宏定义路径下面的指定文件类型

#define IMAGE\_BRUSH( RelativePath, ... ) FSlateImageBrush( FPaths::ProjectContentDir() / "Slate"/ RelativePath + TEXT(".png"), \_\_VA\_ARGS\_\_ )

#define BOX\_BRUSH( RelativePath, ... ) FSlateBoxBrush( FPaths::ProjectContentDir() / "Slate"/ RelativePath + TEXT(".png"), \_\_VA\_ARGS\_\_ )

#define BORDER\_BRUSH( RelativePath, ... ) FSlateBorderBrush( FPaths::ProjectContentDir() / "Slate"/ RelativePath + TEXT(".png"), \_\_VA\_ARGS\_\_ )

#define TTF\_FONT( RelativePath, ... ) FSlateFontInfo( FPaths::ProjectContentDir() / "Slate"/ RelativePath + TEXT(".ttf"), \_\_VA\_ARGS\_\_ )

#define OTF\_FONT( RelativePath, ... ) FSlateFontInfo( FPaths::ProjectContentDir() / "Slate"/ RelativePath + TEXT(".otf"), \_\_VA\_ARGS\_\_ )

使用宏

IMAGE\_BRUSH("Images/SoundCue\_SpeakerIcon", FVector2D(32, 32)) //后面参数表示 屏幕坐标

BOX\_BRUSH("Images/ReplayTimeline", FMargin(3.0f / 8.0f)) //后面的参数表示边缘大小

TTF\_FONT("Fonts/Roboto-Black", 14) //后面的参数表示字号

自定义一个输出宏 在.h 文件中

DECLARE\_LOG\_CATEGORY\_EXTERN(LogOnline, Display, All);

在.cpp文件中， DEFINE\_LOG\_CATEGORY(LogOnline);

// player 0 gets to own the UI

ULocalPlayer\* const Player = GetFirstGamePlayer();

定义枚举，读取枚举的显示中文名字，栗子。。。。。

UENUM(BlueprintType)

enum class NumberMatch : *uint8*

{

one = 1 UMETA(*DisplayName* = "一"),

two UMETA(*DisplayName* = "二"),

three UMETA(*DisplayName* = "三"),

four UMETA(*DisplayName* = "四"),

five UMETA(*DisplayName* = "五"),

six UMETA(*DisplayName* = "六"),

seven UMETA(*DisplayName* = "七"),

eight UMETA(*DisplayName* = "八"),

nine UMETA(*DisplayName* = "九"),

ten UMETA(*DisplayName* = "十")

};

Call function and return ftext。

FText UPlanesTaskInfoUIWidget::GetEnumAsText(NumberMatch enumValue)

{

const UEnum\* enumPtr = FindObject<UEnum>(ANY\_PACKAGE, *TEXT*("NumberMatch"), true); // 解包 找到枚举对象的obj

if (!enumPtr)

{

return FText::*FromString*(*TEXT*("error"));

}

else

{

auto index = enumPtr->GetIndexByValue((*uint8*)enumValue);

return enumPtr->GetEnumText(index);

}

}

静态加载图片

//加载图片 只能在构造函数里面用 不然找不到

AddImage = Cast<UTexture2D>(StaticLoadObject(UTexture2D::StaticClass(), *NULL*, *TEXT*("Texture2D'/Game/UI/Textures/PanelTask/RW\_zhankai.RW\_zhankai'")));

SubsttactImage = Cast<UTexture2D>(StaticLoadObject(UTexture2D::StaticClass(), *NULL*, *TEXT*("Texture2D'/Game/UI/Textures/PanelTask/RW\_shouqi.RW\_shouqi'")));

1. UTexture2D\* Tex = LoadObject<UTexture2D>(NULL, TEXT("Texture2D'/Game/Textures/UI/tex\_test001.tex\_test001'"));

可以用LoadObject加载的文件包括：

Texture、Material、SoundWave、SoundCue、ParticlesSystem、AnimMontage、BlendSpace(1D，2D，3D)、AnimSequence、AnimBlueprint、SkeletalMesh等等。这些文件的父类都是**UObject**，所以也可以先加载为UObject\*然后再强转为具体的类型

1. UObject\* Obj = LoadObject<UObject>(NULL, TEXT("SkeletalMesh'/Game/MyMesh.MyMesh'"));
2. USkeletalMesh\* MyMesh = Cast<USkeletalMesh\*>(Obj);

LoadObject<T>()用来加载非蓝图资源，比如动画、贴图、音效等资源；

LoadClass<T>()用来加载蓝图并获取蓝图Class，比如角色蓝图。如果要用蓝图创建对象，必须先通过LoadClass获取class，然后再通过SpawnActor生成对象。

**LoadClass参数注意事项：**

另外注意：**LoadClass**<T>的模版名称，**不能直接写UBlueprint**，例如：LoadClass<UBlueprint>是错误的，**创建蓝图时选择的是什么父类，则写对应的父类名**，假如是Actor，那么要写成：**LoadClass<AActor>**，否则无法加载成功。

路径名也必须带\_C后缀（LoadObject不需要带\_C后缀），例如，蓝图路径是：**Blueprint'/Game/Blueprints/MyBP.MyBP'**，

加后缀以后，则是：**Blueprint'/Game/Blueprints/MyBP.MyBP\_C**'，

1. UClass\* Test = LoadClass<AActor>(NULL, TEXT("Blueprint'/Game/Blueprints/MapPathBrush\_BP.MapPathBrush\_BP\_C'"));
2. // Load an object.
3. **template**< **class** T >
4. **inline** T\* LoadObject( UObject\* Outer, **const** **TCHAR**\* Name, **const** **TCHAR**\* Filename=nullptr, uint32 LoadFlags=LOAD\_None, UPackageMap\* Sandbox=nullptr )
5. {
6. **return** (T\*)StaticLoadObject( T::StaticClass(), Outer, Name, Filename, LoadFlags, Sandbox );
7. }
8. // Load a class object.
9. **template**< **class** T >
10. **inline** UClass\* LoadClass( UObject\* Outer, **const** **TCHAR**\* Name, **const** **TCHAR**\* Filename=nullptr, uint32 LoadFlags=LOAD\_None, UPackageMap\* Sandbox=nullptr )
11. {
12. **return** StaticLoadClass( T::StaticClass(), Outer, Name, Filename, LoadFlags, Sandbox );
13. }

static ConstructorHelpers::FObjectFinder<UTexture2D> HitTextureOb(TEXT("/Game/UI/HUD/HitIndicator"));

UTexture2D HitNotifyTexture = HitTextureOb.Object; //这个加载和上面是一样的

**一个小坑 中文乱码问题**

**在引擎里面写就不会乱码，在VS 里面写 数据给到引擎就乱码，首先看VS 的中文设置问题，设置编码为 UTF8 编码 windows（LR）标准规范 。我滴天 这个大坑 1一天的时间才发现。 给自己提个醒**

**C++ 定义结构体提供给蓝图调用 ，命名规则是名字前面加F，举个栗子 ：：：**

**USTRUCT(BlueprintType)**

**Struct FGameData //UE4 自己的命名规则 加上F才行 。**

**{**

**。。。。。**

**}**

**//判断是否有中文**

std::string temp = TCHAR\_TO\_UTF8(\*user);

for (int i = 0 ; i < temp.*length*();i++)

{

*WCHAR* h = temp.*at*(i);

if ((h<0XA0B0 && h>122) || h>0xfef7 || (h>90 && h<97) || (h>57 && h<65) || h<48)

{

UE\_LOG(LogClass, Log, TEXT("有中文~！！！！！"));

}

}

在C++中要将一个函数声明为 RPC，您只需将 Server、Client 或 NetMulticast 关键字添加到 UFUNCTION 声明。

　　例如，若要将某个函数声明为一个要在服务器上调用、但需要在客户端上执行的 RPC，您可以这样做：

1 UFUNCTION(Client , Reliable);

2 void ClientRPCFunction();

　　如果要将某个函数声明为一个要在客户端上调用、但需要在服务器上执行的 RPC，您可以采取类似的方法，但需要使用 Server 关键字：

1 UFUNCTION(Reliable, Server, WithValidation)

2 void ServerFlyUp();

　　此外，还有一种叫做多播（Multicast）的特殊类型的 RPC 函数。多播 RPC 可以从服务器调用，然后在服务器和当前连接的所有客户端上执行。 要声明一个多播函数，您只需使用 NetMulticast 关键字：

1 UFUNCTION( NetMulticast,unreliable );

2 void MulticastRPCFunction();

　　接下来定义我们的RPC函数。此时需要注意的是，RPC函数的定义需要在函数末尾添加\_Impementation，这是跟引擎的具体调用有关，这里不深入探讨，有兴趣的朋友可以参考.generate.h文件。我们直接看例子：

1 void AShooterCharacter::ServerFlyUp\_Implementation()

2 {

3 FlyUp();

4 }

1 void AShooterCharacter::FlyUp()

2 {

3 if (this->Role < ROLE\_Authority)

4 {

5 ServerFlyUp();

6 }

7 //implement character fly up

8 //.......

9 }