Damage

# Meta Attributes

So far the gameplay effects that modify Health are reaching into the attribute set and changing the value of Health directly. Typically in RPG games with GAS this is not how it is done, because applying damage to an enemy there are a number of calculations that need to be made, and those calculations have to do with attributes on the attacker and victim. For this reason, to keep the mathematics simple, we often use a placeholder attribute that behaves as an intermediary that allows us to perform all the necessary maths before setting the value of the health, in the case of damage

These intermediary values are called **Meta Attributes**

Normal attributes:

* Often Replicated
* Changed on server and replicated to clients

Meta attributes:

* Temporary
* Not replicated
* Only used on server
* Allows for calculations

Damage:

* Meta Attribute would be *Incoming Damage*

How Incoming Damage works:

A GE causes 11 damage to a target. Instead of subtracting 11 health, the GE instead adds 11 to Incoming Damage

There are ways t respond to attribute changes in the attribute set eg PostGameplayEffectExecute is used to check if the attribute that has changed is the Incoming Damage Meta Attribute. If so, we then calculate if there was a Block, a Crit, is there an Intelligence/Strength Bonus. Whatever, we perform those calculations to determine the final real amount of damage actually inflicted and then subtract that number from Health

A screenshot of a computer

Description automatically generated

The Gameplay Effect is not the arbiter of Health subtracted from the attribute, it’s the rules set by those equations that determines the final value

The Incoming Damage is set by the GE, and the Attribute Set performs the calculations it needs to and then zero out the Incoming Damage Meta, Set the Value of Health and then respond in any other ways needed eg floating text (which may be a different colour depending on bock or crit), apply debuff effects, damage over time, stuns, hit react anim

Meta attributes are therefore often the way that GAS projects handle damage

# Damage Meta Attribute

In GE\_Damage:

A black and white striped background

Description automatically generated

This is an Instant Gameplay Effect that adds a negative value directly to Health. This is really a placeholder Gameplay Effect – it hardcodes a Negative 10, doesn’t use a curve table

Instead we’ll modify a meta attribute, then allow the Ability Set to process the data. We’ll also use the damage meta in the custom calculations as a custom calculation class as Damage will be one of the more complicated types of modifiers

We’ll add the Meta to the Attribute Set

A screen shot of a computer program

Description automatically generated

Easy enough. Note that the UPROPERTY includes a category and BP Readonly, like the other attributes, but there is NO ReplicatedUsing, because this is not a replicated attribute

So how do we actually **use** the meta attribute?

In the CPP:

A computer screen with text

Description automatically generated

Here we can see it the attribute data has the attribute IncomingDamage, in exactly the same way we check for GetHealthAttribute and GetManaAttribute

A screen shot of a computer program

Description automatically generated

So, inside the if check we know we are getting a change to the IncomingDamage Meta Attribute

As IncomingDamage is a Meta, it should be used for it’s value and then reset to zero – we will consume that data!

To do this we will make a local float variable that has the value IncomingDamage

A black screen with white text

Description automatically generated

So we’re caching that value locally and then we can set that meta attribute to zero

A computer screen with white text

Description automatically generated

IncomingDamage is now ready to take a new value, because we have received the data into this function

Now we have the Local incoming damage we can decide what to do with it, and we should only do something with it if its value is greater than zero

So far we have nothing to do with the Meta Attribute other than set the health value, so we can do that for now and handle any additional clamping

A screen shot of a computer program

Description automatically generated

At this point, we can tell certain things about the damage that has been done; eg if NewHealth is 0 then we know that the damage done is fatal. So we can do something like this:

A screen shot of a computer

Description automatically generated

If NewHealth is equal to or less than 0 we know that bFatal will be true - enough damage was done to kill the owner of this attribute set

So this is basically how it should work – we use the meta instead of changing health and the gameplay effects that currently affect health will be changed to affect the Meta on the target, and they will be adding to it, not subtracting, because the subtraction is done here in NewHealth – LocalIncomingDamage

In GE\_Damage:

A screenshot of a computer

Description automatically generated

The Attribute is now set to Incoming Damage and we’ll increase the damage a bit for testing

A video game with a firework

Description automatically generated with medium confidence

The bar is going down; we’re doing damage with a meta attribute, which allows us to do all kinds of things; we can check the damage done, perform custom calculations

There’s no hit react or death at zero, but we’ll worry about that later

GE looks better but we’re still hard-coding 25 for the damage – would be better if this were linked to the Gameplay Ability somehow, so if the Gameplay Ability has it’s own damage variable, how do we make sure that the Gameplay Effect is using the Damage from the Gameplay Ability?

That would be the 4th Magnitude type:

A screen shot of a message

Description automatically generated