

# APooledCharacter

APooledCharacter is mainly responsible for handling the pooling logic for the characters that use it and properly handle instances that aren't pooled.

Inherits from class: ACharacter

Implements interfaces: IPoolable, IEventListener

- Important Variables to Know
- Initialise (yes, we know it's misspelled here)
- SetUsed

# Important Variables to Know

This isn't *all* of the variables that are present but are some that are very important to the APooledCharacter's functionality:

int64 SoulID: A unique identifier for an NPC. This number is used to find the correct data associated with that NPC, so if you give the APooledCharacter a SoulID number, you will always get that "soul" (if it exists). Think of it like the equivalent of an American Social Security Number.

int32 DefinitionID: The order that the DefinitionData is in its corresponding database. It refers to general information about an NPC type, but not to a specific instance of that type (that would be SoulID). Think about this as the "species" that the APooledCharacter will become.

bool blsUsed: Is this instance of APooledCharacter currently in use? Unused instances can be given the data needed to become an NPC while not overriding an instance that's already in use.

bool bUsingPooledMethod: Determines whether or not this instance of APooledCharacter is actually part of a pool. Instances that are in a pool are re-used so that we don't have to constantly create and destroy APooledCharacter instances, which can bog down performance if left unchecked.

# Initialise (yes, we know it's misspelled here)

At this level the initialize function is a lightweight function; all it does is sets the value of the APooledCharacter's soul and DefinitionDataID (as seen in Initialise\_Implementation). The child blueprints need to use the NPCType variable to find the correct definition database that the DefinitionDataID references. When overriding this function in blueprint, ensure that it has a call to its parent function so that the variables are properly set.

```
UFUNCTION(BlueprintNativeEvent, BlueprintCallable, Category = "Pool")
void Initialise(int64 characterSoul = -1, int32 DefinitionDataID = -1, EVE_ObjectType NPCType = EVE_ObjectType::VisitingNPC);
virtual void Initialise_Implementation(int64 id, int32 defID, EVE_ObjectType NPCType) { SoulID = id; DefinitionID = defID; };
```

# SetUsed

SetUsed tells the APooledCharacter instance whether or not it is being used. If the instance is part of a pool, it enables/disables the ActorTick, collision, and visibility, then either begins or ends using the APooledCharacter (more on that later). If the APooledCharacter is NOT part of a pool, then it either begins using the instance or destroys it.

```
void APooledCharacter::SetUsed(bool bUsed)
{
    if (bUsingPooledMethod) {
        bIsUsed = bUsed;

        SetActorTickEnabled(bUsed);

        SetActorEnableCollision(bUsed);

        RootComponent->SetVisibility(bUsed, true);

        GetWorld()->GetTimerManager().ClearTimer(Timer);
        Timer.Invalidate();

        if (bUsed) {
            GetWorld()->GetTimerManager().SetTimer(Timer, this, &APooledCharacter::CallSetUsedFromTimer, Lifetime, false);

            BeginUsed();
        }
        else {
            EndUsed();
        }
    }
    else {
        if (bUsed) {
            BeginUsed();
        }
        else {
            UE_LOG(LogTemp, Log, TEXT("Destroying PooledCharacter"));
            Destroy();
        }
    }
}
```