Gameplay Abilities

Gameplay Abilities are actions or skills that an actor can perform

* Class derived from UGameplayAbility
* Defines what an ability does and the conditions under which it can be used
* Not implemented as a simple function; GAS game ability is an instanced object running asynchronously
* Can be activated at some point in time and run multi-stage tasks that may or may not span across a span of time
* These tasks could branch off into different tasks, apply their own gameplay effects, kick off **gameplay cues**
* Can be conditional, based on player or environmental interaction
* Gameplay abilities have built-in replication and prediction support
* Built in concept of cost and cooldown

A screenshot of a video game

Description automatically generated

## Ability Tasks

Gameplay abilities use ability tasks (although they don’t HAVE to)

* Ability tasks derived from UABILITYTASK
* Specialised form of UABILITYTASK
* Perform asynchronous work during a gameplay ability’s execution
* Can affect execution flow by broadcasting delegates
* Can be used in C++ or in BP through a specialised BP node
* BP node can have multiple output execution pins
* These are execute as a result of delegate broadcasts made in response to occurrences determined by developer – thing happens in game, task broadcasts delegate
* Can design the ability in the BP but make the actual work happen in code, very flexible

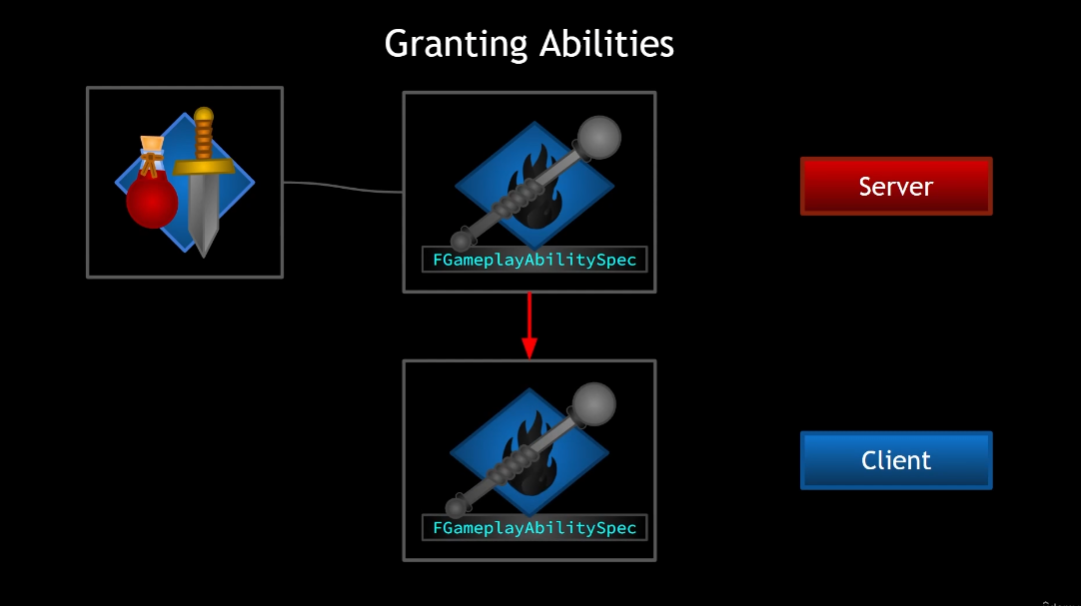
A screenshot of a computer

Description automatically generated

## Granting Abilities

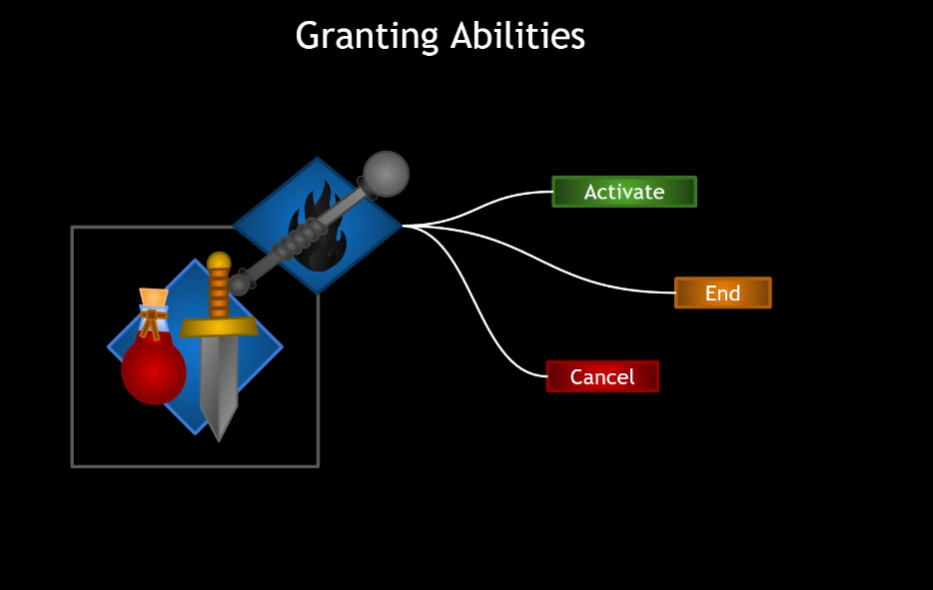
To use a gameplay ability the ASC must be *granted* the ability

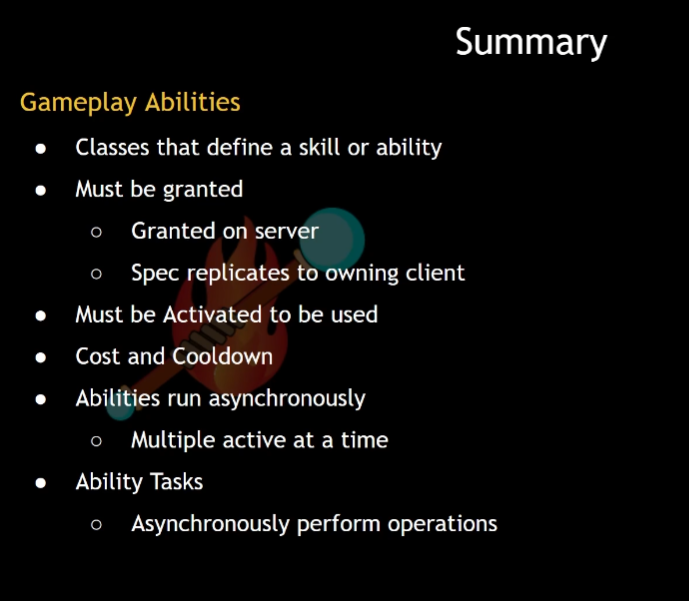
* GameplayAbilitySpec is created
* Spec defined details pertaining to that ability including
* GameplayAbility class itself
* Ability level
* Dynamic info that can be changed at run time
* Abilities granted on the server
* Ability spec then replicates to the owning client



Gameplay Abilities have the concept of *Activation*

* Once activated, gameplay abilities are *Active*
* Until Ended or Cancelled
* Abilities can end themselves or be cancelled exernally





# Granting Abilities

Possible: Make a blueprint based on the gameplay ability class

Instead: create a base class of our own! That way all GPA we create will have the option to have some baseline functionality

A screen shot of a computer

Description automatically generated

A black screen with orange and white text

Description automatically generated

Very basic template to start!

So, we want to grant abilities and this is done on the server

Let’s start by just giving the character a list of abilities that should be given at the start of the game

In AuraCharacterBase:

Add a TArray of GameplayAbility classes(type TSubclassOf)

Then we can set that TArray from within the character blueprint

A screen shot of a computer

Description automatically generated

So there will be an array of abilities, Now we need a way to \*give\* these abilities

This is done through the AbilitySystemComponent, so let’s add a function for adding those abilities

We may want to override in child classes so Protected

A screen shot of a computer

Description automatically generated

This function should only add abilities on the server, so we’ll do an auth check first

A screen shot of a computer

Description automatically generated

If we do have authority we want to:

Grant the ability; done by ASC, so let’s have the ASC actually do it; the CharacterBase will just call it

It should be public

A screen shot of a computer

Description automatically generated

The function will need to take in Gameplay Ability classes – this is the function we’ll call from CharacterBase

Since it’s a list of things we’ll loop through them

A screen shot of a computer

Description automatically generated

To add the abilities we need to create an ability spec from each of the ability classes



This is a struct that requires an ability class, so we’ll pass in AbilityClass

It also requires an ability level

A screen shot of a computer

Description automatically generated

Levels are just int32s. We’ll store the struct locally



And now we have the Spec, we can just use a function that exists on the ASC already called GiveAbility()



We could also give the ability and activate it immediately!

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

AbiltySpec \*cannot\* be const if we’re going to GiveAbilityAndActivateOnce!

A screen shot of a computer code

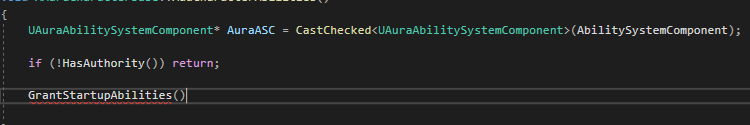
Description automatically generated

GiveAbility will accept either

For the moment we’re gong to use ActivateOnce, because as a learning exercise I want to see what happens when an ability is activated and this is the easiest option that I know so far

Last thing to do here is call the function in AuraCharacterBase

To do this we’ll need to cast to the ASC



Nb this requires an include for the AuraASC.h

A screen shot of a computer

Description automatically generated

So, where to call AddCharacterAbilities?

For the player character, a good place is in PossessedBy (worry about enemy later)

A screen shot of a computer program

Description automatically generated

A screen shot of a computer screen

Description automatically generated

Now we should see the ability added after InitAbilityActorInfo

We should expect, upon calling this function, to be granted all the abilities in the startup ability array

A screenshot of a computer

Description automatically generated

We can set the array of abilities here and there are some defaults, but I will make a new BP for the Abilities

A screenshot of a computer

Description automatically generated

It has Event Activate and Event OnEnd nodes

A screenshot of a cell phone

Description automatically generated

We can also use C++ functions, but we have events here

Testing:

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

Now, pressing play should grant and activate this ability on press play

A purple and green object on a grey tile floor

Description automatically generated

Done, it’s active. But there’s no End Ability print string – Event OnEndAbility was never called!

Events can end themselves

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A close up of a purple object

Description automatically generated