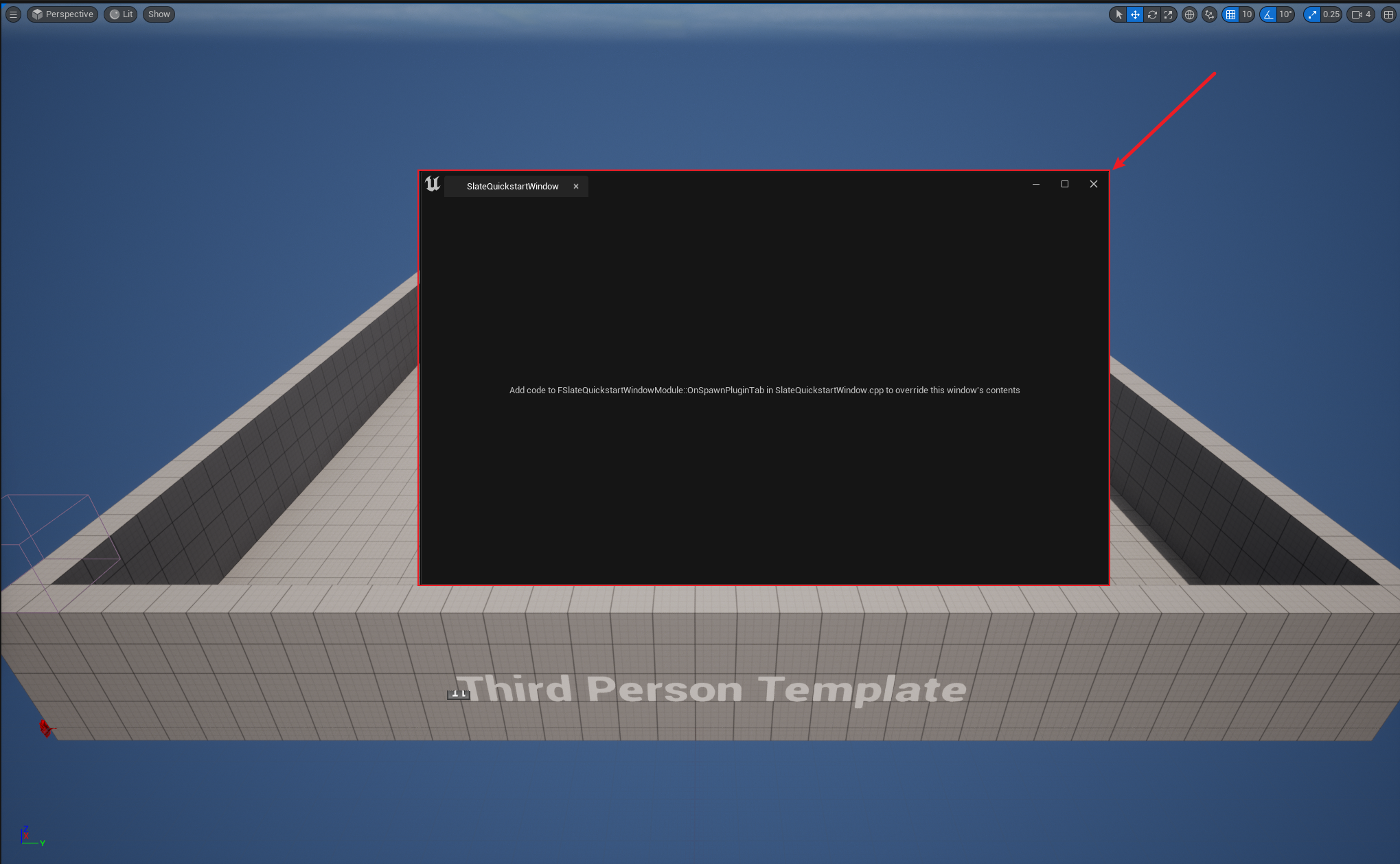
Snew(SDockTab)

新命名一个DockTab，是一个可停靠的窗口

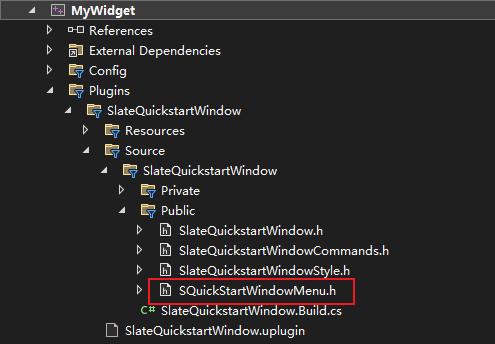


HAlign与HAlign参数是用于对齐作用。

你可以直接将Slate控件添加到插件向导生成的 FSlateQuickstartWindowModule 类。但是，如果你这样做，你在处理控件的回调时将遇到一些限制。因此，你应该创建专用的Slate控件来保存菜单的内容，然后将该控件添加到 FSlateQuickstartWindowModule 类。

所以在创建类的时候，要创建成SlateWidget，并且Module要选择为之前创建的SlateQuickstartWindow(Editor),而不是runtime，否则对于插件来说的话，其代码会被添加到游戏的代码中，而不是Plugin。

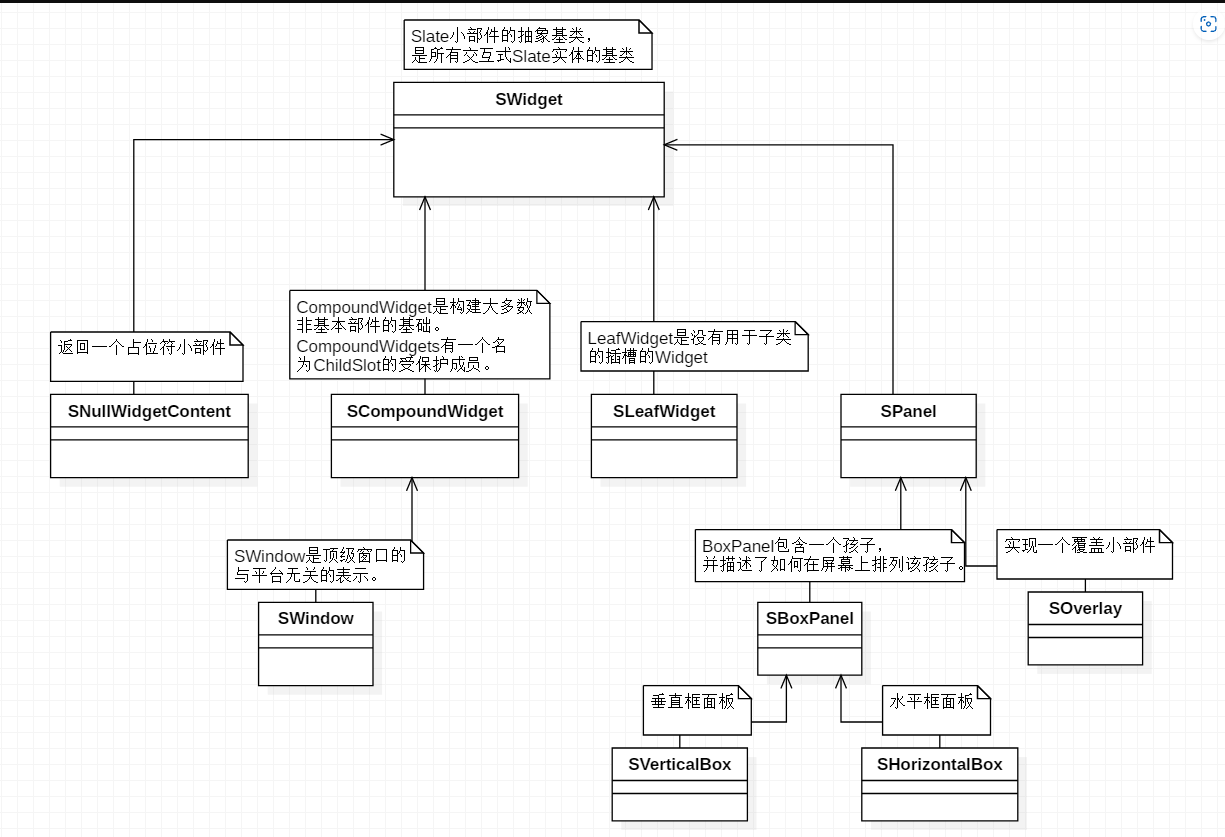
这是在Editor里面的SlateWidget.cpp



而这是放在runtime下面的SlateWidget.cpp



这样就无法被Plugin所识别到。



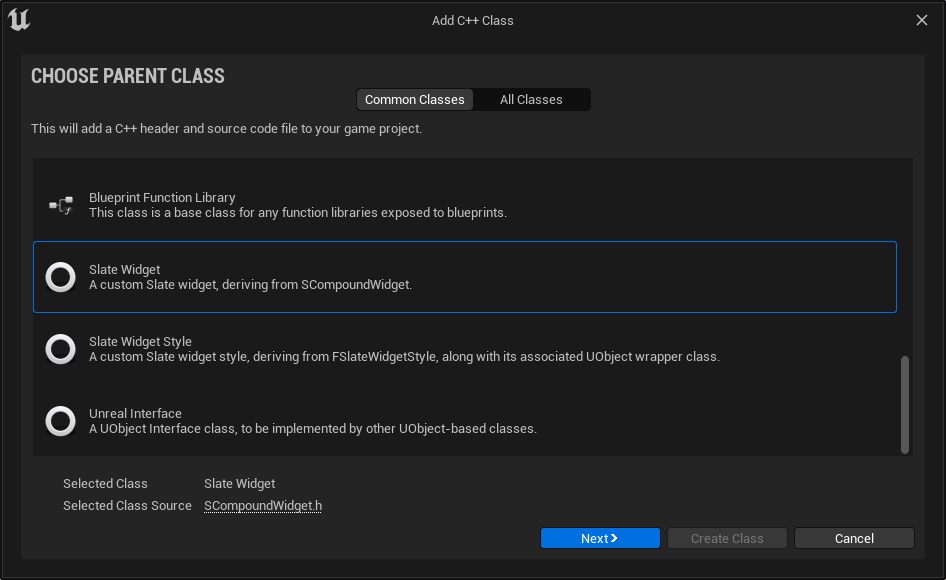
（网图，需确认）

[UE4 Slate三 SlateUI代码讲解 - 腾讯云开发者社区-腾讯云 (tencent.com)](https://cloud.tencent.com/developer/article/2157179)

创建widget应该继承自哪个类？SCompoundWidget，SPanel，SLeafWidget

Slot：插槽，就是说我们在控件下面可以添加多少个子控件，决定了我们用哪个类

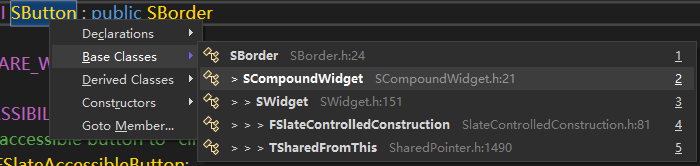
**SCompoundWidget**

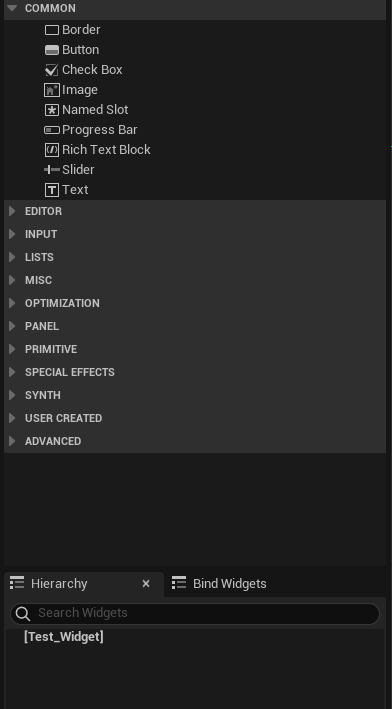


可以看到SlateWdiget是继承自SCompoundWidget的，

我们可以发现SCompoundWidget中含有一个ChildSlot，说明可以添加一个children widget

单个Slot的类，单个插槽。SUserWidget 或者 SCompoundWidget有一个插槽， 固定一个 比如 SCheckBox，SButton, SBox SUserWidget是继承自SCompoundWidget，这两个我们都可以继承，引擎更多的是继承自SCompoundWidget。其实引擎推荐我们自己写的单个插槽类的时候是继承自SUserWidget。





可以直接拖拽下来创建到我们的widget上面

## 2.2>基类:SPanel

多个Slot的类，多个插槽。SPanel: 多个插槽 比如SVerticalBox, SHorizontalBox，SScrollBox

可含有多个Widget，这个类本身没有加Slot结构而是继承于他的子类会去添加自己的Slot结构，并且实现不同的排列方式；



如Soverlay



SBoxPanel

文本

描述已自动生成

## 2.3>基类:SLeafWidget

没有插槽, 比如STextBlock



# 3>该如何新建一个S类组件呢？SNew和SAssignNew

在UMG里面我们是UButton\* pBtn = NewObject, 那么Slate里面我们是如何构造控件？ 可以把SNew和SAssignNew就理解成我们平常调用的NewObject，只不过它在slate里面就是要这么写。因为Slate都是S类，非U类。从这点也说明了我们头文件中为什么都是智能指针包着的S类。

引擎demo代码

/\*\*

\* Slate widgets are constructed through SNew and SAssignNew.

\* e.g.

\*

\* TSharedRef<SButton> MyButton = SNew(SButton);

\* or

\* TSharedPtr<SButton> MyButton;

\* SAssignNew( MyButton, SButton );

\*

\* Using SNew and SAssignNew ensures that widgets are populated

\*/

比如我们要新建一个Button，分别用SNew和SAssignNew写一下

TSharedRef<SButton> MyButton = SNew(SButton);

TSharedPtr<SButton> MyButton;

SAssignNew( MyButton, SButton );

直接说比较含蓄，需要写一下才知道 SNew：

1>返回值不同：返回共享引用

SAssignNew：

1>返回值不同：返回共享指针

2>使用方式不同：链式编程中直接获取值，直接赋值，在链式编程中想获取值就用SAssignNew

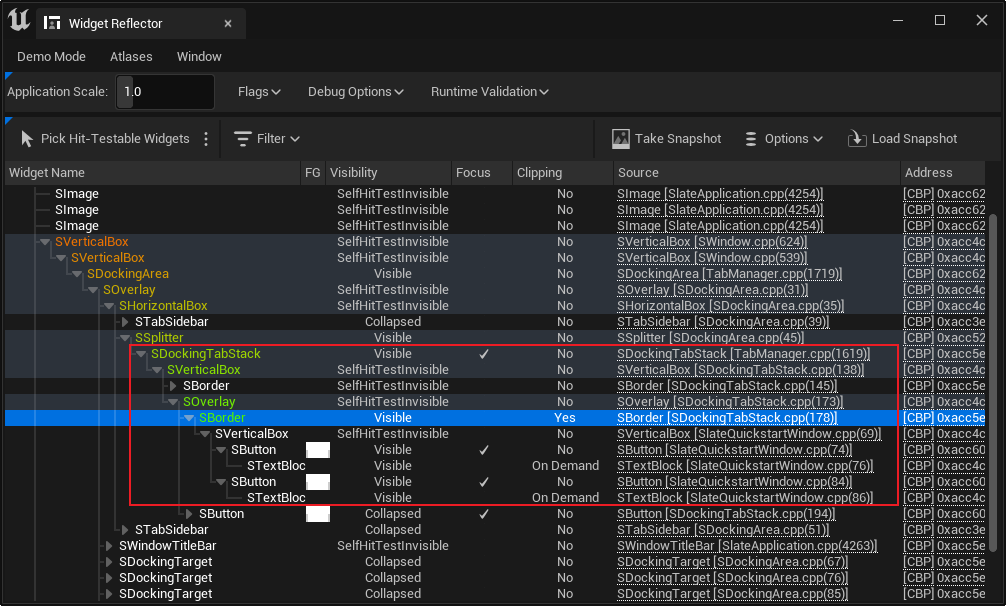
**3.2>链式编程优缺点**

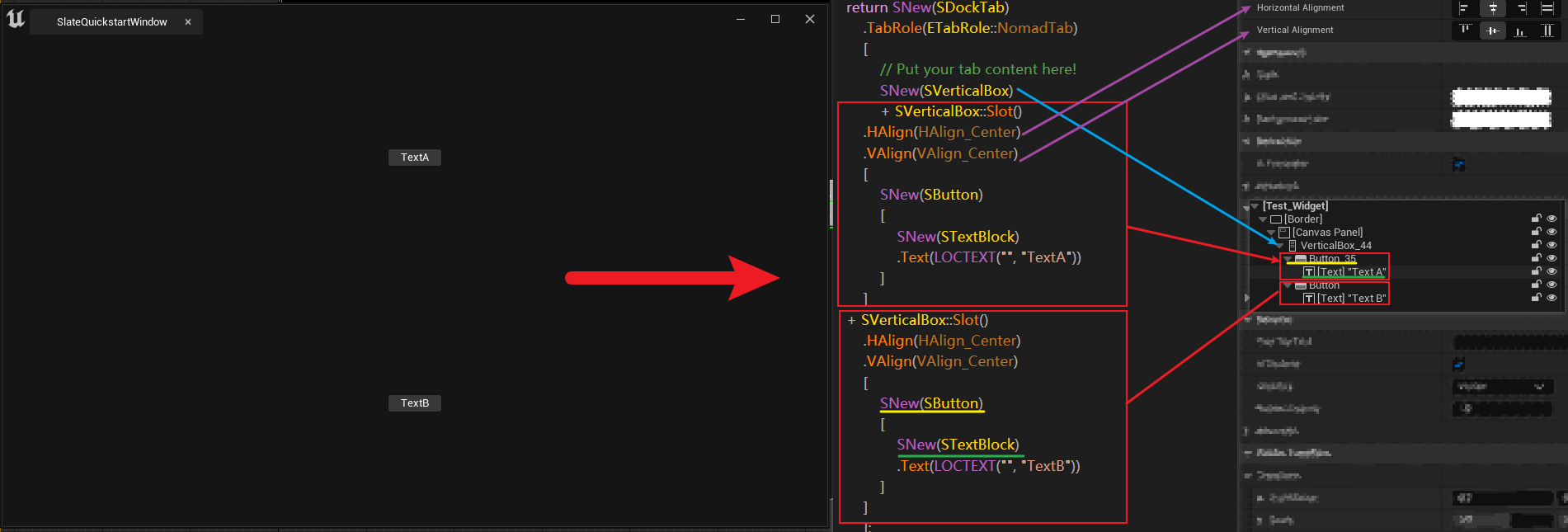
优点： 1>效率比UMG要高，因为UMG封装的就是Slate 缺点： 1>不能断点调试，断点无法命中链式内部 2>编写界面制作麻烦且不易维护

## 链式编程：链式编程就是将多个方法用点语法链接起来,让代码更加简洁,

## 优点： 1>效率比UMG要高，因为UMG封装的就是Slate 缺点： 1>不能断点调试，断点无法命中链式内部 2>编写界面制作麻烦且不易维护

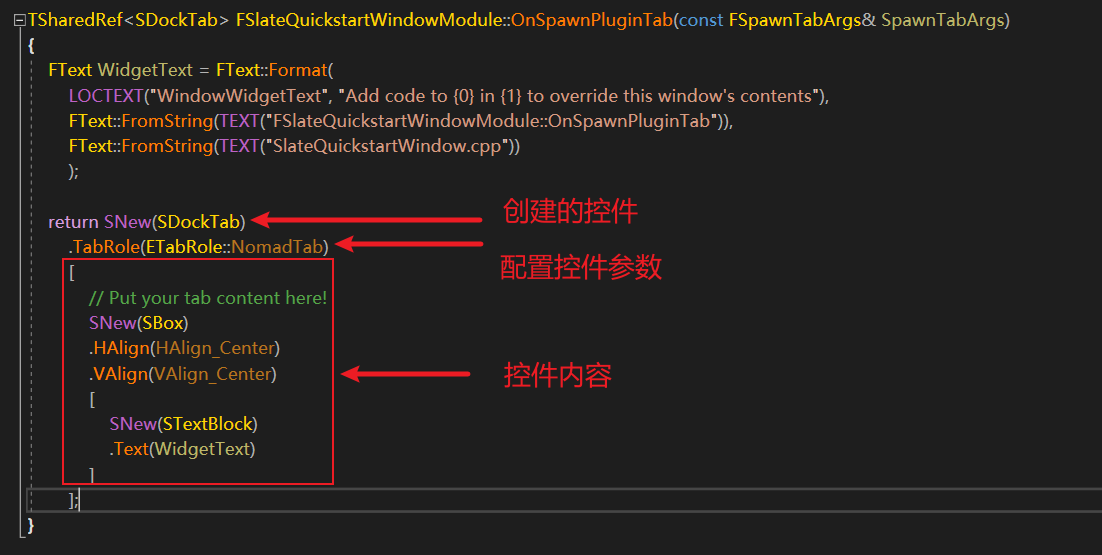
通过一个SNew一个控件类型，然后通过”.XXX”来配置参数或者绑定相关事件，然后再用[]来包含其中内容，例如文字之类的东西

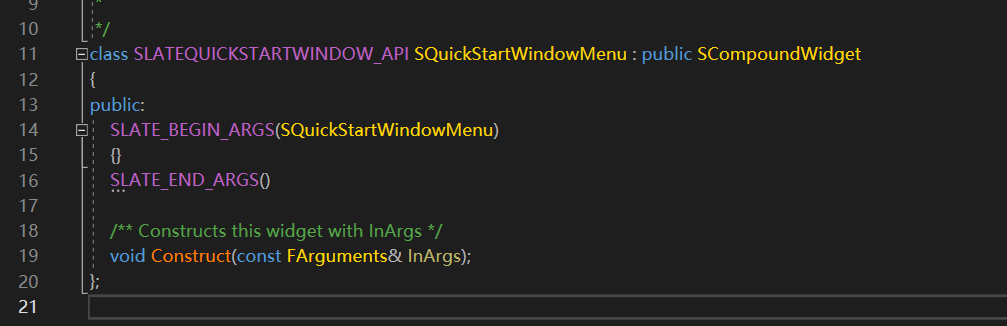




图形用户界面, 应用程序

描述已自动生成





## //SLATE\_BEGIN\_ARGS+SLATE\_END\_ARGS 其实是一个结构体, 内部写的东西都相当于写在了一个结构体里面

## //外部执行SNew或者SAssignNew时候会调用Construct()

## 手机屏幕的截图 描述已自动生成

## ，在Editor Preference-General-Miscellaneous-Display UI Extensions修改，需要重启引擎,可以看到各个ui是隶属于哪个分类的。文本的值是我们应该提供给AddToolBarExtension函数的字符串（AddToolBarExtension后面会说）

## 需要注意的是，将工具栏扩展添加到菜单扩展点将会失败，反之亦然。

## EditorToolBarButton

## 文本 描述已自动生成

## 文本 描述已自动生成

## 函数绑定到事件上

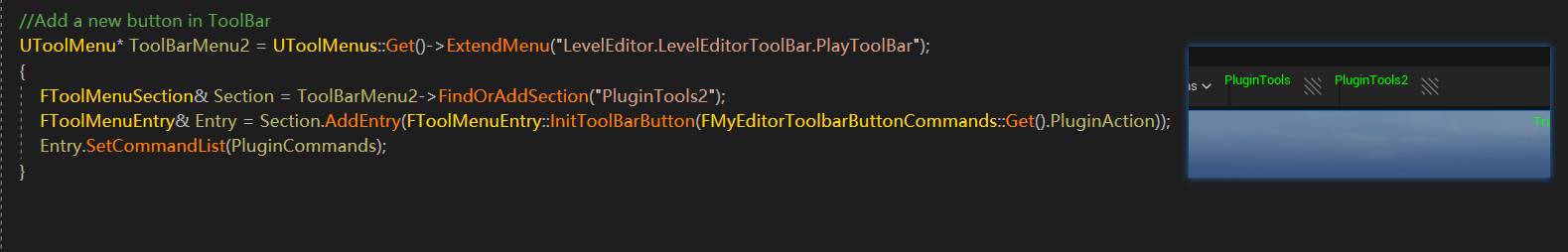
## 图表, 折线图 描述已自动生成

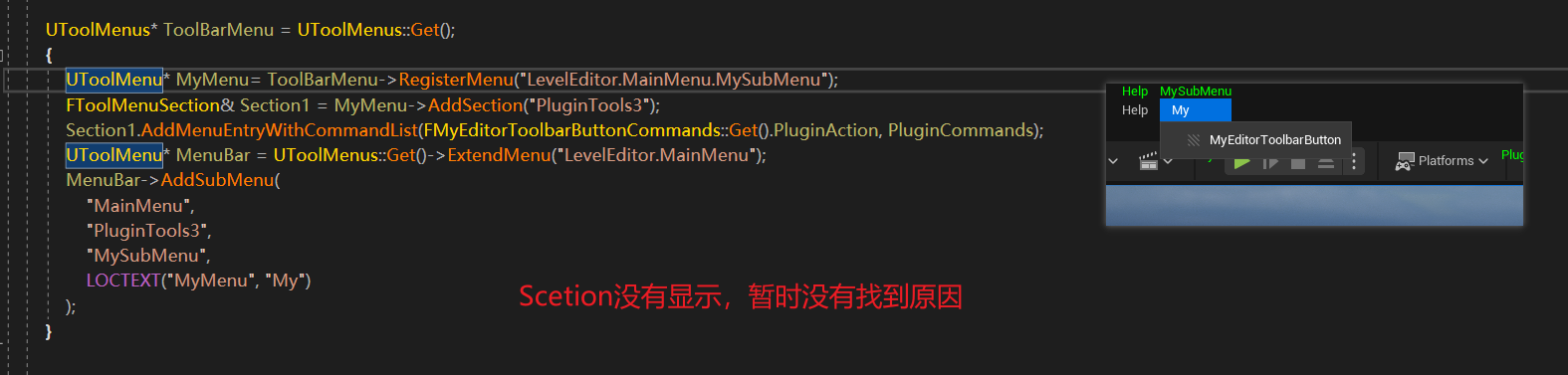
## 文本 描述已自动生成

## 

## 文本 描述已自动生成

## 这段话在扩展菜单 关卡编辑器 关卡编辑器的主菜单 他的窗口





BlankPlugin

[【UE4\_C++】<10-2>C++与UE4的集成-编辑器的拓展部分——创建新的工具栏 - 哔哩哔哩 (bilibili.com)](https://www.bilibili.com/read/cv5529462/)

[TCommands用法 - 知乎 (zhihu.com)](https://zhuanlan.zhihu.com/p/506142825)

Unreal的编辑器的UI是基于命令的概念，允许 UI 和它需要执行的操作之间进行更松散的耦合。

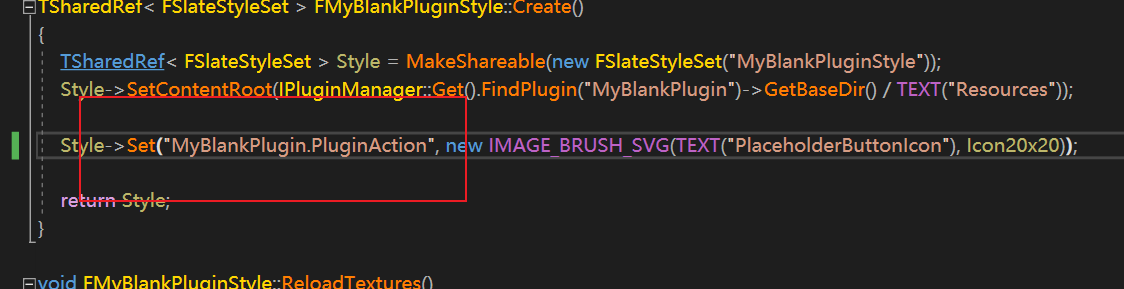
要创建包含一组命令的类，必须从 TCommands 继承。

FSlateStyleSet --SlateStyle样式类，用于设置命令图片

FUICommandList --命令行容器类，用于绑定命令及添加Extender。

TCommands --命令行容器类，用于注册命令

FUICommandInfo --具体的命令，用于设置按钮样式，快捷键，文本以及提示



要与

图示

中度可信度描述已自动生成

相同

文本

描述已自动生成

Tcommands 是一个模板类，它利用了奇异递归模板模式控制协议(CRTP)。

CRTP通常用于 Slate UI 代码中，作为创建编译时多态性的一种方法。

在 FMyBlankPluginCommands 构造函数的初始化器列表中，我们调用父类构造函数，传入一些参数:

第一个参数是命令集的名称，是一个简单的 FName。

第二个参数是工具提示 / 人类可读的字符串，因此，使用 FText，以便在必要时支持本地化。

如果有一个命令的父组，第三个参数包含组的名称。 否则，它包含 NAME\_None。

构造函数的最后一个参数是 Slate Style Set，它包含命令集将要使用的任何命令图标

这是基类的 RegisterCommands()

文本

描述已自动生成

RegisterCommands ()函数允许 TCommands 派生类创建它们需要的任何命令对象。 从该函数返回的 FUICommandInfo 实例作为成员存储在 Commands 类中，以便可以将 UI 元素或函数绑定到命令。

这就是为什么我们有成员变量 TSharedPtr<FUICommandInfo> MyButton。

在类的实现中，我们只需要在 RegisterCommands 中创建命令。

用于创建 FUICommandInfo 实例的 UI\_COMMAND 宏希望定义一个本地化名称空间，即使它只是一个空的默认名称空间。 因此，我们需要将 UI\_COMMAND 调用包含在 #defines 中，以便为 LOCTEXT\_NAMESPACE设置一个有效值，即使我们不打算使用本地化。

实际的 UI\_COMMAND宏需要一些参数:

第一个参数是用于存储 FUICommandInfo 的变量

第二个参数是命令的可读名称

第三个参数是命令的描述

第四个参数是 EUserInterfaceActionType

文本

描述已自动生成

UI\_COMMAND 的最后一个参数是输入和弦，或者是激活命令所需的键的组合。此参数主要是定义链接到相关命令的热键的组合键，而不是用于定义按钮？？

现在，我们有了一组命令，但是我们需要添加到编辑器工具栏中，我们就需要在startmodule/shutdownmodule中放入一些东西



Mapaction 的第一个参数是 FUICommandInfo 对象，我们可以使用它的静态 Get ()方法从 FMyBlankPluginCommands 检索实例。

FMyBlankPluginCommands 是作为单例实现的——一个具有单个实例的类，存在于整个应用程序中。 我们将在大多数地方看到这个模式——引擎中有一个静态 Get ()方法可用。

MapAction 函数的第二个参数是绑定到要在执行命令时调用的函数的委托。

因为 FMyBlankPluginModule是一个原始的C++类而不是 UObject，而且我们希望调用一个成员函数而不是一个静态函数，所以我们使用 CreateRaw 创建一个绑定到原始C++成员函数的新委托。

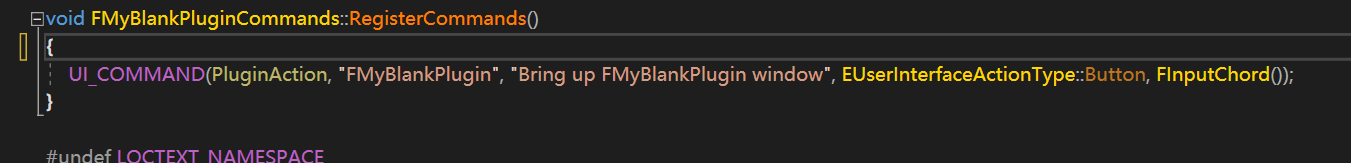
CreateRaw 需要一个指向对象实例的指针，以及一个函数引用来调用该指针。

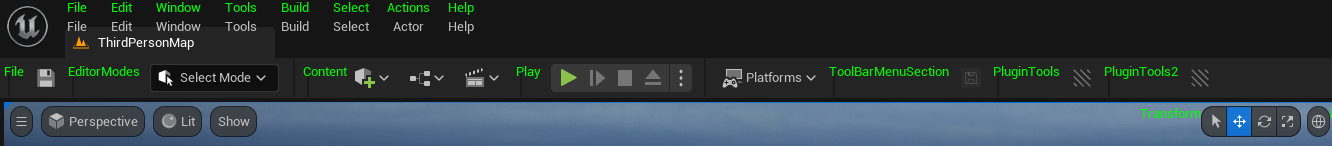
MapAction 的第三个参数是一个调用的委托，用于测试该操作是否可以执行。 因为我们希望该命令始终可执行，所以可以使用一个总是返回 true 的简单预定义委托。

现在我们在命令与调用的操作中创建了关联，现在需要告诉扩展器我们想要添加按钮了

图形用户界面, 文本, 应用程序, 聊天或短信

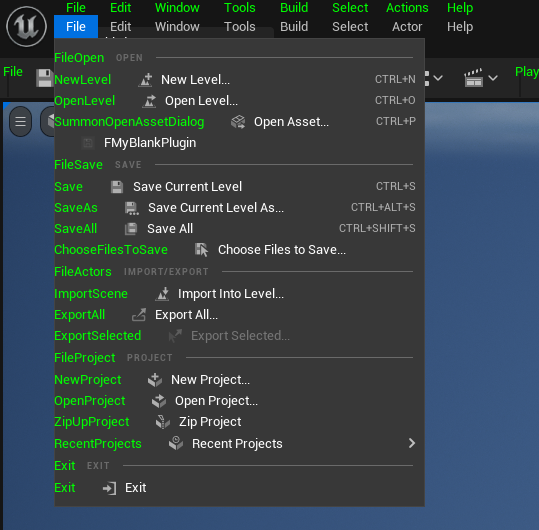
描述已自动生成

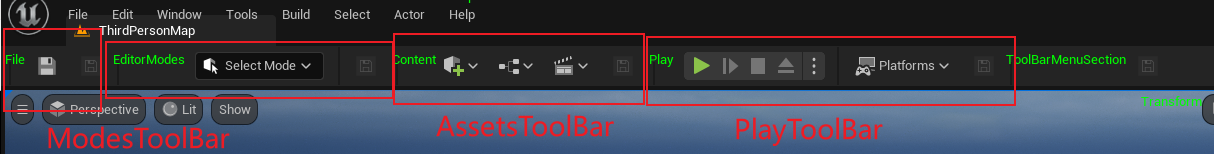




文本

描述已自动生成





具体细节请查阅代码



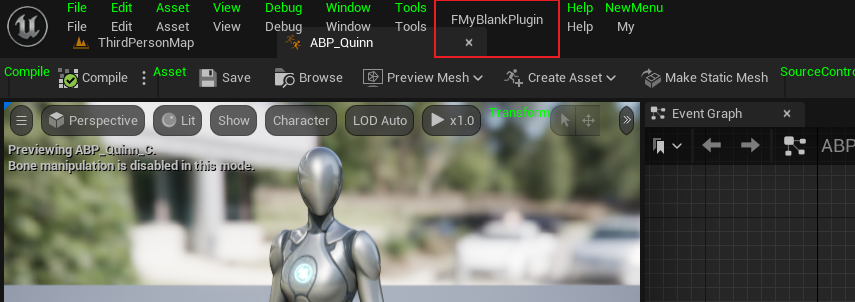
文本

描述已自动生成

蓝图动画菜单：AnimationBlueprintEditor

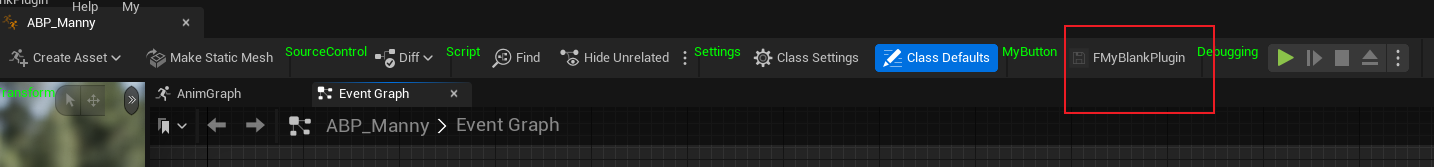
蓝图：Kismet

在动画蓝图的菜单栏里面加一个按钮



文本, 应用程序

描述已自动生成



图形用户界面, 文本

描述已自动生成

图形用户界面, 文本, 应用程序

描述已自动生成

文本

描述已自动生成

如何知道想添加的菜单是否具有扩展能力

例如：IAnimationBlueprintEditorModule

去他的父类看

