Ability Tasks

We now have a FireBolt gameplay ability which spawns a firebolt when you are hovering over an enemy and LMB

Ability Tasks are like the workers a GA employs to do tasks, either instantaneously or they might span some period of time.

There are a number of tasks that already exist in the gameplay system, and we can also create our own

For example:

A screenshot of a computer

Description automatically generated

This is what is called a Latent Node, latent as in asynchronous or ‘will be executed and then, depending on what happens later, there are a number of output pins that can be executed’

A screenshot of a video game

Description automatically generated

There’s an input pin to the left, and regular output execution pin to the top right

The top right output is executed immediate after we reach the node, allowing other functions and logic to be called or performed.

But then there are also a number of circumstances that can trigger the other outputs to flow like ‘On Completed’ when the montage is completed, or ‘On Blend out’ when the montage starts to blend out and is not interrupted. Depending on blend settings this could be before On Completed (usually is). On Interrupted is self explanatory, on Notify Begin is executed when using a play montage notify, or a play montage notify window inside the montage, as we can put notifies in the montage. On Notify end for when a notify window ends – there are a number of things that can cause any of these to trigger.

This is just a Blueprint latent node (aka an Async Task). GAS has more specialised forms of these latent nodes which are more ingrained in the ability system itself – for example Play Montage And Wait under Ability-Tasks

A screenshot of a computer

Description automatically generated

This is an ability task and if we click this it looks a lot like the other one but has more options

A screenshot of a computer

Description automatically generated

There are a number of entries in the GameplayAbility.h related to montages; a given ability knows which montage is currently playing if we’re using the GAS version PlayMontageAndWait

This node will do what it says – play a montage and then wait for one of the outputs to happen

We can hardcode in a montage to play:

A screenshot of a video game

Description automatically generated

The new Montage is set to the default slot but we need to be using that slot, so we should double check in the AnimBP ABP\_Aura:

A screenshot of a computer

Description automatically generated

Yes, default slot is there, so if we play the montage we will actually see it played

Now we have created the Anim montage we can add it to the node in the GA:

A screenshot of a computer

Description automatically generated

We can confirm that the topmost execution wire is executed right away by printing a string

A screenshot of a computer program

Description automatically generated

A group of purple objects on a grey surface

Description automatically generated

The firebolt spawns immediately rather than in time with the animation

On the montage: Blend options we’ll update from 0.25 to 0 (no blending) and reduce the blendout to maybe 0.1

A screenshot of a computer

Description automatically generated

The print message is also played immediately on pressing because it’s from the top right output

But we can also do things later!

A screenshot of a computer program

Description automatically generated

A kite flying in the air

Description automatically generated

There’s a time gap between the cast, the blend out and the completed, allowing us to do things later in time without using timers – we can base activity on the length of a montage

How does this BP node work, become Async?

This appears to be a class based on UAbilityTask

A screenshot of a computer

Description automatically generated

“/\*\* Ability task to simply play a montage. Many games will want to make a modified version of this task that looks for game-specific events \*/”

The task has a number of delegates  
A screen shot of a computer program

Description automatically generated

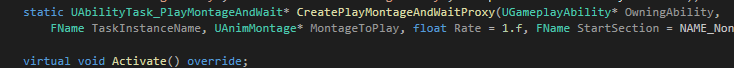
Those are the names of the output execution pins

When you create a Ability Task and add NP assignable delegates to it those become output execution pins

Next we have this:

A screen shot of a computer code

Description automatically generated

Static function 

“UFUNCTION(BlueprintCallable, Category="Ability|Tasks", meta = (DisplayName="PlayMontageAndWait",

HidePin = "OwningAbility", DefaultToSelf = "OwningAbility", BlueprintInternalUseOnly = "TRUE"))

static UAbilityTask\_PlayMontageAndWait\* CreatePlayMontageAndWaitProxy(UGameplayAbility\* OwningAbility,

FName TaskInstanceName, UAnimMontage\* MontageToPlay, float Rate = 1.f, FName StartSection = NAME\_None, bool bStopWhenAbilityEnds = true, float AnimRootMotionTranslationScale = 1.f, float StartTimeSeconds = 0.f);

This returns a pointer of type “UAbilityTask\_PlayMontageAndWait”

It’s BP callable, has a category and a meta with displayname (so the Blueprint will not show the “UAbilityTask\_” at the start.

It also has “HidePin = "OwningAbility"”, so the BP will not show an input pin for that parameter, which is also set to ‘Self’ by default (in C++, it would be ‘this’)

In the CPP file:

A screen shot of a computer

Description automatically generated

This function creates a New Ability Task with an internal function of the same name.specifying this particular task, and then setting a number of parameters on that new task and returning it

So these ability tasks have a static function that creates an instance of themselves and returns it after setting a number of input parameters that can be exposed to the BP

It’s made to be created in Blueprint, although the function could be called in C++, and we know it has a number of delegates

As soon as we create this ability task, ability tasks have an activate function and when they are activated, whatever happens inside this function is what happens

A computer screen with white text

Description automatically generated

PlayMontageAndWait will obviously play a montage

A computer screen with colorful text

Description automatically generated

It calls play montage on an ASC

How does it well which one? Ability tasks have an owning ability system component

A screen shot of a computer program

Description automatically generated

It’s also binding some function callback to some delegates (the output pins from the header file)

A screen shot of a computer code

Description automatically generated

It’s binding it’s own OnMontageBlendingOut function here

A screenshot of a computer program

Description automatically generated

The function calls ClearAnimatingAbility on the ASC, gets the character from the AvatarActor, checks authority, sets anim root motion scale etc etc

A screen shot of a computer program

Description automatically generated

It checks if it’s being interrupted and if so broadcasts that, same for onBlendOut

An Ability Task is a C++ class that has a static function – a factory of sorts – that constructs an instance of itself, that is BP Callable, and has a number of delegates that are broadcast based on things that are happening. It also has an activate function in which the delegates have some callbacks that are bound to them. These delegates can be broadcast based on whatever logic we implement here (this example happens to play a montage and then broadcast delegates when the montage is finished or other related cases)

The ability task gets the anim instance from the ActorInfo associated with the AbilityTask’s ability and owning ASC



So now we have an understanding of how this works.

But what if we want to wait for something that is NOT one of the cases specified here?

A screen shot of a computer

Description automatically generated

What if we want to wait until a certain point in the montage before we do the next thing like Spawn the projectile?

A screenshot of a video game

Description automatically generated

# Sending Gameplay Effects

We’ve seen some cool stuff in PlayMontageAndWait as an ability task, but how it fits together isn’t clear yet because we’ve not yet created one

Well, we just use a notify in the anim montage to indicate at what point the effect actor spawns

We’ll be making a custom anim notify that sends a gameplay event, not just a SpawnActor in a Blueprint – gameplay events can be sent along with data to a given actor and a gameplay ability can listen for events – there’s an ability task for that

A screenshot of a computer program

Description automatically generated

Deleting the test print strings and instead immediately after PlayMontage and replacing it with the ability task Wait Gameplay Event

A screenshot of a computer

Description automatically generated

This will wait for an Event to be sent to the actor that has the ASC that is activating this ability

We specify what event we are waiting for – event is identified by a gameplay tag. So we can create a gameplay tag designed to identify specific events eg Launching a Fireball

Since the tag is not one we need to access from C++ we can just add it to the project in Editor (non-Native)

A screenshot of a computer

Description automatically generated

It’s an Event tag, it relates to Montages and it’s for the FireBolt ability

A computer screen with a black background

Description automatically generated

If we want to spawn the Firebolt from the montage we’ll send an event that has this gameplay tag

A screenshot of a computer program

Description automatically generated

So we just need to send that event in a custom anim notify

New BP class of type AnimNotify

A screenshot of a computer

Description automatically generated

To be used for any montages that want to send a gameplay event

Function to override:

A screenshot of a computer

Description automatically generated

This function will be called when we reach the notify

A screenshot of a computer program

Description automatically generated

We have the mesh component associated with the animation montage that is executing this notify, so we can call get owner on it

We can also get from the menu Send Gameplay Event to Actor from the Ability system static library

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

As we are sending the tag to the actor, we can also send additional data in this payload struct…

A screen shot of a computer

Description automatically generated

And it matches this payload output in GA\_FireBolt, along with the Event Received execution wire

As soon as the owning actor receives the event, Event received will be executed and will receive the payload

Promoting the Event Tag to an instance editable variable

A screenshot of a computer

Description automatically generated

Payload:

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

Contains a ton of data, and ‘Target Data’ is itself another struct!

For now we’ll just send the tag

A screenshot of a computer

Description automatically generated

The event tag variable was exposed, so we can specify that here

A screenshot of a computer

Description automatically generated

No, the montage reaches the notify and triggers it, the Notify then sends the tag using Send gameplay event to Actor. The wait gameplay event was waiting for the event with the tag, receives it and then executes the Event Received output node

A screen shot of a graph

Description automatically generated

A grey tile floor with orange text

Description automatically generated

Of course the FireBolt is still spawning immediately, because it’s being done in the Activate Ability C++ - we’ll want to update this. The main thing is we’re receiving the event thanks to the notify which called the AS blueprint library function send gameplayeventtoactor – we can even call this in C++

Next: make sure we spawn the projectile only when we receive the event – it doesn’t look right when spawned immediately

A black background with white text

Description automatically generated

A screen shot of a computer code

Description automatically generated

Automatically called when the ability is activated and spawns the projectile. However we need to spawn the projectile a little later, so this is not the place to do it.

Instead make a BP Callable function so this can be done in response to a gameplay event and migrate most of the code into this function

A black screen with white text

Description automatically generated

A computer screen shot of a program

Description automatically generated

A screen shot of a computer

Description automatically generated

However we now do not have the ActivationInfo



No problem:

A screen shot of a computer

Description automatically generated

This will return an actor that is the avatar, if we have one. If so we can call HasAuthority and use that instead

A black background with white text

Description automatically generated

So this new function, SpawnProjectile, will check if we are on the server and if so spawn the projectile in response to an event. It’s BP Callable, so we only need to call it from the BP

A computer screen shot of a program code

Description automatically generated

The function is BP callable and GA\_Firebolt is a child of Aura Projectil Spell so:

A screenshot of a computer screen

Description automatically generated

The FireBolt now spawn as soon as the anim reaches the notify, not instantly!

This means that we can now trigger a gameplay effect in response to any event we choose in theory!

Now that we have spawned the projectile we can end the ability here, and then we can use it again

It’s not a long animation so currently it can be spammed, so we’ll need to add a short cooldown later, but there are some other issues to address

ToDo:

Projectile Rotation

Orient the character in the direction of the target

For this we need to know the location in space of the target

# Custom Ability Tasks

In order to orient the character and get the correct vector for the fireball the best info is the data from the cursor hit result.

We can create an ability task that will get this data for us

In GA\_FireBolt we could get the Cursor hit result as soon as the ability is activated (or the location associated with it), and use that. Test in single, then adapt to make it work in multiplayer as well

New AbilityTask class!

A screenshot of a computer program

Description automatically generated

It’s based on GameplayTask and note that the engine has a number of AbilityTasks already built-in

Note the use of the name ‘Target Data’ – in GAS Target Data has a specific meaning, more later!

A computer screen with white text

Description automatically generated

We’ve already seen PlayMontageAndWait and saw that I had a static function that created an instance of the class and returned it. We’ll follow that pattern here

A screenshot of a computer

Description automatically generated

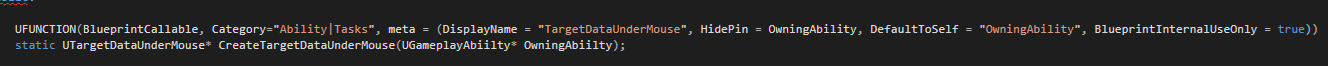
The function will receive the owning GameplayAbility, which is typical for AbilityTasks, they take in a UGameplayAbility pointer

A screen shot of a computer

Description automatically generated

When we call this function from within a GameplayAbility this parameter should be passed in by default and it should be the self pointer (in BP it’s self, in C++ it’s this)

UFunction macro:



The function is Blueprint Callable in a category

The meta means that the node when called in BP uses this name

The Owning Ability pin is hidden, and defaults to self

It’s an internal function, basically a factory

A screen shot of a computer

Description automatically generated

So we create a new AbilityTask of type TargetDataUnderMouse, save it to a pointer named MyObj and return that

We could also set any number of variables on it, could take in any number of input params to be passed in on the BP etc

Let’s see what it looks like!

A screenshot of a computer

Description automatically generated

Currently all this does is return the instance of the TargetDataUnderMouse object. Not much we can do with that…we can call GetObjectName and print to screen, that’s about it

A grey tile floor with blue text

Description automatically generated

We can hook it up, but we need it to return some data

Not the clock icon in the corner, this is an asynchronous output node, so it can have multiple output pins

These output pins recall are created by broadcasting delegates, so any output execution pin needs to exist as a delegate variable on this class



A black background with white text

Description automatically generated

A screenshot of a computer

Description automatically generated

Now there’s an async output and a data pin, which is an FVector by reference! :D

Right now ValidData isn’t going to be executed, because the only way to execute is to broadcast the delegate, and when the delegate is broadcast is when we get the output from Data

To trigger this broadcast, we will need to Override the AbilityTasks’s Activate function!

Then, as soon as the BP node is executed, Activate will be called

Activate is originally defined in GameplayTask (not even AbilityTask!)

A screen shot of a computer

Description automatically generated

As we can see in the definition

A screen shot of a computer program

Description automatically generated

The parent of this function just makes a log note; no point calling Super, doesn’t do much

A black and white text

Description automatically generated

AbilityTasks have an internal variable Ability, and from this we can GetCurrentActorInfo

A black screen with white text

Description automatically generated

A screenshot of a computer program

Description automatically generatedThis returns FGameplayAbilityActorInfo

Which has a number of variables on it including the player controllers



Player controller is a TWeakObj pointer so we need to .Get() on it



Now we can create a local FHit and pass it in



Finally, we broadcast the data (the location of the hit result) with the delegate

A screen shot of a computer program

Description automatically generated

So, as soon as we activate the ability, we broadcast the delegate and the ValidData execution pin will be executed (but it doesn’t HAVE to be, it’s an async task, so if a game mechanic calls for a delay we can do this)

A screenshot of a computer

Description automatically generatedA screenshot of a video game

Description automatically generated

**In multiplayer however:**

A screenshot of a video game

Description automatically generated

The debugsphere draws OK in client, but not on the server, so so the same would be true for the firebolt target!

Actually, it is being set…

A screenshot of a video game

Description automatically generated

At the world origin, indicating the the data is empty!

That’s because the click is only local; so the server doesn’t know about it

A screenshot of a computer

Description automatically generated

Activate Ability is being activated locally, and because Abilities are replicated, it’s also being called on the server. But on the server the delegate is broadcast even though we don’t know where the mouse cursor is!

Next step: get data (mouse cursor location) from client up to the server in a reliable way