UI详细信息

UI名称，UI预设路径，是否在显示之前加载数据。如下图



## 开发规范

* 目录管理。除了需要动态加载的UI资源，其他的比如atlas、材质、贴图，都不要放在Resources目录下
* UI脚本尽量不要提供Instance，防止对UI的过渡依赖以及对UI造成不安全性。与UI的交互可以见常见问题——[逻辑如何与UI交互](#_逻辑如何与UI交互)
* UI提供的接口，见UIBase。比如显示、隐藏这些基本接口，见UIBase “UI提供外部的接口”部分

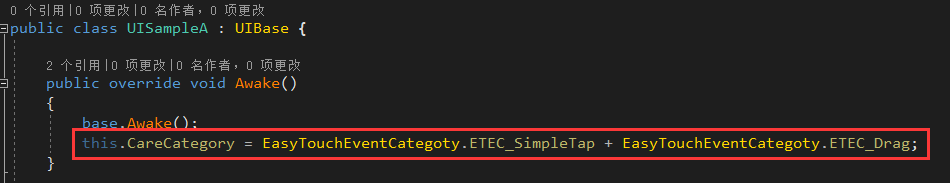


* 具体使用方式，可以参见示例UISampleA, UISampleB, UISampleC

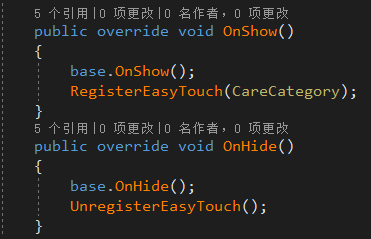
## 常见问题

### 如何使用多点触碰事件

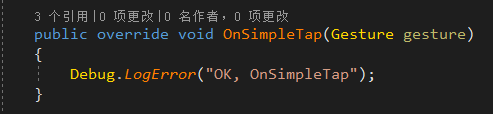
* Awake中设置CareCategory



* OnShow时RegisterEasyTouch，OnHide时UnregisterEasyTouch

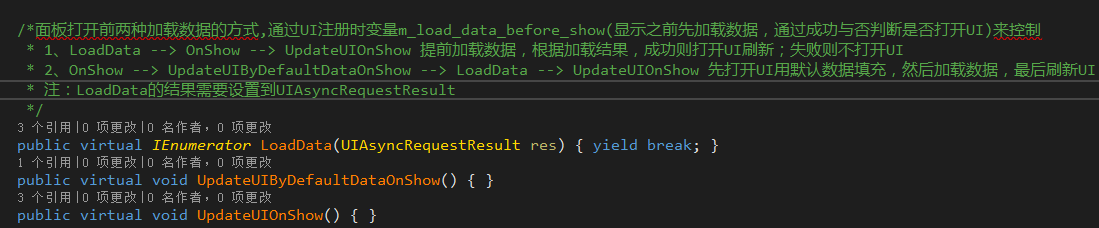


* 重写相关的回调事件即可



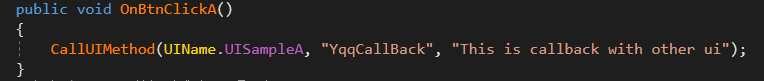
### 如何在打开面板前控制数据加载

我们经常会有这样的需求，在打开界面前先加载数据，根据加载数据结果，如果成功则打开UI，失败则不打开UI。在框架里，需要在给UI注册时设置YUIFramework.UIRegisterInfo.m\_load\_data\_before\_show为true，相关的重写函数如下图



### 已经打开的UI面板之间如何交互

用法：

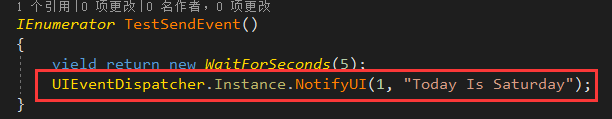


### 逻辑如何与UI交互

#### 局内

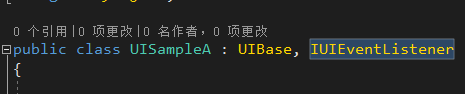
* + UI🡪逻辑：UI请求逻辑发送Command。每款游戏局内的实现会有区别。
  + 逻辑🡪UI：EventDispatcher。每款游戏局内的实现会有区别，所以还需要和局内系统接洽。

**发送事件的逻辑**

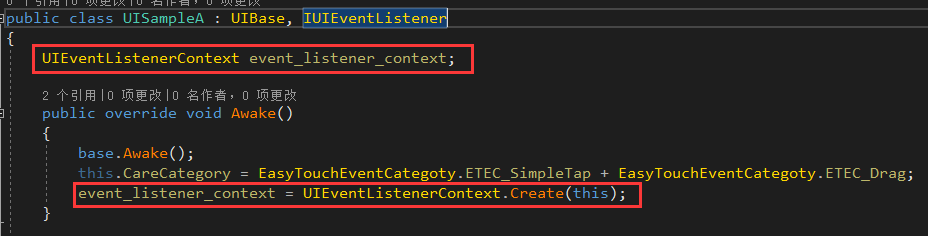


**接收事件的UI**

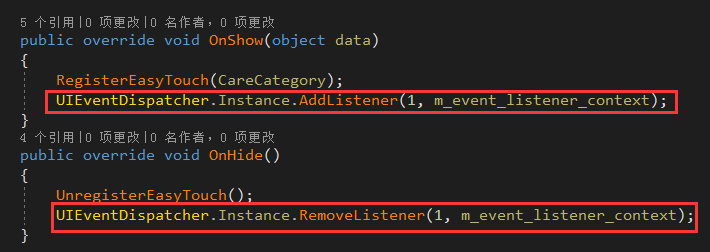
* 对应的接收事件的UI类需实现接口IUIEventListener



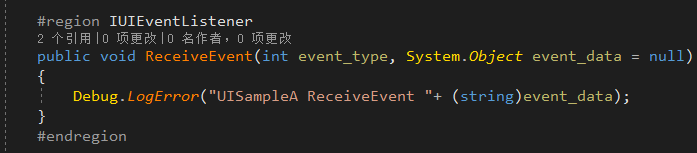
* 创建Context



* UI打开的时候往通知中心添加listener，关闭的时候从通知中心移除listener



* 在实现的统一接口里处理逻辑回调的事件



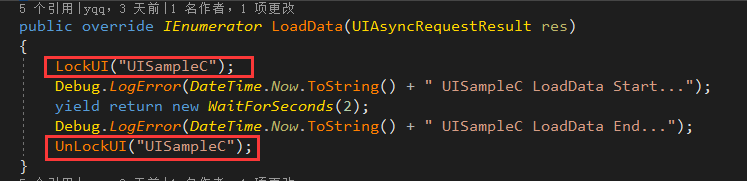
#### 局外

* UI🡪逻辑：协程里发送异步请求。此时请求等待回复时，为了防止等待的时间内有其他操作，要加锁控制。
* 逻辑🡪UI：逻辑回调UI，直接在上述UI请求逻辑的协程里完成。

### UI请求逻辑等待时如何防止对UI的操作？

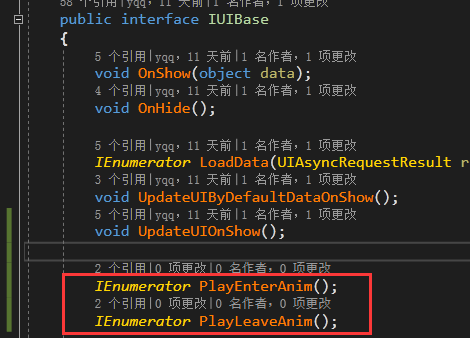
使用UI锁来处理。UI锁的界面暂时设置了一个空界面，项目里可以自己更换成别的样式，比如转菊花。

在对应的UI脚本(继承UIBase)里这样设置，解锁操作要和加锁操作成对，比如请求前加锁，请求完成解锁。



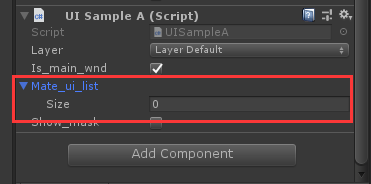
### 有UI面板动画流程么

有，子UI去实现这两个接口控制进场、离场动画，不重写则默认没有动画。



### 多个状态UI面板公用某些UI时，可以不用每个UI面板都加上这些UI么？

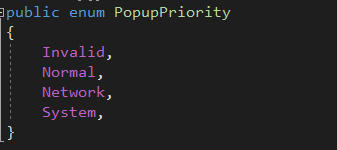
可以，将公用UI提取出来，作为单独的UI，然后在UI面板编辑的时候，将这些公用UI的名字填到状态UI 的Mate UI列表里。



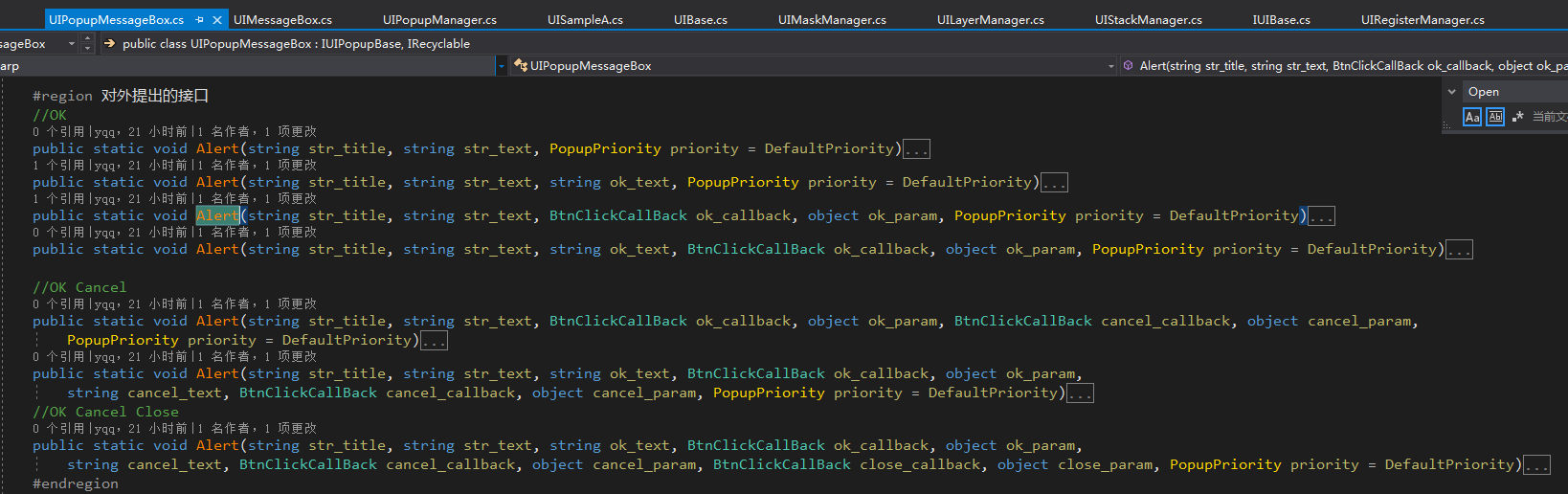
### 有系统提示框么

一般系统提示框是公用的，每个项目都可以做成自己的样式，Hierarchy中放在UISystemPopupLayer层。每个项目根据UI版本是NGUI或UGUI分别去扩展NGUIMessageBox或UGUIMessageBox，实现自己项目的系统提示框

系统提示框有优先级，当前只会显示一个，优先级高的先显示。



其（UIPopupMessageBox）提供的接口有：



## 其他UI开发注意事项

### Drawcall优化