Gameplay Ability System Components

Gameplay Tags (if)

They are the **conditions**, sorted as a ***hierarchical*** list.

- ***Parent.Child.GrandChild***
  State.Buff.HealthRegen
  State.Debuff.Stun

Given to an Object these Tags are set on the owning ASC. It has an Interface to take care of current active Gameplay Tags and will help identify which Ability or Gameplay Effect will be applied.

To store Gameplay Tags, it's wise to use a ***GameplayTagContainer***. Helping with Replication or counting the active Tags to see if the Object ***HasMatchingTag().***

Gameplay Tag Container

Is made to hold the collection of Gameplay Tags that we defined. Allows for checking if a Tag is present or removing and adding new ones. Can be used to apply Effects or Abilities via a specific Tag.

Overridable with CanActivateAbility()

This allows for activation even if the Tag would block the application.

Gameplay Attributes (what)

They represent **Values** for specific ***Actor data*** and ***states***:
Health, Ammo, AvailablePotionUsages etc.

An Attribute is made of two different Values: ***BaseValue*** & ***CurrentValue***
BaseValue: The permanent Value
CurrentValue: The permanent Value + Modifications

Permanent changes to the BaseValue: Instant and Periodic Gameplay Effects
Changes to the Current Value: Duration or Infinite Gameplay Effects

Attribute Sets

Attributes are ***stored inside AttributeSets*** and it is your choice if you are using one or many different Attribute Sets. Attributes can be marked for Replication inside the AttributeSet and should only be ***changed through Gameplay Effects***.

Attribute Sets can be subclassed, but only use one AttributeSet from the same Class on the same Actor.

To fine tune changes for Attributes there are two Methods available:

PreAttributeChange:

To make changes to the CurrentValue before the change is active, like clamping. But this is no permanent change, if other Calculations need a clamped value, you have to clamp there again.

```
virtual void PreAttributeChange(const FGameplayAttribute&
Attribute, float& NewValue) override;
```

PostAttributeChange:

Reacting to changes of the BaseValue. Could be used for showing damage numbers, adding experience points, calculating damage mitigation from shields or armor.
All Values that are changed by instant Gameplay Effects can be accessed here.

```
virtual void PostAttributeChange(const FGameplayAttribute&
Attribute, float OldValue, float NewValue) override;
```

Gameplay Effects (how)

These effects allow to ***change Gameplay Tags*** and ***Attribute Values*** via Abilities.
With the ***duration policy*** (instant, duration, infinite) we can create different types of Attribute changes. They can be instant like Damage/Healing or will be applied over time like a Mana drain debuff.

In short: we will change the behavior of our character through Attribute manipulation.

Periodic Effects
Use the Period value for Effects that should be ***applied for X amount of Seconds***.
This could be used for ***Damage Over Time*** or ***Health Regeneration***.

There are many different ways to apply a Gameplay Effect, but you always start from the ASC and use any of the available Functions:

ApplyGameplayEffectToSelf: takes the template with our modifying further, simpler
ApplyGameplayEffectSpecToSelf: modify the GameplayEffect before adding, advanced

Gameplay Effect Spec

- Gameplay Effect: How the Effect is build, what it is made of
- Effect Context: Owner, Target and other relevant information
- Effect SpecHandle: Handle which helps to remove, change or find this specific Effect

Applying a Gameplay Effect will create a Gameplay Effect Spec and this is ultimately applied to the target.

Example: You make a projectile holding a GameplayEffectSpec, this will be applied OnHit to the Target Actor and returns a FActiveGameplayEffect.

Gameplay Effect Context

This structure carries information about the Gameplay Effect Instigator, Target Data and is a good way to pass around data from Attribute Sets, EffectExecuteCalculations or Gameplay Cues.

Gameplay Effect Execution Calculation

Used to make detailed changes what the Ability or Effect will ultimately do and could be depending on other factors like Attributes of the Player or the Target.

Gameplay Cues

Each Gameplay Effect has the option to add/execute a ***Gameplay Cue*** for a ***visual representation*** of the current effect.

Instant Effects will call an ***execute*** Gameplay Cue/Gameplay Tags
Infinite, Duration Effects will call ***add and remove*** on the Gameplay Cue/Gameplay Tags

Abilities (why)

Are the main ***brain of the system*** and ***define*** any ***action*** or ***skills*** that are available for the player during the game. There can be more than one Ability active at the same time.

Abilities can be actions like:

- Jumping
- Aiming
- Shooting
- Using Items
- Interacting
- Collecting

But these should not be implemented with Abilities:

- Basic Movement Input
- UI Interaction

TryActivateAbility

Makes an attempt to activate the Ability, needs the InputID and the Ability to activate.

EndAbility

It's important to call this Node, omitting leads to issues in calling the next Ability.

Commit

Abilities are not fired off immediately they are queued. This node will activate the Ability and there is no way back.

Cooldown

Adding these to the properties will check the time, since the last activation and will not fire until the cooldown is reset.

Ability Tags

Allows for grouping Abilities together with the same Tag. Calling this Tag will involve all Abilities which fall into this category.

Ability Spec

Holds data about the Ability that was activated, like its owner, the duration and Gameplay Effects attached to it. Also used for replication to other clients on the Network.

Ability Tasks (when)

These tasks are **asynchronous** and can wait for an event to happen. If you want to play an animation montage, the task is your preferred choice. Also has functionality to listen for Animation Notifies and will tell you when the animation is done.

Also able to remove Gameplay Effects. Often the project will need its own customized tasks tailored to its specifications.