Blueprints and Code

•••

ElectronicArmory.com
3D Game Development Course

Script Types

Construction Script - Ran when an actor is created. Works like a constructor

Event Graph - Run based on actor's events (begin, collision, movement, ticks)

Tick, Tick, Tick

Begin Play - Runs right before the actor is set to receive tick events

Tick - A notification of a frame

Delta Seconds - Difference in time between ticks/frames

C++ Macros

```
UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category = Camera, meta =
(AllowPrivateAccess = "true"))
```

UPROPERTY

VisibleAnywhere

EditAnywhere

BlueprintReadOnly

BlueprintReadWrite

BlueprintAssignable

UProperty Macro Values

VisibleAnywhere

VisibleDefaultsOnly

VisibleInstanceOnly

UFUNCTION

BlueprintPure - Useful for BPs that don’t need to modify C++ (getters or calculations)

BlueprintCallable

BlueprintNativeEvent - Event in BPs that have a C++ implementation. Can be overridden in BP

Category

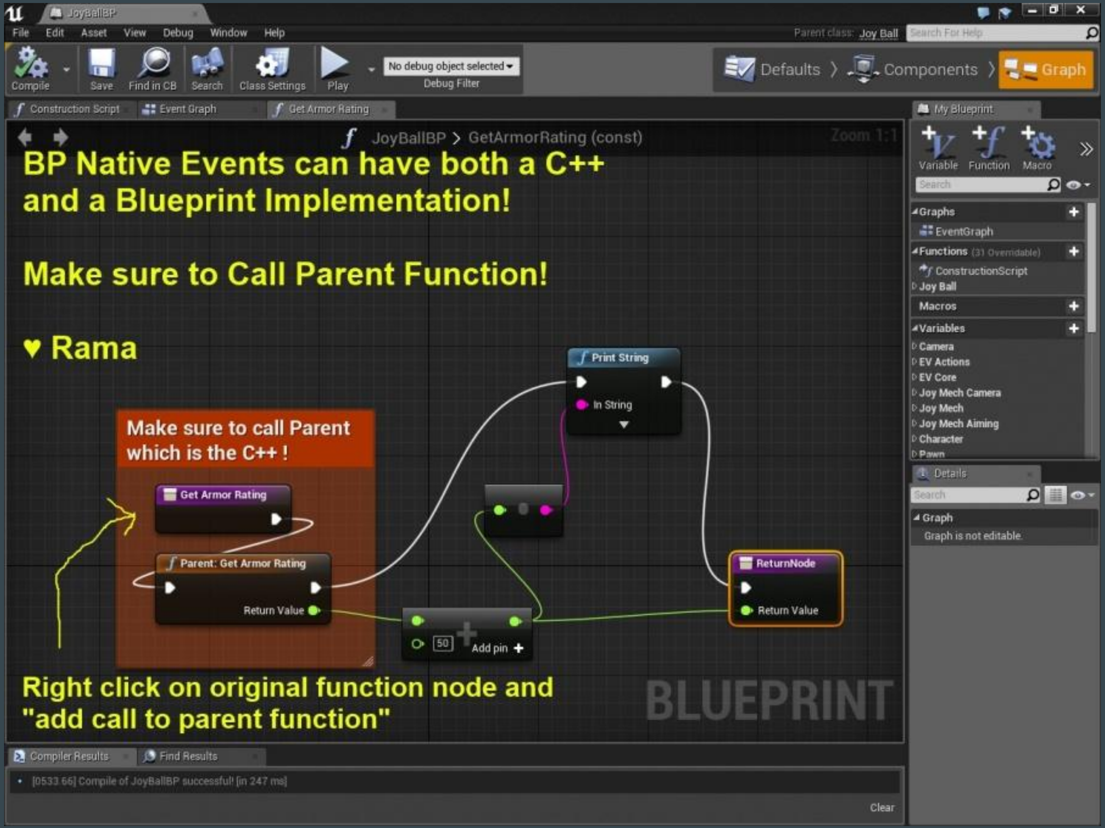
BlueprintNativeEvent

```
UFUNCTION(BlueprintNativeEvent, BlueprintCallable, Category="JoyBall")
float GetArmorRating() const;



float AJoyBall::GetArmorRating_Implementation() const
{
    //remember to call super / parent function in BP!
    V_LOG("C++ Happens First");
    return 100;
}
```

Call Parent Implementation (Super)


BP Native Events can have both a C++ and a Blueprint Implementation!

Make sure to Call Parent Function!

♥ Rama

Make sure to call Parent which is the C++ !

Right click on original function node and "add call to parent function"

[0533.66] Compile of JoyBallBP successful! [in 247 ms]

C++ Character Implementation (.h or .hpp)

```
Public:
    AOGWCharacter();

protected:
    virtual void BeginPlay();

public:
    /** Base turn rate, in deg/sec. Other scaling may affect final turn rate. */
    UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category=Camera)
    float BaseTurnRate;
```

C++ Character Implementation (.cpp)

```
void AOGWCharacter::MoveRight(float Value)
{
    if (Value != 0.0f)
    {
        if (UGameplayStatics::GetPlayerController(GetWorld(),
0)->IsInputKeyDown(EKeys::LeftShift))
        {
            Value *= 4;
        }
        // add movement in that direction
        AddMovementInput(GetActorRightVector(), Value/2);
    }
}
```

Player Controllers, Pawns and Characters

```
APlayerController *PlayerController =
UGameplayStatics::GetPlayerController(GetWorld(), 0);

APawn *PlayerPawn = GetWorld()->GetFirstPlayerController()->GetPawn();

AOGWCharacter *Character = Cast<AOGWCharacter>(PlayerPawn);

if( Character ){
    Character->MoveRight(30.0f);
}
```

```
UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category = Camera, meta =
(AllowPrivateAccess = "true"))
class USphereComponent* CollectionSphereComponent;
```

Creating A Pickup Object

- Create the pickup
  - Static Mesh Component (RootComponent, instantiate)
  - Is Active (Boolean, Getter/Setter)
  - Expose to Blueprint
- Extend Character to collect pickups
  - Collection sphere (USphereComponent, Instantiate)
    - SetupAttachment(RootComponent)
    - SetSphereRadius(200.0f)
  - Collection Input
    - InputComponent->BindAction(TEXT("Collect"), IE_Pressed, this, &[ClassName]::CollectPickups);
    - Setup “Collect” in Project Settings