**程序主体结构及注意事项**

|  |  |  |
| --- | --- | --- |
| **日期** | **修改人** | **描述** |
| 2016-08-05 | 钟国良 | 添加文档草案 |
| 2016-08-12 | 钟国良 | 添加UI框架说明 |

# 关于Wrap文件

## Wrap文件绑定

对于C#类需要在Lua层中使用时，ToLua需要对C#类对象进行映射关联，我们在生成绑定信息时，将C#类分成三大类：

1. 具体游戏模块功能类，比如所有处在GameScript目录内的C#类对象
2. 游戏通用引擎模块类，比如GEngineScript目录内的C#类对象
3. 原生UnityEngine类，比如UnityEngine库和UnityEditor库等

## 具体绑定操作

在Editor/ToLuaEditor目录内的GameBindType.cs就是用于绑定设置。

逻辑代码部添加了三个变量用于存储以上三类文件的绑定，比如：

1. GameScripteBinds用于绑定游戏模块类对象
2. GameEngineBinds用于绑定游戏引擎对象
3. UnityEngineBinds用于绑定与Unity原生有关的对象

具体绑定操作，参考工程代码。

## Lua层导入

ToLua已经不需要再对生成的Wrap文件进行手动require导入，但是对于存在namespace的类对象而言，我们在使用过程，仍会对类对应进行二次命名，方便Lua层进行使用，比如

GUILayout = UnityEngine.GUILayout

SceneLoader = GOE.SceneManager.SceneLoader

### 分类简化操作

Lua层的简化调用也会同Wrap文件的分类保持一致，分成三层。主体框架在初始化时，会自动加载一个名为tolua.lua的文件，我们的简化lua文件（GameScript.lua , GEngine.lua,UnityEngines.lua）会在tolua基本库加载完成后进行加载。

1. GameScript.lua 用于简化游戏具体模块类对象
2. GEngine.lua 用于简化游戏引擎类对象
3. UnityEngines.lua 用于简化原生Unity类对象

# 关于Lua调试测试

当我们需要独立于具体游戏功能模块，而又想使用部分引擎核心模块功能时，我们提供了另外一种测试方案，具体脚本使用TestLuaManager.cs

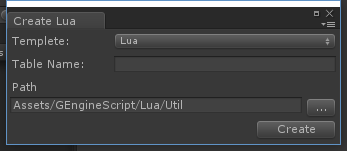
## 关于部分参数和功能说明

1. LuaPath , Lua文件的加载相对路径。比如：Assets/GameScript/Lua/Test/LoaderTest.lua 就可以使用 Test/LoaderTest.lua 。
2. Function, 默认执行的LuaFunction名称（可空） 比如：LoaderTest.Test1
3. 关于OnGUI()自定义集成，测试Lua代码中只要包含”OnGUI”方法，在TestLuaManager中就是自动检测，并进行调用。

**注意：Lua文件名必须和luaTable名保持一致**

## Lua脚本模板

在工程中我们提供了一种快速创建Lua脚本模板的方式。点击菜单栏中AssetTools/Lua选项，会弹出脚本创建界面，输入对应的文件名，并选择模板类型后，点击Create。



### 自定义Lua模板

自定义所需Lua模板，非常简单。只需要以下三个步聚

1. 在Assets/Editor/Script目录创建对应的模板文件Txt文件
2. 定义模板的名称，命名规则：模板名+Templete.txt
3. 模板内的Lua Table表名字符必须是tableName，这样才能进行字符替换

# UI框架说明

**注意：UI界面的Prefab名称必须同逻辑Lua脚本名保持一致**

## 生命周期



* 资源加载完成后，就会加载处理Page的逻辑，并调用OnAwake方法。

通常我们将界面的组件解析、界面组件事件绑定都放在OnAwake文件内

* OnShow/OnHide会在Page每次显示/隐藏时调用

消息系统中消息事件绑定，应该在OnShow方法中进行注册，当界面隐藏时，UI系统会自动释放当前Page中的消息列表

* OnDestroy只会发生在界面缓存销毁时，如果界面一直被缓存，将不会被执行。

## UIPage的界面设置

每个界面的显示都需要告诉UIManager管理器，自身的属性配置，UIManager才能正确分配界面显示的层级先后，显示模式，碰撞类型，是否需要返回时创建等信息。

### 界面属性

#### EPageType界面类型

目前界面类型分为三层，Fixed , Normal , PopUp，后期视情况再添加引导层

* Normal常规界面层，层次为0

Normal类型的界面，可以理解成一级界面，处理Fixed界面显示之下;并且当界面自动返回时，会根据EShowMode(显示类型)自动创建

* Fixed固定界面，层次为250

Fixed固定层，主要用于制作界面固定工具栏等，不会自动随关闭。

* PopUp 弹出界面层， 层次为500

PopUp类型的界面，会显示在Fixed和Normal层的最上层，并且PopUp类型的界面，不参与返回操作的自动创建

注：类型就是区分界面的显示层级

#### EShowMode显示模式

* None

普通显示，不影响其它界面

* HideOther

独占并隐藏其它同EPageType类型的界面

* NeedBack

如果界面被打开，在自动返回操作时，需要被创建

* NoNeedBack

如果界面被打开，在自动返回操作时，不需要被自动创建

#### ECollider碰撞模式

暂时未使用，后期考虑自动添加Page的遮拦碰撞

* None 显示界面不包含碰撞
* Normal 透明背景碰撞
* WithBg 非透明背景图碰撞

### 设置界面属性

在UI框架的Page基础对象内，统一封装接口SetPage()可以用来设置界面属性。

接口形式：

SetPage(EPageType type , EShowMode mode , ECollider collider)

示例：





## 关于界面Lua Table中的内置字段

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **字段名** | **描述** | **示例** |
| GameObject | gameObject | 界面的根结点 | Main={}  Print(Main.gameObject.name) |
| Transform | transform | 界面根结点的Transform对象 | Main.transform.localScale = Vector3.zero |
| LuaUIPage.cs | ui | C#层UIPage逻辑对象 | Main.AddClick(“btn01” , doSomething) |
| Event(events.lua) | event | Lua层的事件管理器 | Main.Event:AddListener(“eventID” , doSomething) |

注：更多使用方法，请查看项目示例或相关API文档

## 界面管理

UIManager.cs主要用于处理界面加载、缓存、切换等。常用API:

* UIManager:Show(uiPageName , pageData)

显示指定的UI界面 ， pageData 表示界面显示时的透传参数

* UIManager:Hide(uiPageName)

隐藏指定名称的UI界面

* UIManager:AutoBackPage()

自动返回上层界面

更多API接口，请查看API详细文档

## 界面组件点击事件

### 按钮Button事件绑定

LuaUIPage.cs对象提供了快捷的组件查找及点击事件绑定

* AddClick(string goHierarchy , Luafunction luafunc)
* AddClick(GameObject gameObject , Luafunction luafunc )
* RemoveClick(GameObject gameObject)
* RemoveClick(string goHierarchy)

### 普通组件点击

Todo 暂不支持

## 消息监听

Page界面在初始化时，会自动给Lua层的界面LuaTable添加事件管理对象event。Event对象的具体对应Lua文件为events.lua

### AddListener消息注册绑定

假设存在一个MainUI的界面,那么消息的注册如下：

MainUI.event:AddListener(“eventId” , function() print(“on event here”) end )

或者

MainUI.event:AddListener(“eventId” , MainUI.dispatchEvent);

注：在调用AddListener时使用的是“：” ，所有event对象内的方法都是采用”:”调用

### RemoveListener删除消息

如果在某些情况需要手动删除消息时，就可以调用RemoveListener方法。比如：

MainUI.event:RemoveListener(“eventId” , MainUI.dispatchXXX)