RPG Character Classes

Enemy health is now being affected by Gameplay Effects (although it’s only a placeholder, since we want damage to be affected by the secondary stats for a real RPG combat experience)

Also, character is just using the default attribute value; in a real RPG different character types start with different attributes

These character types are often separated into different character classes

A screenshot of a video game

Description automatically generated

To properly initialise enemy attributes we should think about how we will do this; Aura will level up and get stronger; we also have enemy class level (a class variable) and for an enemy to spawn in at a different level that should **mean** something – higher level enemy = higher attribute values

So, how to initialise attributes for a character based on it’s character class?

Warrior: mainly melee attacks

Rangers: ranged attacks

Elementalist: magical spells

So characters should have a way to easily set their character class, which determines their starting attributes. Starting Level should also affect starting attributes

Loading in a character with a certain class and level is mostly related to enemy characters; Aura starts with base stats, unless being loaded from a save

Enum perhaps?

An asset to store data…a data asset

Gameplay effects can use curve tables to scale gameplay effect magnitudes based on level

Curve tables are a single asset that can hold multiple curves, like for Primary Attributes for example

A screenshot of a video game

Description automatically generated

So each character class, specified with character class Enum, can have a curve table with curves for each primary attributes storing the starting values that can scale up as level increases

Will also apply a Gameplay effect to initialise the attributes, then populate the secondary attributes based on the primaries, and an effect for Vitals once everything else is set

Finally any Abilities the enemy character class has should be in this data asset so they can granted at the beginning of the game

Some of the enemies may have the same abilities and effects eg Death or hit react as an ability and each enemy share those ability classes, so the data asset should also have all abilities and effects that all enemies should be given

A screen shot of a black and yellow text

Description automatically generated

6: could be in the AuraAbilitySystemLibrary, take in the Asset, ASC, and character level and apply all effects to initialise attributes

# Character Class Info

A screenshot of a computer program

Description automatically generated

Asset created, now need to store data for each character class and Enum to categorise classes

A screen shot of a computer

Description automatically generated

We can add constants for each character class

A screen shot of a computer program

Description automatically generated

Now we can distinguish between character classes

We also need a struct, with all the information for each class

New struct for the character class default info:

A screenshot of a computer program

Description automatically generated

In this struct we’ll need a gameplay effect to apply the primary attributes

A screen shot of a computer program

Description automatically generated

Now primary attributes can be set here

We could have Secondary and Vital gameplay Effects per class, or share the same Secondary and Vital among all classes

Let’s share them outside of the struct, in the main body

A computer screen shot of a program code

Description automatically generated

The asset now needs a way to store the structs, one for each class.

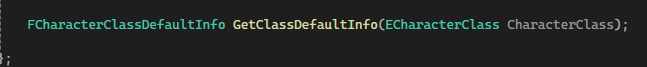
TArray or Map?

Map, so we can map the Enum to the struct

A screen shot of a computer

Description automatically generated

We want to retrieve the information for a given enum constant, so we’ll make a function to look up the info



A screen shot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

FindChecked is good because it will perform an assertion and if no enum we’ll get an assert

New Data Asset BP in Unreal based on the class:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A black and white screen

Description automatically generated with medium confidence

Now we have a data asset we can create some new Gameplay effects to add to the pulldown menus

# Default Attribute Effects

We have already created default attributes for the Aura Character, but we will need them for the enemies as well

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

The Secondary Attribute Effects could just be the same some shared among all classes unless there was a reason to tweak the coefficients in the calculations made in the modifiers

A screenshot of a computer

Description automatically generated

So this could be set up to be different for each class with a separate GE so for example Warriors get more armor from their resilience or something, but we don’t have to do that

For this exercise I’ll use the same Secondary Attributes formulae and rename GE\_AuraSecondaryAttributes to GE\_SecondaryAttributes

A screenshot of a computer

Description automatically generated

Now we have some Gameplay effects they can be set in the Data Asset

A screenshot of a computer

Description automatically generated

# Curve Tables – CSV and JSON

Currently these Primary Attribute GameplayEffects have no modifiers – with modifiers we can have curve tables that allow the modifier magnitude by a modifier based on what the curve table returns using a lookup by level

GE\_Primary Atributes will be a gameplay effect with instant duration policy. There will be 4 modifiers, one for each attribute and for each we’ll use a curve table. For the curve tables we’ll have different value based on level, so that when we spawn in an enemy we can apply the gameplay effect at the level specified.

Curve tables can be created in a number of different ways, and the data can be stored in a number of different ways.

Creating a new curve table you have interpolation options:

A screen shot of a computer

Description automatically generated

Linear: Between each XY value pair(each key in the curve) the curve is a straight line

Cubic: the curve is a smooth curve

Constant: no interpolation at all

For Elementalist going with Cubic, and renaming the curves to match the Gameplay tags of the attributes

A screenshot of a computer

Description automatically generated

Cool thing about interpolation types is we don’t have to have a value for each level; can enter one for every 5 or 10 levels, and if we try to retrieve a value at a give point we get the interpolated value on the curve

A screenshot of a computer

Description automatically generated

A screen shot of a graph

Description automatically generated

NB in the curve table the values are assumed to be seconds and are marked with an S, but they don’t have to be used that way

A black screen with white text

Description automatically generatedAdd a second value:

A graph with a line going up

Description automatically generated

Select the points, R-click and auto interp:

A screenshot of a graph

Description automatically generatedA graph with a line going up

Description automatically generated

This table has a cubic interpolation type, so we can get a mathematical graph of a cubic function

A screenshot of a video game

Description automatically generatedNew Key:

A graph with a line going up

Description automatically generated

Normalized view mode will normalise to all keys involved:

A graph with a line going up

Description automatically generated

As we can see the curve smoothly interpolates from one value to the next, so if we query the key at level 2

A screenshot of a computer

Description automatically generated

This can save some time if we want to shape out the curve by shape and just interpolate between major values, then we don’t have to put in a value for each level

New Key:

A graph on a black background

Description automatically generatedA graph with a line going up

Description automatically generated

We can also click and move the existing values to make the curve look more as we like, if we wish, 10.5 at 10 is a more pleasing curve:

A graph with a line

Description automatically generated

So far this is looking a bit linear so maybe cubic is overkill, but that’s OK, we’re only querying once at spawn anyway

New key:

A graph on a screen

Description automatically generated

Str isn’t THAT important for an elementalist so 14 at 20 is fine

We can add one last for fun at 40!

A graph on a black background

Description automatically generatedA graph with a line going up

Description automatically generated

Now we have a curve for Strength for the Elementalist

A blue rectangle with white text

Description automatically generatedNew curve:

A screenshot of a computer

Description automatically generated

In GE\_PrimaryAttributesElementalist:

A screenshot of a computer

Description automatically generated

Remember to Override!

Repeat for Warrior and Ranger.

Additional possible formats for the curve table data:

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedThe first row determines X Axis value (levels)

At level 1 Str 5, int 15 etc

Some are empty because I did not fill them in

A white grid with black numbers

Description automatically generatedUseful because we can look at the values and edit them here

In Notepad it looks like this:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generatedA screenshot of a computer program

Description automatically generated

New values are present and no longer interpolated

A graph on a black background

Description automatically generated

If select all and r-click can no longer Auto, forced to be linear – limitation on importing data

CSV:

So, we can import and export – can also create curve tables direct from CSV by making a CSV file (or copying the Elementalist and saving with a new name and values as I did) and import as before

JSON:

Exported Ranger and saved as a JSON:

A screenshot of a computer

Description automatically generated

Defaulted to open in VS but I reopened in Notepad

A screenshot of a computer

Description automatically generated

Import File CT\_PrimaryAttributes\_Warrior.json:

A screenshot of a video game

Description automatically generated

JSON allows for cubic curve import:

A graph with a line going up

Description automatically generated

Now we have the curve tables and have associated them all with the GE Primary Attributes gameplay effects, we can use the Data asset to apply those values

Nb: currently the Secondary attributes affect, which is shared with Aura, is infinite so that Aura’s secondaries can update in response to changes in the Primaries. But we don’t plan on having the enemies update primary stats at runtime – they get attributes based on level and that’s it

So we can duplicate the Secondary, set it to an instant Gameplay effect and add it

A screenshot of a video game

Description automatically generated

We could use Infinite for enemy secondaries if they can level up, but they won’t in this project

# Initialising Enemy Attributes

We have the Character Class Info data asset, but where should it be stored?

One on every character seems redundant – not every character needs the data asset

Store in one central location – Game Mode, because this is where the rules of the game are stored! And this Data asset basically contains a lot of rules, like what attributes enemy have based on level

A screen shot of a computer program

Description automatically generated

Fwd declared and added as a variable in the AuraGameModeBase header. How do we access it and apply the properties?

The BlueprintLibrary is a good place for this function

This will be a static function, and it will initialise default attributes based on a character class and level



ECharacterClass is not defined here, it’s in CharacterClassInfo so we can just include that



The function will be callable in BPs and has a category



What the function needs to do:

It needs to get the Character class info data asset, which exists on the Game Mode

A screenshot of a computer

Description automatically generated

Requires a World Context Object, we can just copy-paste from the other functions in this class which have one

A screenshot of a computer

Description automatically generated

A screen shot of a computer program

Description automatically generated

The function returns a GameModeBase, so we will cast it to AuraGameModeBase

A screen shot of a computer

Description automatically generated

And only if this succeeds should this continue



Once we have the Game Mode we can access the data asset Character Class Info



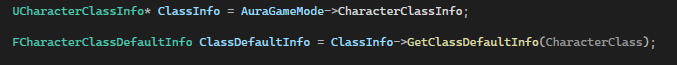
CharacterClassInfo has a lookup function : A black screen with white text

Description automatically generated

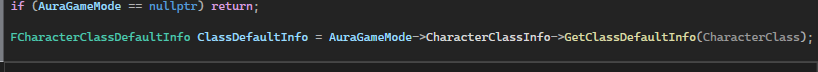
Which returns the struct A computer screen shot of text

Description automatically generated

Now in the Library we can build a struct for class default info Calling the ClassInfo asset’s GetClassDefaultInfo function and pass in Character Class



There’s no real need for the local variable so we’ll refactor



Next we need to apply some gameplay effects that exist in this struct, and we need to apply them to an AbilitySystemComponent

So the function also needs to pass in an ASC



And fwd declare

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

This function needs a GameplayEffectSpec

A screenshot of a computer

Description automatically generated

MakeOutgoingSpec requires a TSubclassOf GE, GE class, level and Context Handle

First thing is the ContextHandle, since it’s easy



Gameplay Effect: The ClassDefaultInfo struct already has this internally: 

So we can access it here: 

And then just put it right into the function call

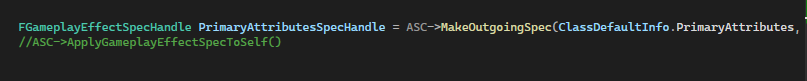
A screen shot of a computer screen

Description automatically generated

Along with Level and the Context



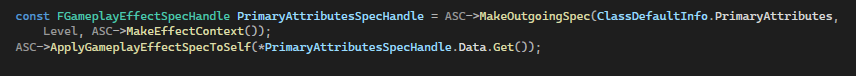
This returns an FGameplayEffectSpecHandle



Which we can pass into Apply to self, although it requires a Spec, not a handle so:



Nb the handle can be a const



So now we have a function that takes a context object, a character class Enum constant, a level and an Ability System Component. It’ will get the Data Asset from the Gameplay Mode, call the Class Default Info and from that get the Primary Attributes. After we apply the Gameplay Effect for Primary Attributes we can apply the Secondary and Vital as well

The process is the same, but Secondary and Vital are not in the struct, they’re direct on the Asset itself

So going to reinstate the local variable and use it:

A screen shot of a computer code

Description automatically generated

Plus one for Bital

A screen shot of a computer program

Description automatically generated

All 3 GEs are now applied

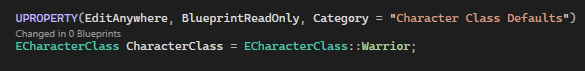
This all assumes that the Game Mode have the Data Asset on it; if it doesn’t we’re attempting to access a nullptr and will get a crash

If we get a crash on 50 it’s because Character class info is null

Once that is set on the Game Mode we can call this function to initialise the attributes of the enemies, provided they have a valid Level and Character Class

We have not yet given the enemy character classes!

In AuraEnemy.h (#include "AbilitySystem/Data/CharacterClassInfo.h")



Now we have a character class we can initialise the default attributes

A screen shot of a computer program

Description automatically generated

We’ll do it here, because we know that the ASC has has it’s avatar actor set

In fact, we can see we are calling InitialiseDefaultAttributes from the Character Base class, which we can override if we make it a virtual and then use the new function we added to the library

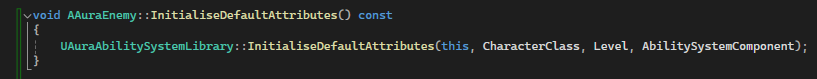
A black screen with white text

Description automatically generated 

A screen shot of a computer

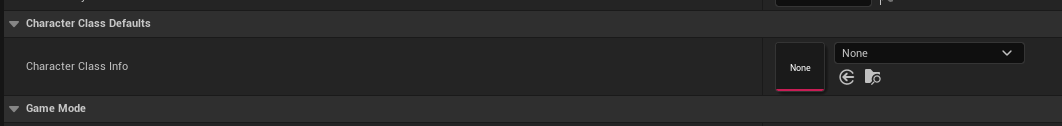
Description automatically generated A screenshot of a computer error

Description automatically generated



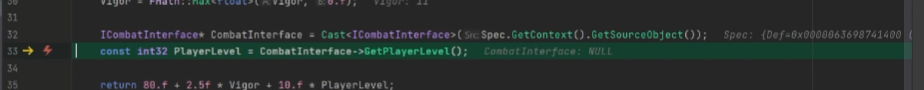
So now we should see the enemies spawn with he correct stats for their class and level!

In the Game Mode BP:

 A screenshot of a computer

Description automatically generated

EXCEPTION!



This is because we didn’t set a source object!

That’s OK, we’ll do it in the new function and we can do it by accessing the ASC, which has the Avatar

A black background with white text

Description automatically generated

This can be the source for the gameplay effects.

To SET the source, we do it on the effect context so:

A screen shot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

And then the same for Secondary and Vital