BLUEPRINTS TO C++

UNREAL ENGINE 4 - C++ PROGRAMMING GUIDE

EPISODE 9

UENUM BASICS

OUTLINE

1. Standard UEnum
2. Standard UEnum Namespace Technique
3. Enum Class UEnum
4. Bitmask UEnum
5. Tip of the Day

CREATING A STANDARD UENUM

```
#include "BarrierType.generated.h"

UENUM(BlueprintType, Category="GameRules")

enum EBarrierTypeStd

{

    EBT_None            UMETA(DisplayName = "No Barrier"),

    EBT_Moderate        UMETA(DisplayName = „Moderate Barrier"),

    EBT_Difficult       UMETA(DisplayName = „Difficult Barrier"),

    EBT_VeryDifficult   UMETA(DisplayName = „Very Difficult Barrier"),

    EBT_Impassable      UMETA(DisplayName = „Impassable Barrier"),

};
```

Note: Value names of enum must be unique overall, so best use Prefixes for values

STANDARD UENUM USAGE

Properties:

```
UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "Config|Grid")
TEnumAsByte<EBarrierTypeStd> BarrierType = EBT_Moderate;
```

Functions:

```
bool AConeActor::GetBarrierType(TEnumAsByte<EBarrierTypeStd>& OutBarrierType)
{
    …
    OutBarrierType = EBT_Moderate;
    return true;
}
```

Note: Standard UEnums cannot be declared with their type when used as a UProperty. So it must be defined as a TEnumAsByte type instead to work as a UProperty

STANDARD ENUM NAMESPACE TECHNIQUE

Example from UE4 Core Enum:

```
UENUM(BlueprintType)
namespace ESplinePointType
{
    enum Type
    {
        Linear,
        Curve,
        Constant,
        CurveClamped,
        CurveCustomTangent
    };
}
```

Note: Standard UEnums with the namespace technique cannot be declared with their type when used as a UProperty. So it must be defined as a TEnumAsByte type instead to work as a UProperty

STANDARD ENUM NAMESPACE USAGE

Properties:

```
UPROPERTY(EditAnywhere, BlueprintReadWrite)
TEnumAsByte<ESplinePointType::Type> SplinePointType = ESplinePointType::Type::Curve;
```

Functions:

```
bool AConeActor::GetSplineType(TEnumAsByte<ESplinePointType::Type>& OutSplineType)
{
    …
    OutSplineType = ESplinePointType::Type::Curve;
    return true;
}
```

Note: This technique is used massively in the UE4 Game Framework. It‘s an older way to write enums, try use the class enum way of creating UEnums which is shown next

CREATING AN ENUM CLASS UENUM

```
UENUM(BlueprintType, Category="GameRules")
enum class EBarrierType : uint8
{
    None            UMETA(DisplayName = "No Barrier"),
    Moderate      UMETA(DisplayName = „Moderate Barrier"),
    Difficult         UMETA(DisplayName = „Difficult Barrier"),
    VeryDifficult  UMETA(DisplayName = „Very Difficult Barrier"),
    Impassable     UMETA(DisplayName = „Impassable Barrier"),
};
```

Note: Afte the enum keyword, it is followed by the class keyword. Also the enum must be of type uint8 to work.

ENUM CLASS UENUM USAGE

Properties:

```
UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "Config|Grid")
EBarrierType BarrierType = EBarrierType::Moderate;
```

Functions:

```
bool AConeActor::GetBarrierType(EBarrierType& OutBarrierType)
{
    …
    OutBarrierType = EBarrierType::Moderate;
    return true;
}
```

Note: With a class UEnum you can declare a UProperty with the enums Type name and so you don‘t need to use TEnumAsByte like you have seen in the other standard enum types

BITMASK UENUM

```
UENUM(BlueprintType, Meta = (Bitflags))
enum class EUnitKeyword : uint8
{
        None      = 0  UMETA(Hidden),
        Activated = 1  UMETA(DisplayName = "Activated"),
        Stopped   = 2  UMETA(DisplayName = "Stopped"),
        Prone     = 4  UMETA(DisplayName = "Prone"),
        Running   = 8  UMETA(DisplayName = "Running"),
};
…
```

BITMASK ENUM USAGE

```
UPROPERTY(BlueprintReadWrite, meta = (Bitmask, BitmaskEnum = "EUnitKeyword"))
uint8 UnitKeywords = EUnitKeyword::Activated;
…
bool Unit::HasKeyword(EUnitKeyword Keyword) const
{
    return (UnitKeywords & static_cast<uint8>(Keyword));
}
void Unit::AddKeyword(EUnitKeyword Keyword)
{
    UnitKeywords  |= static_cast<uint8>(Keyword);
}
void Unit::RemoveKeyword(EUnitKeyword Keyword)
{
    UnitKeywords &= ~static_cast<uint8>(Keyword);
}
```

TIP OF THE DAY – VALUE / DISPLAY VALUE AS STRING

Get Text/String from UEnum

```
EBarrierType BarrierType = EBarrierType::Moderate;

…

FString ValueString;

UEnum::GetValueAsString(BarrierType, ValueString)

UE_LOG(LogTemp,Warning,TEXT("BarrierType Value is %s"), *ValueString);

…

FText DisplayName;

UEnum::GetDisplayValueAsText(BarrierType, DisplayName);

UE_LOG(LogTemp,Warning,TEXT("BarrierType Display Value is %s"), *DisplayName.ToString());
```

More Functions can be found here:

https://docs.unrealengine.com/en-US/API/Runtime/CoreUObject/UObject/UEnum/index.html

THANK YOU FOR WATCHING

IF YOU WANT TO GET NOTIFIED WHEN NEW VIDEOS ARE COMING OUT

THEN PLEASE SUBSCRIBE TO THE CHANNEL