代码规范

为什么

- 代码需要长期维护，自己或别人
- 一致的风格降低团队其他成员阅读成本
    - 可维护性，可读性
- 降低复杂度
    - 搜索便利
    - 注释
- 对个人而言，一以贯之的风格，书写高效
    - 减少内耗
    - 内心认同，书写愉悦

命名

```
int price_count_reader;// 无缩写
int num_errors;// "num" 是一个常见的写法
int num_dns_connections;// 人人都知道 "DNS" 是什么
int n; // 毫无意义
int nerr; // 含糊不清
int n_comp_conns; // 含糊不清
int wgc connections; // 只有贵团队知道是什么意思
int pc _reader; // pc太多可能的解释
int cstmr_id; // 缩减若干字母
```

格式

- 缩进，4空格
- 换行
- 用空行将代码片段分开

代码书写

- 直白易懂
- 关键逻辑添加注释
- #if #endif 宏结束时注释标识在哪个宏（Rider自动标出）

UE5 C++代码规范

命名

- 清晰、明确、避免过度缩写
- 变量名:大驼峰式(CamelCase)，bool类型须加b前缀
- 类型名：前缀+大驼峰式

- 引用传入可能修改的函数变量：加Out前缀

开头字母	表示含义	示例
T	模板类	TMap
U	继承自U0bject	UMoviePlayerSettings
A	继承自AActor	APlayerCameraManager
S	继承自Swidget	SCompoundwidget
I	抽象接口类	INavNodeInterface
E	枚举	EAccountType
b	布尔变量	bHasFadedIn
F	其他类	FVector

数据类型

基础类型

数据类型	意义	大小(字节)
bool	布尔值	sizeof(bool)
TCHAR	字符	sizeof(TCHAR)
int8/uint8	有符号/无符号字节	1
int16/uint16	有符号/无符号短整数	2
int32/uint32	有符号/无符号整数	4
int64/uint64	有符号/无符号整数	8
float	单精度浮点数	4
double	双精度浮点数	8
PTRINT	与指针同样大小的整数	sizeof(PTRINT)

```
/** If true, when the actor is spawned it will be sent to the client but re... */
UPROPERTY()
uint8 bNetTemporary:1;
/** If true, this actor was loaded directly from the map, and for netwonking...
*/
UPROPERTY()
uint8 bNetstartup:1;
/** If true, this actor is only elevant to its owner. If this flag is chang... */
UPROPERTY(Category=Replication, EditDefaultsOnly, BlueprintReadonly)
uint8 bOnlyRelevantToOwner:1;
/** Always relevant for network(overrides bonlyRelevantToOwner). */
UPROPERTY(Category=Replication, EditDefaultsonly, BlueprintReadwrite)
uint8 bAlwaysRelevant:1;
```

- 基础类型：uint64/int16等
- 字符串类：使用UE定义的FString/FText/FName/TCHAR等
- 容器类：应避免使用stl，使用UE定义的TArray/TMap等

代码风格

- 大括号换行
- if-else对齐
- 使用Tab缩进
- switch-case语句中，必须要有default，如果fall-through，必须有明确注释

命名空间

- UnrealHeaderTool仅支持全局命名空间的类型
- 不要在全局命名空间使用using 声明
- 一些宏在命名空间内可能会失效，可以尝试UE_前缀的版本

C++11

- 不要使用NULL宏，使用nulptr来表示空指针
- Lambda函数须明确指出返回类型
- 不要使用auto，除了:Lambda函数、迭代器声明，模板中类型推导

```
auto AddFunc = [](int A, int B) -> int
{
    return A + B;
}
```

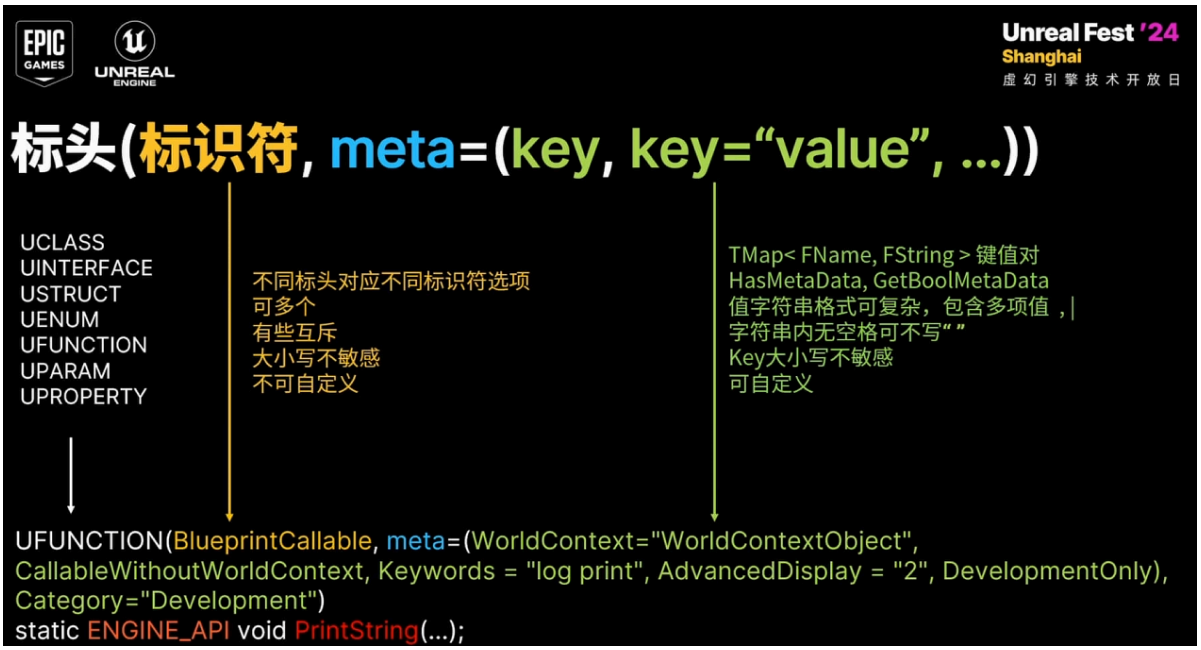
Epic C++ Coding Standard for Unreal Engine | Unreal Engine 5.5 Documentation | Epic Developer Community | Epic Developer Community

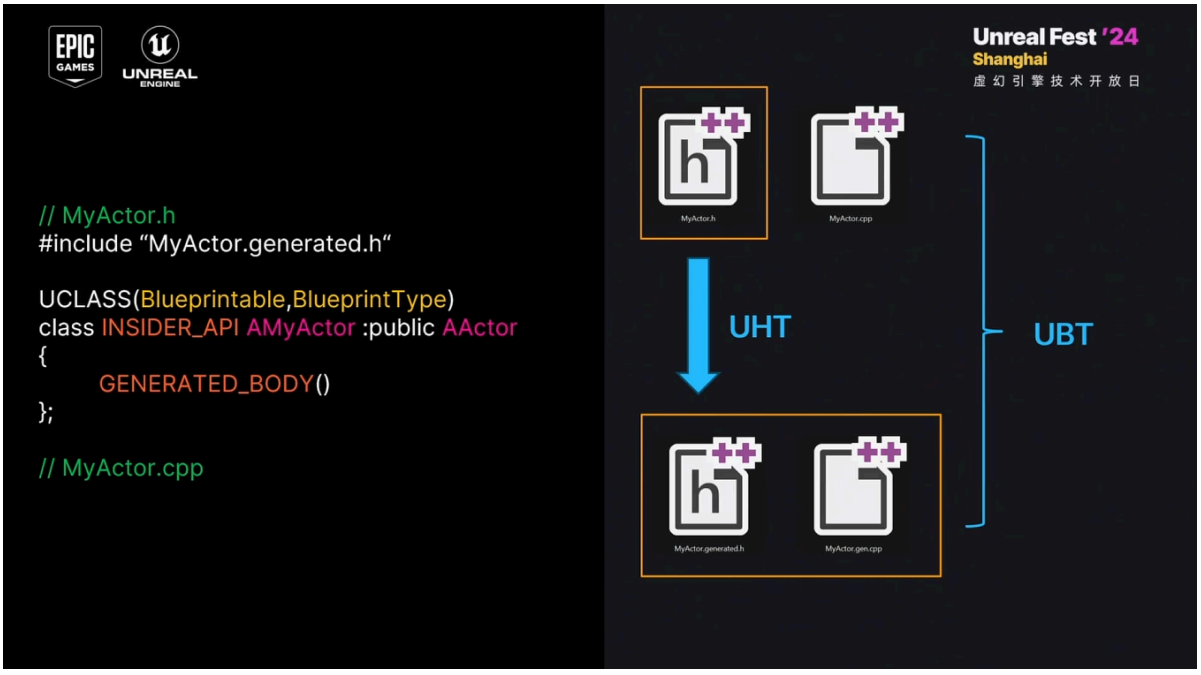
如何写符合规范的代码

- 熟悉代码规范，理解规范设立的原因
- 使用辅助工具，标注出不符合规范的代码
    - Cpplint, Resharper C++等

UE5中的标识符

fjz13/UnrealSpecifiers: UE5标识符详解，包含100多个标识符以及300多个meta的解释和示例。

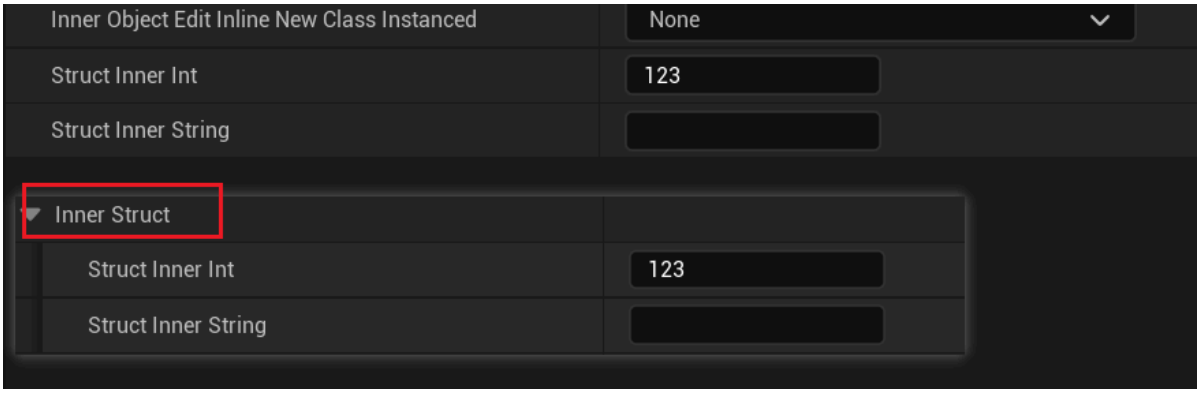




UPROPERTY

```
USTRUCT(BlutprintType)
struct Fcat
{
    GENERATED BOOY()
    UPROPERTY(EditDefaultsonly)
    FString Name;
    UPROPERTY(EditDefaultsonly)
    int32 Age;
    UPROPERTY(EditDefaultsOnly)
    FLinearColor Color;
};
UPROPERTY(EditAnywhere, category="Cat without SHowOnlyInnerProperties")
FCat Cat;
/* 把结构属性的内部属性直接上提一个层级直接展示 */
UPROPERTY(EditAnywhere, category="Cat without SHowOnlyInnerProperties", meta=
(ShowonlyInnerProperties))
```

```
FCat Cat;
```



Visible Anywhere

```
/* 在默认值和实例细节面板均可见，但不可编辑 */
UPROPERTY(VisibleAnywhere)
int32 VisibleAnywhereNumer;
```

Edit Anywhere

```
/* 在默认值和实例的细节面板上均可编辑 */
UPROPERTY(EditAnywhere)
int32 EditAnywhereNumer;
```

Category

在蓝图的右键菜单中为该函数指定类别分组，可以嵌套多级

```
UPROPERTY(EditAnywhere,Category="Animals")
bool bIsCute;
UPROPERTY(EditAnywhere, Category="Animals|Dogs" )
FString BarkWord;
UPROPERTY(EditAnywhere, Category="Animals|Birds” )
int32 FlyingSpeed = 99;
```

Display Name

此节点在蓝图中的命名将被此处提供的值所取代，而非代码生成的命名。

```
UPROPERTY(EditAnywhere,meta=(DisplayName="Display Font"))
FSoftObjectPath DisplayFontPath;
```

Advanced Display

把函数的一些参数折叠起来不显示，需要手动点开下拉箭头来展开编辑。

```
UPROPERTY(EditAnywhere, Category="Toy")
FString Name;
UPROPERTY(EditAnywhere, Category="Toy" )
int32 HappyPhraseCount;
UPROPERTY(EditAnywhere, Category="Toy",AdvancedDisplay)
bool bEnableEvilMode;
```

Clamp Min / Max

指定数字输入框实际接受的最小/大值

```
UPROPERTY(EditAnywhere, meta=(ClampMin=0,ClampMax=100))
int32 RestoreHealthPercent;
```

Blueprint Read Only / Blueprint Read Write

蓝图变量只读，或蓝图变量可读可写

UFUNCTION

- BlueprintCallable

蓝图可调用函数

```
UFUNCTION(Blueprintcallable, category=Actor)
virtual void SetOwner( AActor* NewOwner );
/** Get the owner of this Actor, used primarily for network replication. */
UFUNCTION(Blueprintcallable,category=Actor)
AActor* Getowner() const;
```

- Meta 函数蓝图参数默认值

```
public:
    // (CPP_Default_InLocation = 1.0, 2.0, 3.0, ModuleRelativePath =
Function/Param/MyFunction_TestParam.h)
    UFUNCTION(Blueprintca11ab)
    FString MyFuneTestParam_DefaultVector(FVector InLocation = FVector(1, 2, 3));
    // (Location = "4,5,6", ModuleRelativePath =
Function/Param/MyFunction_TestParam.h)
    UFUNCTION(Blueprintcallable, meta = (Location = "4,5,6"))
    FString MyfuncTestParam_DefaultInMeta(const uobject* WorldcontextObject,
FVector Location, float Size);
```

UCLASS

引擎生成，XXX_API为导出宏，使用后对应函数/类/结构体等可在其他模块使用

USTRUCT

UENUM

Ref

https://benui.ca/unreal/uproperty/
https://github.com/fjz13/Unrealspecifiers
https://dev.epicgames.com/documentation/en-us/unreal-engine/metadata-specifiers-in-unreal-engine