NGUI培训

# 概述

项目的UI使用NGUI制作，目前版本为3.9.4。本文档为客户端UI的培训，主要从简单UI制作，进阶组件，逻辑脚本几个方面来讲解。

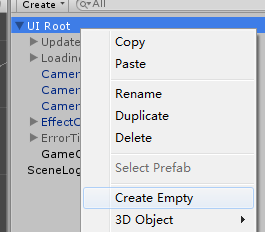
# 简单UI制作

因为客户端架构关系，客户端UI需要在LaunchScene中制作和编辑。

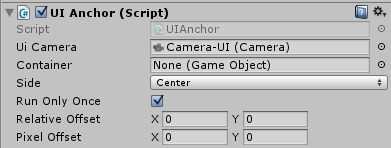
如果是新建工程，则首先需要创建UI Root，然后在UI Root下制作UI。目前LaunchScene中已有UI Root，所以直接在其下面制作即可。

## 创建根节点

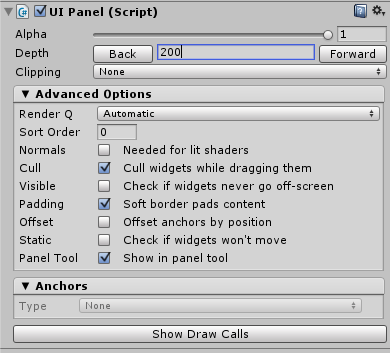
直接在UI Root右键，选择Create Empty，在UI Root下创建一个空的GameObject节点，单击或右边Inspector改名为自己需要的。

制作新的UI需要为其加锚点组件，默认居中。在Inspector标签下点击Add Component，搜索Anchor，添加UI Anchor组件。Side表示锚点位置，Relative Offset和Pixel Offset为位置偏移，可根据需要设置，其他设置项不需要改变。



如果制作的是二级UI，即弹出的窗口、提示等，则需要添加UI Panel组件。在Inspector标签下点击Add Component，搜索Panel，添加UI Panel。UI Panel的作用是把它下面的所有节点作为一个单独的渲染批次，可以理解每个UI Panel下为一个渲染单位。主要需要设置Depth值，Depth会影响各节点之间的遮挡。目前项目规划，普通的弹出UI，如背包、属性界面，Depth为200-300；各类Tips，为400-500；消息提示，如获得装备提示，600-700；其他特殊UI，如死亡复活等，按需要设置。



根节点挂载了UI Anchor后便不能改变Transform的值，如果需要调整根节点的位置，UI Anchor的两个微调不够灵活，则可以视需要在根节点下增加一个Offset节点，专门用来调整位置偏移。



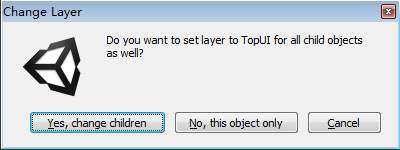
Unity中每个GameObject都有自己的层级Layer，UI默认的Layer为“UI”。UI共有三个层级，先后深度关系为UI < TopUI < TopTopUI。所以比较游戏画面中某两个UI的层深，需要先比较Layer，如果Layer相同再比较父节点的UI Panel。



各Layer的规划，一般默认情况下大多数为UI；有些UI需要遮挡住预览模型，还有挂在UI上的特效，则需要为TopUI；剩下最特殊的，如在TopUI层级的UI上显示了预览模型，同时弹出了UI需要遮挡住该模型，则需要为TopTopUI，目前该类型只有宠物Tips上点开的宠物技能Tips需要此设置。

上文说的Depth规划，是针对于在UI层级下的规划，其他层级视情况而定。

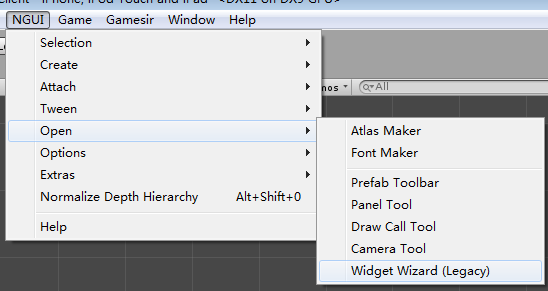
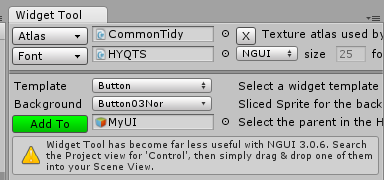
如果改变层级，且该节点下有子节点，则会弹出提示是否将所有子节点的层级同时改变，需要注意不要改错。



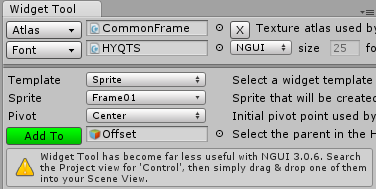
至此，UI根节点创建完成。

## 添加图片

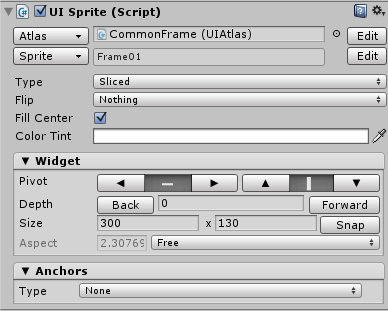
首先打开NGUI->Open->Widget Wizard，弹出的窗口为制作UI的主要窗口。

选中Offset节点，Widget Tool窗口中设置。Template为组件类型，选择Sprite。Altas表示图片所在的图集，UI需要的图集由策划负责制作好，此处选择CommonFrame。Sprite选择需要的图片，Pivot为图片的锚点，默认Center。设置好后点击Add To按钮添加。

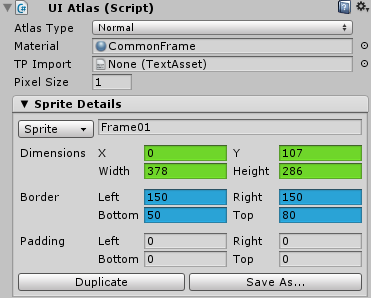
 

点击后，图片添加完成。Game视图中可看到图片。其本质是创建Sprite节点，上面挂载着UI Sprite脚本，所以同样可以通过创建空节点，点击Add Component添加。

脚本常用设置：Type，Sprite类型，分为Simple（简单贴图，图什么样显示就什么样），Sliced（按九宫格切割图片的边界，再根据Size来拉伸图片，具体点上面Edit可进行编辑），Tiled（把图片作为一个单元，按Size重复填充图片），Filled（可以一定方向来截取显示一部分图片，如技能CD的计时）。

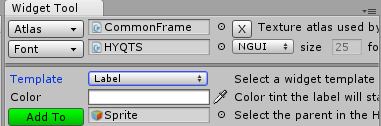
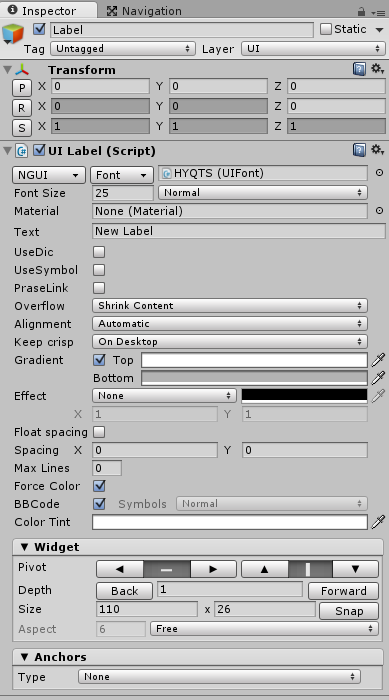
点击Edit如图，其中Dimensions表示切割范围，改变Size的宽时则区域1、4、7和3、6、9固定，中间2、5、8的部分拉伸显示。Border和Padding表示Atlas中切图的范围，不用过多关心。

脚本常用设置：Flip，图片的镜像偏转，一般不需要设置。Fill Center，填充中心，不需要设置。Color Tint，图片颜色，一般图片为美术制作好，设置保持白色即可。Pivot，锚点。Depth，深度，和前面提到的Layer，Panel，Depth优先级由高到低联合决定图片的深度。Size，图片的大小，Snap按钮为恢复图片在Atlas中的原本尺寸。

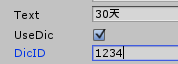
## 添加文字

文字也可以从Widget Tool中添加，选中Offset节点，Widget Tool的Template中选择Label，字体Font选择默认的HYQTS，点击Add To。

脚本常用设置：最上面的NGUI和Font按钮是可以选择字体类型和字体文件，目前项目使用NGUI字体，不使用Unity字体。Font Size，字号，后面可选择整个Label的格式，普通、粗体、斜体、粗斜体。Material，文字纹理，不需要配置。Text，文字内容，不允许UI中出现中文，中文交由字典实现。

UseDic，项目添加的选项，勾选后可以在DicID中配置字典号，自动读字典表将内容设置到Text中。需要注意如果勾选了UseDic，逻辑中再改变该Label的text会和DicID冲突。



UseSymbol，项目添加的选项，勾选后可以在SymbolAtlas中配置图集，在游戏运行时会自动检索文字中是否包含符合SymbolAtlas的图片的标记，进行图文混排。



PraseLink，项目添加的选项，勾选后可以在LinkItem中设配置预置GameObject，在游戏运行时会解析文字中的链接，并用LinkItem实现。

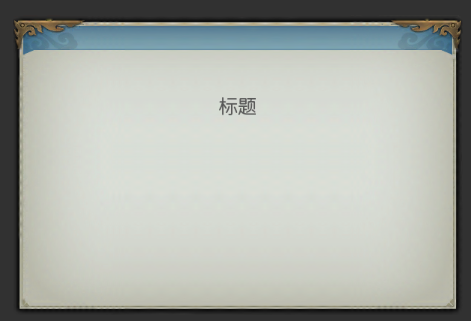


Overflow，Label类型。Shrink Content（固定Label大小，文字达到Label边界时自动缩小）， Clamp Content（固定Label大小，文字达到Label边界时不再显示），Resize Freely（完全不限制Label大小，随文字内容自动扩大缩小），Resize Height（限制Label宽度，不限制高度）。选择Clamp Content时会多出选项Use Ellipsis，勾选后会在文字达到边界时自动补足省略号。



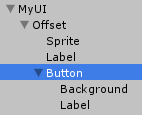
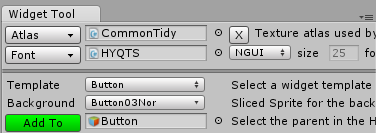
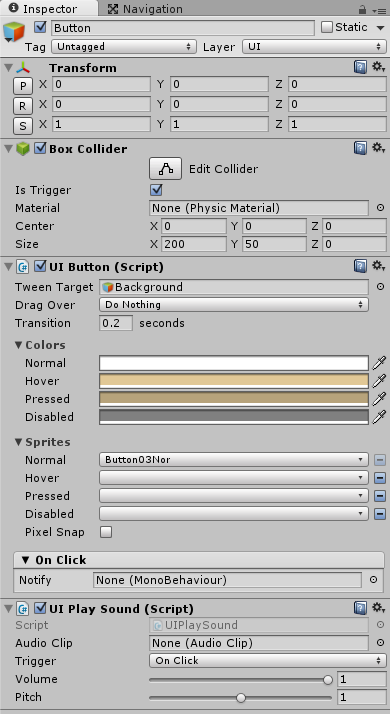
Alignment，对齐方式，Authentic，自动；Left，左边；Center，居中；Right，右边；Justified，合理的自动调整。Keep crisp，动态字体锐化，目前不需要配置。Gradient，颜色渐变。Effect，添加描边，3D阴影等文效果。Spacing，横纵间距。Max Lines，最大行数，0表示不做限制。Force Color，项目添加的选项，文字严格按照颜色码变色，不会和Color Tint的配置叠加。BBCode，是否支持文字效果码，勾选后可输入[c][b]等改变一段文字的显示方式。Color Tint，Label的整体颜色。Pivot，Depth，Size和Sprite的一样配置。

至此，UI效果如图。



## 添加按钮

按钮依旧从Widget Tool中添加，选中Offset节点，Widget Tool的Template中选择Button，此时Atlas需要设置，如果按钮有文字则还需要设置Font，点击Add To添加。

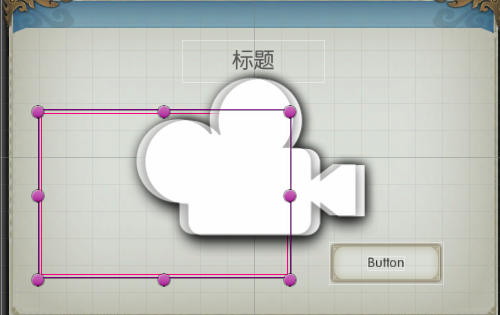
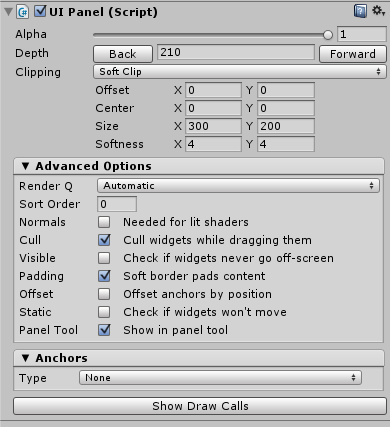
添加按钮会同时三个节点，Button为按钮根节点，同时挂在Box Collider（点击判定范围），UI Button（按钮脚本），UI Play Sound（播放音效脚本）；下面的Background和Label为按钮的底图和文字。

脚本常用设置：Box Collider，Center和Size表示判定框的中心位置和大小，其他不用更改，Is Trigger可以勾掉。UI Button，Colors和Sprites表示按钮处于普通、悬停、按下、失效四个状态时的颜色和图片。On Click为按钮点击的事件函数，需要挂在程序逻辑脚本后配置。UI Play Sound，一部分按钮音效需要配置在UI上，此处只配置Audio Clip即可。

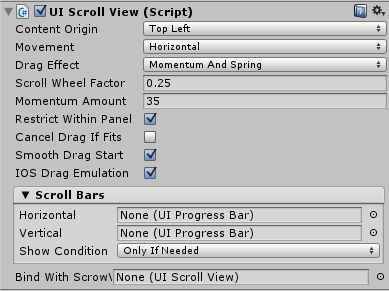
此处建议，Button下的Background和Label如果没有特殊需求，Transform都保持默认值0。

## 添加拖动列表

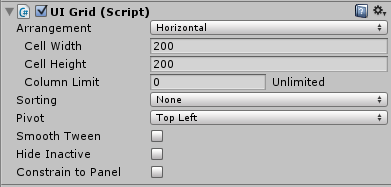
拖动列表无法通过NGUI直接添加，需要单独制作，其本质是一列可以拖动的Item，显示在一定剪裁区域中。剪裁需要用Panel来实现，即制作拖动列表需要为其单独添加一个Panle。因为显示在UI上，Panel的Depth需要比根节点Panel的Depth大，否则会被前面加的图片挡住。Clipping选择Soft Clip，下面Size为剪裁区域，可直接在Scene视图中编辑，Softness为边界渐变区域，主要配置这两个选项即可。

在此之后，在同一节点下添加UI Scroll View，负责管理列表的拖动，需要关注的配置只有Movement（表示列表的拖动方向，Horizontal水平，Vertical竖直），Cancel Drag If Fits（如果列表不足填满剪裁区域，是否取消拖动，一般都会勾上）。下面可以配置进度条联动，不过NGUI进度条有bug，一般不配置。

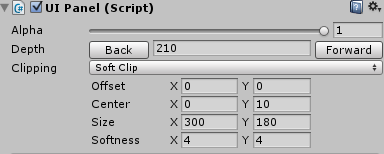
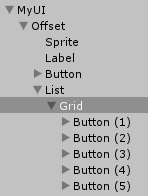
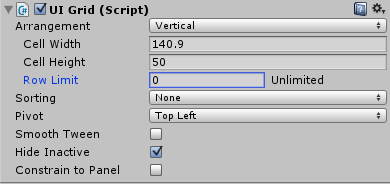


列表的Item需要自动排列，在Panel下添加一个Grid来管理Item的的排列，如果Item尺寸不同则需要添加Table来管理，以Grid为例。

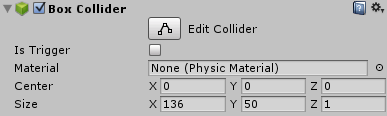
 

脚本常用设置：Arrangement，Item的排列方式，Horizontal水平，Vertical竖直。Cell Width，Cell Height，表示单元格的宽度和高度，Grid实际上是根据单元格来进行整理，并不完全按照Item的尺寸来整理。Column Limit，表示每一行/列的数量上限，该配置和Arrangement联系，如果是水平排列，则表示每行的数量上限，如果是竖直排列，则表示每列的数量上限，0表示无上限。Sorting，Item的排序，常用的有两个Alphabetic，按Item名字字符串排序，Number，项目内添加，如果Item名字是数字时按数字大小排序，主要用于区别如名字为“2”和“10”的两个Item的排序情况。Pivot，锚点。Sommth Tween，重排时各个Item是平滑展开，或直接设置位置。Hide Inactive，如果列表中某个Item被隐藏，是否要把它的位置补齐。

拿前面的Button当做列表的Item，复制多个，全部放到Grid下，再进行一些设置后得到如下效果。

至此Item已添加，但还无法响应拖动，为每个Item手动添加Box Collider（响应一切事件都需要Box Collider）和UI Drag Scroll View。可以保留一个Item，剩下的删除，加完脚本后重新复制够数量的Item。UI Drag Scroll View不需要设置，Scroll View属性会在运行时自动找到父节点中第一个UI Scroll View。

至此，拖动列表制作完成。

## 保存

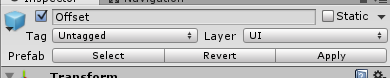
工程中的所有UI都需要保存成Prefab，UI的Prefab保存路径为Assets/Game/Resources/Bundle/UI/Prefab。



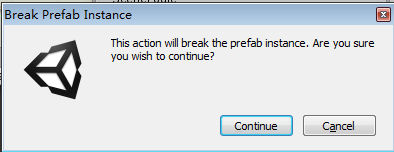
保存一个新的Prefab时，直接把UI根节点拖拽到该文件夹下，再Ctrl+S即可，保存后UI节点变蓝色，且文件架构中可以看到它的Prefab。

以后如果修改UI，则点击Apply按钮，再Ctrl+S即可。



如果删除Prefab中的某个GameObject，改变Prefab中结构，改变某个GameObject上脚本的顺序，都会破坏Prefab的结构，将Prefab打散。此时编辑完成后，选中原有节点，点击Apply按钮，再Ctrl+S即可。



切记和Prefab有关的任何操作，完成后都需要Ctrl+S。

UI中的拖动列表Item，基本都是不固定数量，显示数据用的。所以大部分列表中的Item不应该保存在UI中，而是应该单独保存，由程序在逻辑中控制加载数量。将Grid下的Item取一个出来，其他的删除。



Item的Prefab有自己的保存路径，Assets/Game/GameRes/UI/Prefab，保存方法和前面一样。



# 进阶组件

## UI Event Trigger

灵活的事件配置脚本，包含各种事件触发器。从上到下依次为：悬停进入，悬停移出，按下，抬起，获得焦点，失去焦点，点击，双击，拖动开始，拖动中，拖出，拖动结束。其中点击，按下，抬起比较常用，悬停和焦点手游用不到。

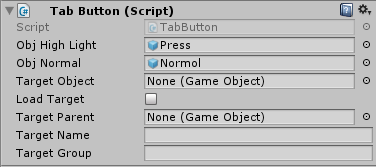
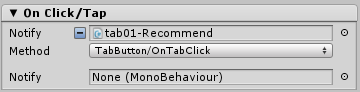


## 单选标签分页

相关脚本：TabController.cs TabButton.cs

参考Prefab：Shopping-ShangCheng



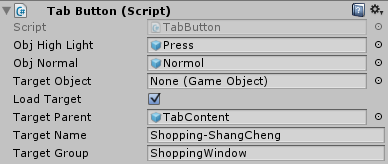
主要作用为控制分页的显示，一个TabController管理下面的所有TabButton。每个TabButton配置高亮和普通状态下的显示，点击事件函数必须为自身TabButton的OnTabClick。逻辑中可以在分页切换时接收事件，控制界面的逻辑。TabController的Start Select Tab可配置默认高亮标签索引，需要程序来决定选择时配置-1。

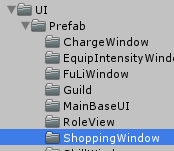
## 单选标签动态加载

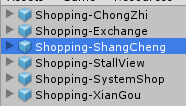
参考Prefab：ShoppingMall

一些大型UI通常包含单选标签，此时如果把每一页内容全部保存在UI中，会大幅增加第一次打开UI需要的时间。这种情况下可以改变TabButton中的一些参数，然后将每一个分页内容单独保存成Prefab，在点击标签时动态加载。

勾选Load Target选项，表示需要加载分页。Target Parent表示加载后的分页的父节点，Target Name表示分页Prefab的名字，Target Group表示分页的组名。分页Prefab文件不可和其他UI的Prefab放在一起，需要单独管理。路径还是Assets/Game/Resources/Bundle/UI/Prefab，但需要放在以Target Group命名的文件夹中。







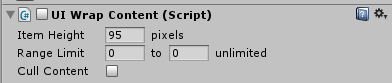
仅仅是改变了加载部分的逻辑，其余的逻辑和普通的单选标签完全一致。脚本内部判断某分页是否已被打开过，没有则开始加载，加载完成后把当前显示分页隐藏，如果已有则直接显示。每一个分页在加载一次后不再卸载，直到UI关闭，可以提高UI的打开速度。

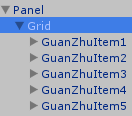
## 循环列表

参考Prefab：ContentPerfusion

一些拖动列表需要加载大量的Item，如果按照普通的加载方式会消耗很多时间。循环列表的作用，就是用一页的Item，随着新Item的划入和老Item的划出，循环设置Item显示所有数据，可以大幅减少加载时间。





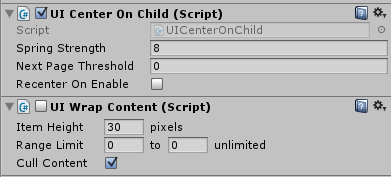


与普通的加载方式不同，此时需要在列表中放置刚好一页的Item，且不挂载UI Grid，手动整理好对应的位置。List Wrap Controller为项目制作的脚本，控制列表循环的相关逻辑。循环设置内容的核心脚本为UI Wrap Content，挂在List Wrap Controller的下一级节点，Item Height为Item的宽度/高度。按照图中设置后，逻辑中需要调用相关接口。

## 数字拨盘

参考Prefab：TeamRoot

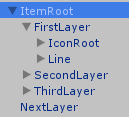
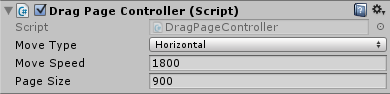
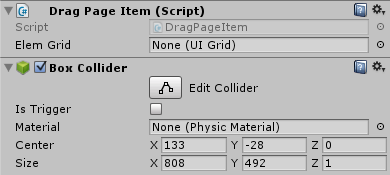
和循环列表的原理大致相同，主要用来显示数字，模仿手机中的拨盘控件。大致配制方法和循环列表一样，需要在UI Wrap Content的节点增加一个UI Center On Child，用来获取当前选中的Item的值。



## 拖拽翻页

参考Prefab：SkilInfoRoot

在技能、聊天表情等界面，需要分页显示大量数据，拖拽翻页美化UI效果。在拖拽翻页后，逻辑中可以获取当前页码。Drag Page Controller脚本控制下一级每个页面的移动，Drag Page Item是每一个分页，主要用于响应拖动事件。同时每一个分页中可能会排布另外一些Item，如表情加载，此时可以在Drag Page Item中添加Grid，并配置到Drag Page Item的Elem Grid上。目前逻辑中只有聊天表情用到此处。

# 逻辑脚本

## 注册

如果是新增的UI，需要现在客户端代码中注册才可以打开和关闭。在UIPathData.cs文件中注册。

public static UIPathData MyUI = new UIPathData("MyUI", UIPathData.UIType.TYPE\_POP);

UIPathData第一个参数为之前制作的UI的Prefab文件名，第二个参数为UI类型。UI类型分为：

TYPE\_ITEM，一部分用UI实现的会重复使用的显示单元，如名字板、伤害板，并不算是真正意义上的UI。

TYPE\_BASE，主界面，如头像、任务追踪等。因为之前UI使用Bundle加载，注册函数和其他UI有些差别，需要在Prefab名处填写老Bundle组名“MainBaseUI”，类型后填写的才是真正的Prefab名，且UI的Prefab需要放在对应目录Assets/Game/Resources/Bundle/UI/Prefab/MainBseUI中。

TYPE\_POP，普通的弹出UI，如玩家属性、强化、活动界面。所有TYPE\_POP类型的UI互斥，如果当前已打开TYPE\_POP的UI，打开一个新的TYPE\_POP的UI会将老的关闭。

TYPE\_POPCENTER，同样也是弹出UI，但可以和同类型以及TYPE\_POP共存不互斥，目前该类型只有PopMenu弹出菜单。

TYPE\_STORY，起先是剧情UI专用，打开时会将其他所有UI的父节点都隐藏用来展示剧情，目前也有同样需要隐藏其他但和剧情没关联的UI使用。

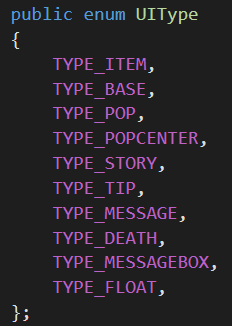
TYPE\_TIP，展示Tips的UI，如物品Tips、装备Tips等，不互斥。

TYPE\_MESSAGE，消息提示UI，如屏幕中央提示、获得物品提示等，不互斥，一般不屏蔽UI外面的点击。随客户端版本进行，TYPE\_TIP和TYPE\_MESSAGE的差别基本越来越小。

TYPE\_DEATH，ReliveRoot复活界面专用，同样会隐藏一系列UI父节点。

TYPE\_MESSAGEBOX，带按钮的消息提示框专用，TYPE\_MESSAGEBOX内互相互斥。

TYPE\_FLOAT，起先录屏、锁屏等功能的UI，可以拖动，目前和其他类型没有什么关联。

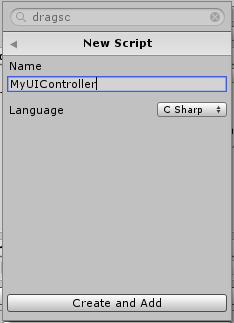


注册的构造函数还有两个参数bool bDestroyOnUnload = true和bool \_isCloseChangeScene = true。前者是关闭时是否卸载，一般某些比较大的UI加载起来较慢，打开一次后关闭可以选择卸载或设为隐藏；后者是切换场景时是否关闭。

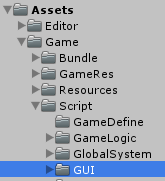
注册后可以调用UIManager.ShowUI(UIInfo.MyUI);和UIManager.CloseUI(UIInfo.MyUI);来打开和关闭UI。

## 添加脚本

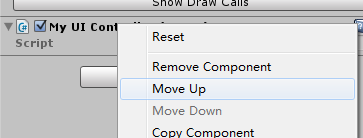
注册后可以在UI上添加逻辑脚本实现功能，一般来讲每个UI的根节点上都要挂载一个主逻辑脚本，下面的子节点可视需求添加。Inspector标签点击Add Component，选择New Script，输入脚本名，语言选择C Sharp，点击Create and Add。

新添加的脚本会创建在Assets根目录，需要把它移至客户端UI脚本的目录Assets/Game/Script/GUI，该目录下具体位置不限。



新添加的脚本会在GameObject的最下面，可以把它移至最上面方便以后的更改，右键MyUIController，选择Move Up。此时Prefab会被打散，挪好位置后Apply，Ctrl+S。



新创建挂载的脚本会在文件中自动创建同名MonoBehaviour类，需要注意的是，脚本的文件名和类名必须保持一致，否则Unity编辑器内会挂载错误。

脚本添加后可以在脚本中定义public变量，来控制UI上的任何组件。如：



此时Prefab上挂载的脚本会同步变化（需编译通过，Unity右下角有进度条）。



把需要控制的包含UI Label的GameObject拖拽到此处，脚本中的m\_TitleLabel便可以控制该UI Label，在拖拽前m\_TitleLabel为null。



脚本中的public变量的类型，只能选择可以挂载到GameObject上的，表示自动从该GameObject上取得第一个此类型组件。如果该GameObject挂有多个该类型的脚本则会有问题，不过一般不会出现挂多个同类型脚本的情况。如果变量类型是GameObject则直接操作该GameObject。

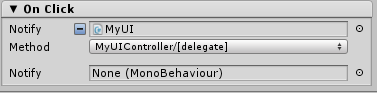
此时添加代码m\_TitleLabel.text = "1234";，等同于在UI制作时把text内容设置为“1234”。需要注意如果Label外部勾选了UseDic，此时脚本中修改text会和字典号DicID冲突。所以如果逻辑需要脚本中动态设置Label的text，则Label外部不可勾选UseDic。

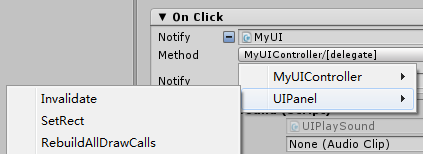
## 事件函数

前面添加的按钮可以加入点击事件。

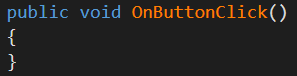


此处的Notify需要拖拽进来一个MonoBehaviour类，实际上就是需要一个挂着脚本的GameObject。拖拽MyUI节点进来后多出Method，可在此处选择事件函数。

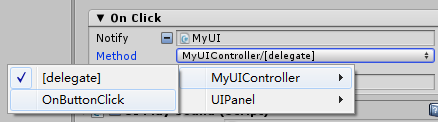




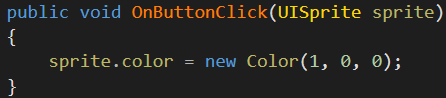
该GameObject上挂载的所有脚本中的，任何public void函数都可以当做事件函数。在MyUIController中添加函数。



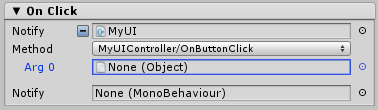
此时在Method处会增加新添加的函数，选择后当该按钮被点击时就会调用此函数。



事件函数可以带参数，参数类型为GameObject或其他MonoBehaviour类，和public变量类似。如该函数，执行后会设置sprite的color为红色。



此时外部On Click会同步改变。



像配置public变量一样，拖拽一个挂载着UI Sprite的GameObject，再选择UISprite作为参数即可。

