BLUEPRINTS TO C++

UNREAL ENGINE 4 - C++ PROGRAMMING GUIDE

EPISODE 3

UFUNCTIONS

OUTLINE

UFUNCTION

```
UFUNCTION([specifier1=setting1, specifier2, ...], [meta=(key1="value1", key2, ...)])

ReturnType FunctionName([Parameter1, ..., ParameterN1=DefaultValueN1, ParameterN2=DefaultValueN2]) [const];
```

Example:

```
UFUNCTION(BlueprintCallable, Category=“Startup”, meta=(DisplayName=“Initialize Cone Actor"))

void Initialize(int32 Width);
```

IMPORTANT UFUNCTION SPECIFIERS

1. BlueprintCallable
2. BlueprintPure
3. BlueprintImplementableEvent
4. BlueprintNativeEvent
5. Category

BLUEPRINT CALLABLE

1. The function is implemented in C++ and can be executed in a Blueprint or Level Blueprint graph.
2. When executed in blueprints the function appears as a blue node with an in and out execution pin

BLUEPRINT PURE

1. The function is a const function that is implemented in C++ and that can be called from Blueprints.
2. It does not affect or change any data and it returns one or more values. The function appears in a graph as a green node without an execution pin.
3. A BlueprintCallable function that has one or more return values and is declared as const, also appears as a blueprint pure node .

BLUEPRINT IMPLEMENTABLE EVENT

1. The function is not implemented in C++, but can be called from C++. The implementation is done in Blueprints by overriding the Function.
2. If the Function has a return value it appears as a function in blueprints when overridden
3. If the function has no return value, it appears as a custom event in the event graph when overridden

BLUEPRINT NATIVE EVENT

1. This function can be overridden by a Blueprint, but it also has a default C++ implementation.
2. Besides declaring the native event function, an additional function named the same as that function, but with _Implementation added to the end needs to be declared and implemented.
3. If the Blueprint does not override the function, the C++ implementation function is called.
4. If the blueprint overrides this function , it can choose to call the parent c++ function in its implementation

IMPORTANT META TAGS

1. DisplayName=“Property Name” (Blueprint Displayed Name)
2. Tooltip=“Long Tooltip”
3. ShortToolTip="Short tooltip“
4. HideSelfPin

FUNCTION PARAMETERS

1. Define Input parameters that are not changed as const
2. Output parameters are passed as references (Type& Param)
3. For pointers to classes to appear as output params in blueprint also add a reference to it (ClassType*& Param)
4. Input Parameters that are structs and not changed are best passed as const reference (const Type& Param)
5. If you have several Output parameters, and one is a bool, then use the bool as the function return value and the others as output parameters

FUNCTIONS AND GENERAL TOPICS

- Camel Case Naming Convention (function names must begin upper case)
- Shadowed Variables are not allowed
  - For Function Parameter Name use In/Out Naming Convention

Documentation Link to Functions

https://docs.unrealengine.com/en-US/Programming/UnrealArchitecture/Reference/Functions/index.html

TIP OF THE DAY

UPARAM(ref)

Non const reference parameters are usually recognized as output parameters in Blueprint. So they appear as output pins on the right side of the node. If you want to pass a non const Reference as Input parameter and that Blueprint recognizes the parameter as an input pin, then use UPARAM(ref) as a declaration before that parameter.

Example:

```
UFUNCTION(BlueprintCallable, Category=“Config”))
bool InitializeData(int32 MinHeight, UPARAM(ref) FVector& Location)
```

THANK YOU FOR WATCHING

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

THEN PLEASE SUBSCRIBE TO THE CHANNEL