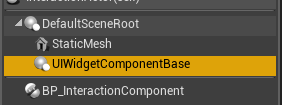
## 使用方法：

制作的UI请继承UserWidgetBase，全屏UI请继承FullScreenWidgetBase，

## 挂载在物体身上的UI：

挂载至物件的UMG，在蓝图中可以直接加载组件UIWidgetComponentBase

，与Widget一样在右侧选择WidgetClass即可，



勾选该属性则此UI会一直朝向玩家

若在C++ 动态添加，

示例：

UUIWidgetComponentBase \*temp\_base = UUIWidgetComponentBase::CreateUIWidgetBase(tempMesh->GetRootComponent(), true,true);

//创建Component并挂载在物品身上

UTestTouchUI temp\_touchUI = temp\_base->SetUI<UTestTouchUI>("WidgetBlueprint'/Game/StoryTest/UI/TouchUITest.TouchUITest\_C'");

SetUI有三种重载，

UUserWidget\* SetUI(UClass \*\_userWidget);

//通过使用蓝图路径，创建一个UI

UUserWidget\* SetUI(FString \_bluePrintPath);

//通过使用蓝图路径，创建一个UI,模板函数

template< typename T1 >

T1\* SetUI(FString \_bluePrintPath)

任意选择一种即可,

注，如果是挂载在STATCIMESHACTOR，则不会调用更新函数

## 全屏不清除UMG详细说明

全屏UMG放在

GameInstance->GetGameViewportClient()->AddViewportWidgetContent(m\_RootWidget->TakeWidget(), 0);

ViewPort中，统一加在一个FullScreenRoot类型的Root根节点下，根节点为CanvasPanel类型，

所有放入的UI，都可以使用UserwidgetBase基类中的设置位置、锚点、Zorder方法（目前只写了这几种）来进行位置等的设置,具体为

SetPosition

SetZorder

SetAnchors

分别有两种重载

## 全屏UMG

示例，通过获取GameInstance中的UMGManager来创建全屏UMG，不会随着场景切换清除

FString screenPath2 = "WidgetBlueprint'/Game/StoryTest/UI/BP\_FullScreenTest2.BP\_FullScreenTest2\_C'";

UFullScreenWidgetBase \*NewWidget = GameInstance->GetUMGManager()->CreateInstanceWidget(GetWorld(), screenPath2);

testUID = NewWidget->GetUID();

UID为创建后返回的该控件ID，可以通过该ID，

UFullScreenWidgetBase\* GetInsUMGByUID(FString UID);

以上方法在蓝图中也有可调用版本

调用函数查找

删除与清空，调用：

//删除全屏UMG

void DeleteInsUMGWidget(FString UID);

//删除全屏UMG

void DeleteInsUMGWidget( UFullScreenWidgetBase\* widget);

//清空全屏UMG

void ClearInsUMG();

CreateScreenWidget则为创建在ViewPort中的UI，会随着场景清除，还需要重构，目前写法不是很完善,用法与上面类似，方法有两种

UFullScreenWidgetBase\* CreateScreenWidget(const UObject\* WorldContextObject, FString \_widgetBlueprintPath, TSubclassOf<UFullScreenWidgetBase> \_widgetType, FString \_widgetName, int32 \_zorder = 0);

//向屏幕添加一个UMG

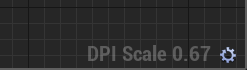
template<typename T>

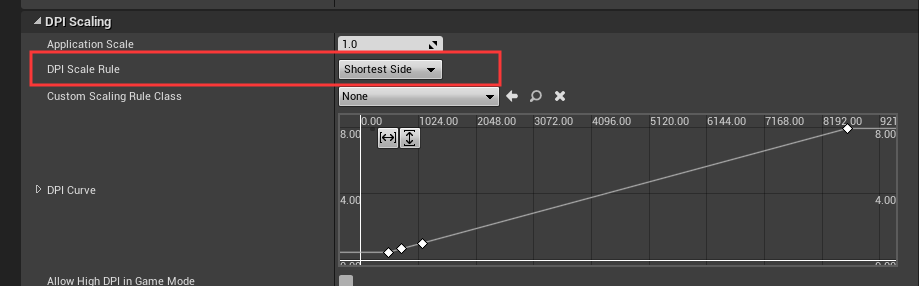
T\* CreateScreenWidget(FString \_widgetBlueprintPath,UWorld\* \_world, TSubclassOf<UFullScreenWidgetBase> \_widgetType, FString \_widgetName, int32 \_zorder = 0)

同样可用于蓝图

DPI解析：

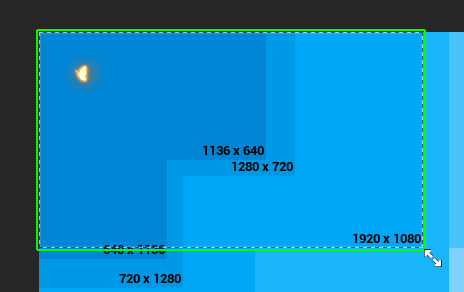
UMG自动适应分辨率，但是要适当设置，

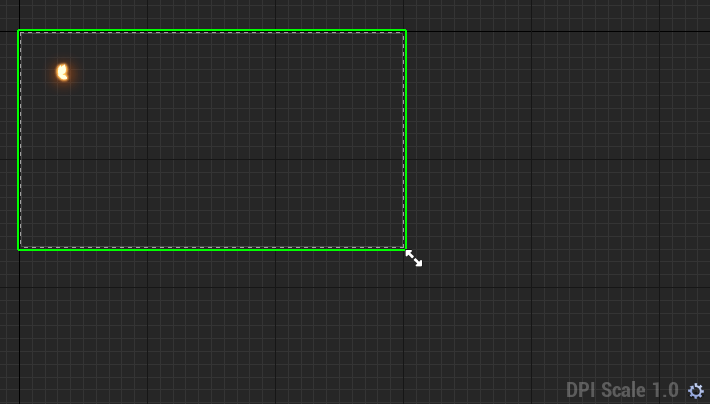




一般使用最短边来进行适配；

首先选择设计分辨率来作为适配基数，例如游戏设计分辨率诶1920X1080，则我们设置该分辨率为DPI的缩放比例1





之后再将最小尺寸比例与最大进行适当设置即可，具体详情可以参考

<https://docs.unrealengine.com/zh-CN/Engine/UMG/HowTo/ScalingUI/index.html>

大概思路及说明：

FullScreenWidgetBase继承自UserWidgetBase，设计是想用来对全屏UMG特殊属性做处理。

UserWidgetBase继承自UserWidget，设计为所有UI的基类，目前有几种设置位置的方法，并且在创建时会根据时间和名字生成一个UID。