Code Based Leaning

Documentation

Version 1.6.5
Enhanced Flexibility

Documentation made by : Chopstig

Contents | Code Based Leaning

Component Setup | Documentation

Setting up this Plugin is actually pretty easy 😃
For now the only thing you need to do is attach the Component to your Character
[Image 01]

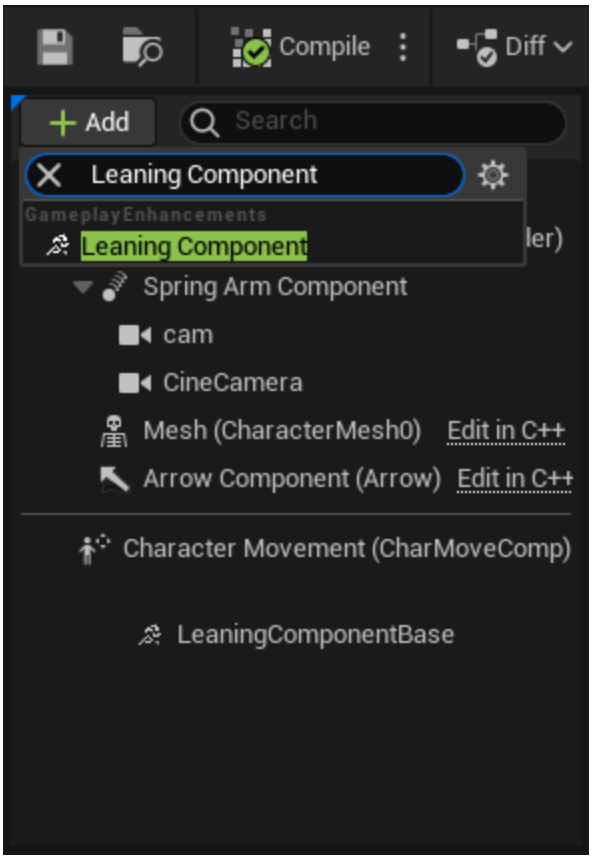


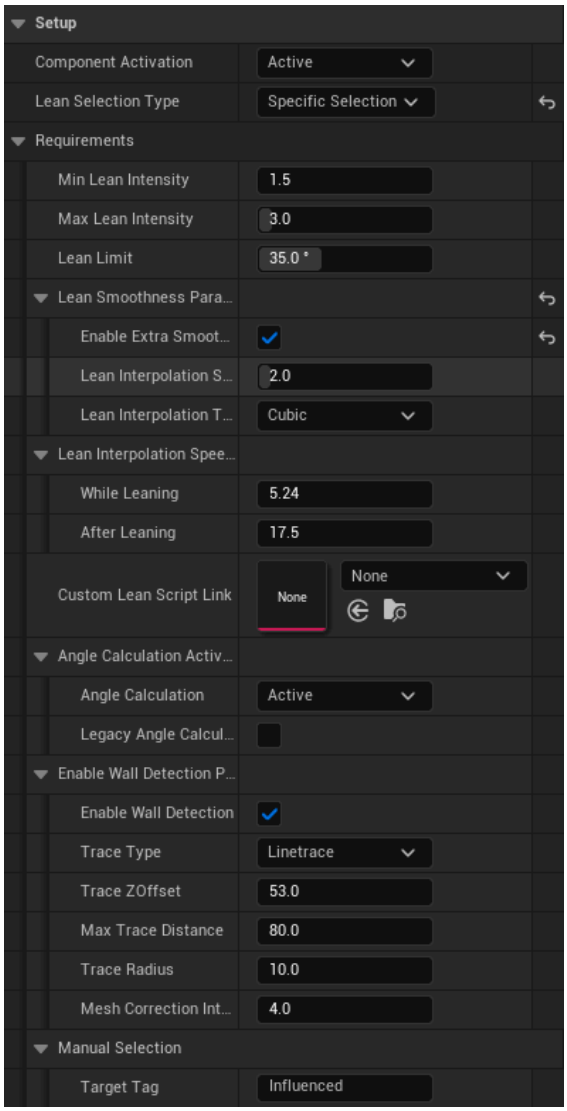
Image 01

You do not need to do anything else ! Leaning will automatically work
[Result : Image 02*]*

Inside the **Blueprint** | Documentation

On the right side of the Blueprint Viewport is the **Details Panel**. This is where you can change anything to suit your needs. Intensity, Limits, Smoothness, Interpolation, Interpolation Type, Collision Detection etc. etc.

In *[Image 03]* you are able to see each Parameter available for you to play with.



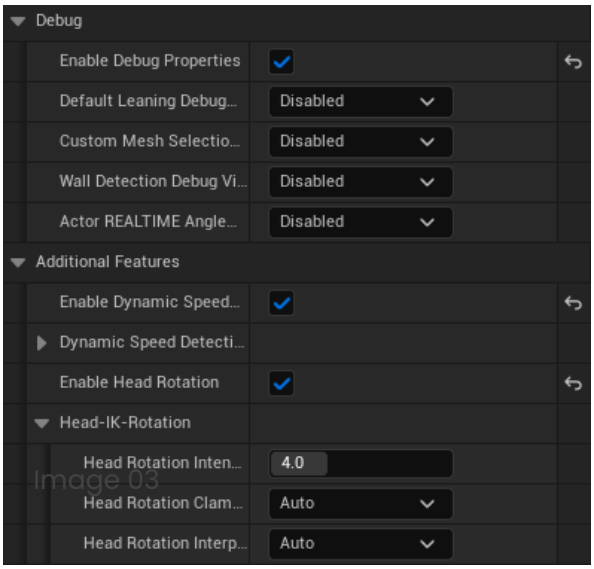


Image 03

Most of the Parameters are **self explanatory** therefore you may not need the detailed explanation…
Example Parameters :

1. ***Lean Intensity*** *[**Min** and **Max** Variants are the same; will get explained in a different Part of the Documentation],*
2. ***Lean Limit**,*
3. *All Parameters in the **Enable Wall Detection Params** Section,*
4. *All Parameters in the **Head-IK-Rotation** Section*
5. *All Parameters in the **Lean Interpolation Speed Params*** Section

Here are some brief explanations on how to use those Parameters

Lean Intensity
The Intensity the Character leans.
The lower the Value the more intense the Leaning is.

Lean Limit
Clamp the Lean Rotation to a certain Angle.

Enable Wall Detection Params
Click here to get redirected to the proper section

Head-IK-Rotation
Click here to get redirected to the proper section

Lean Interpolation Speed Params
You can set the Speed of the Lean Interpolation
There are two Float Values, one when the Character is currently moving ***(while Leaning)*** and the other when he just stopped ***(after Leaning)***.
Altering those Values will heavily change the Characters behavior when Leaning.

Inside the **Blueprint** | Documentation

Custom Mesh Selection

One Customer once asked how to implement Leaning when having a modular Character Mesh *[Image 04]*

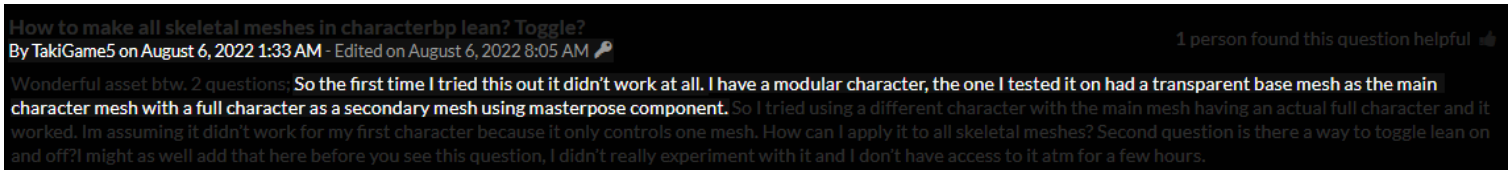
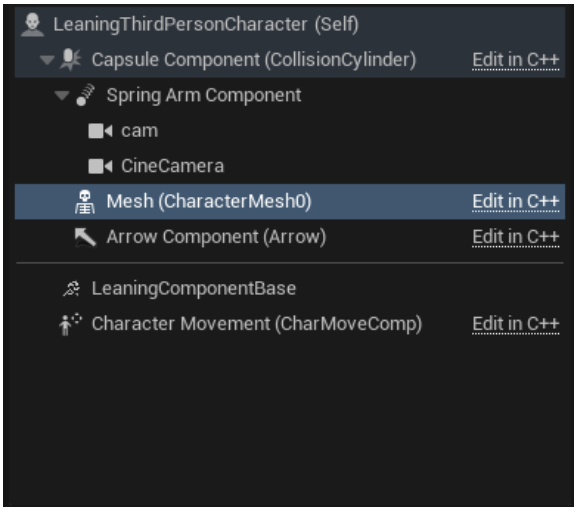


Image 04

So with this new addition **you** as the Developer are now able to easily select Meshes in your Blueprint Hierarchy to get affected by the Leaning Component.

This System is **Tag - Based** meaning that you need to Tag each Mesh in your Hierarchy *[Image 05]*



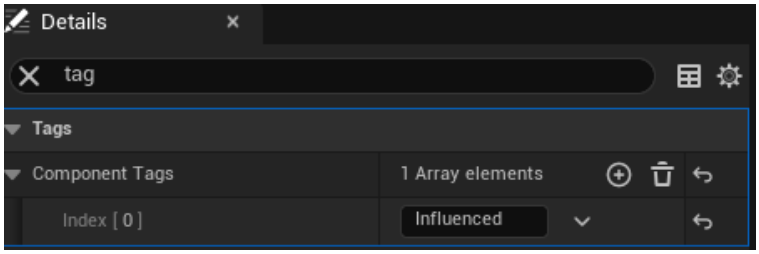
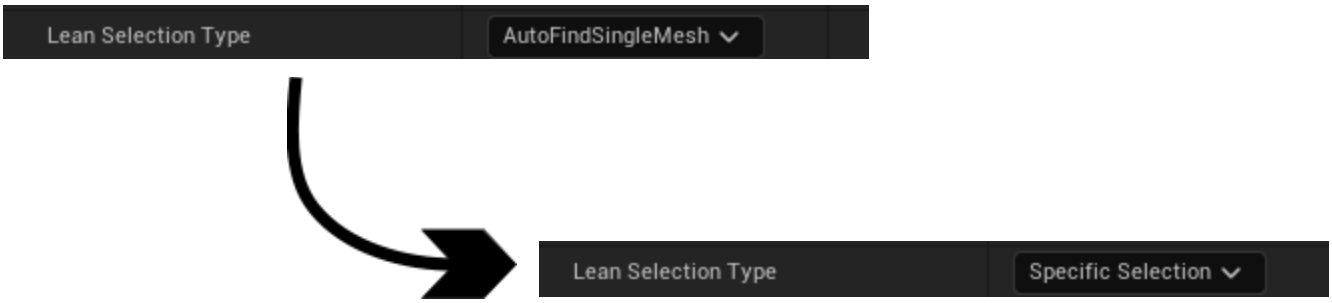


Image 05

After selecting each Mesh you wanted to get affected by the Component you need to enable the Option itself.
The Second Parameter in the Setup Section *(Lean Selection Type)* is by default set to **“AutoFindSingleMesh”.**



ℹ️ A new Detail Section should appear by enabling this Option.
[Image 06]

Image 06

(This should appear below the **“Enable Wall Detection Params”** Section)

🆘 ***If it does not show up, then just recompile the Blueprint***

The new Section that just showed up is where you can set your Tag.

This is **Context Sensitive** meaning that each Character must exactly match with each Mesh you want to apply the Leaning on !

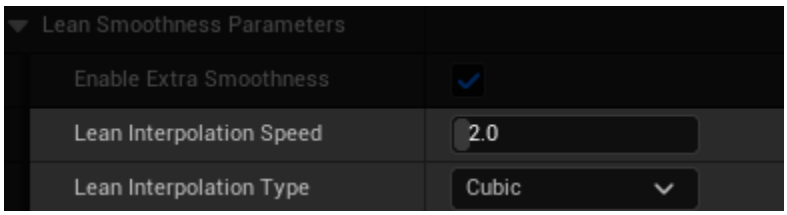
Inside the **Blueprint** | Documentation

Lean Smoothness Parameters

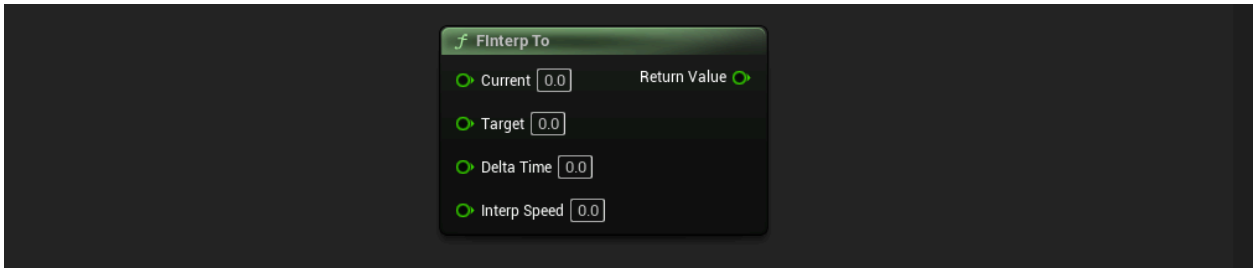
You want your Character to lean even Smoother.....
Well with this addition you can make Leaning feel just as smooth as you desire.
(Using this felt just like enabling V-Sync for me 😂)

You can do that by first enabling this boolean

Enabling it will expose two more Properties.



Lean Interpolation Speed



The complete Interpolation is based of this Function *(it is actually the **FMath** Function in C++ but it is pretty much the exact same 😁)*

The **Lean Interpolation Speed** Float Value is just the **Interp Speed** Float Value in the **FInterpTo** Function

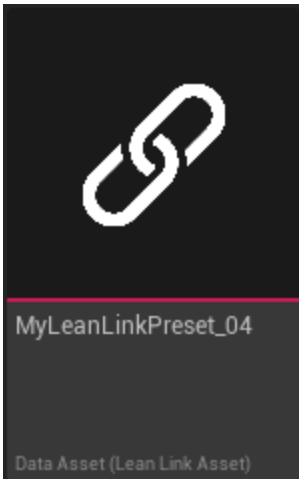
- ☐ The higher the Value the less smooth the Interpolation is

If you want to get more detailed Information about the **FinterpTo Blueprint** Function click **here**

The second Parameter *(Lean Interpolation Type)* does pretty much what it says. It can change the Interpolation Type.

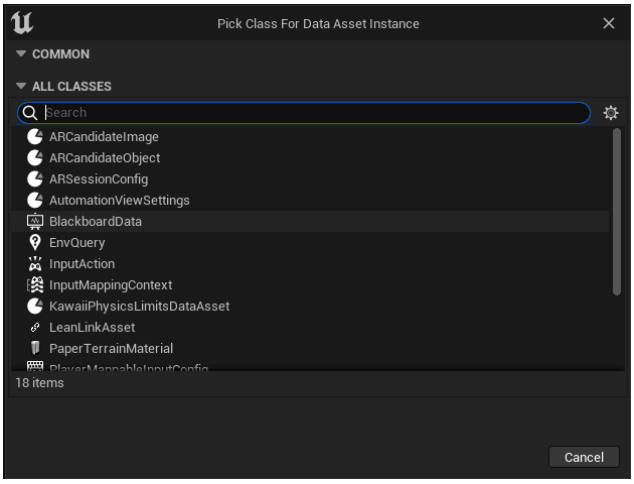
You can imagine it like it is shown in *[Image 7]*

Image 07



You can create a **Custom Lean Link Asset** where you can store and reuse certain Values for certain Characters.

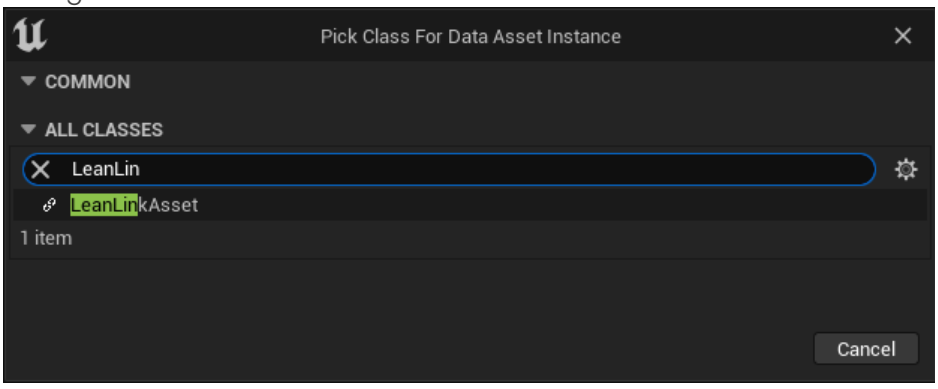
To create this just Right Click in the Content Browser go to the Miscellaneous Tab and click the **Data Asset** Asset.



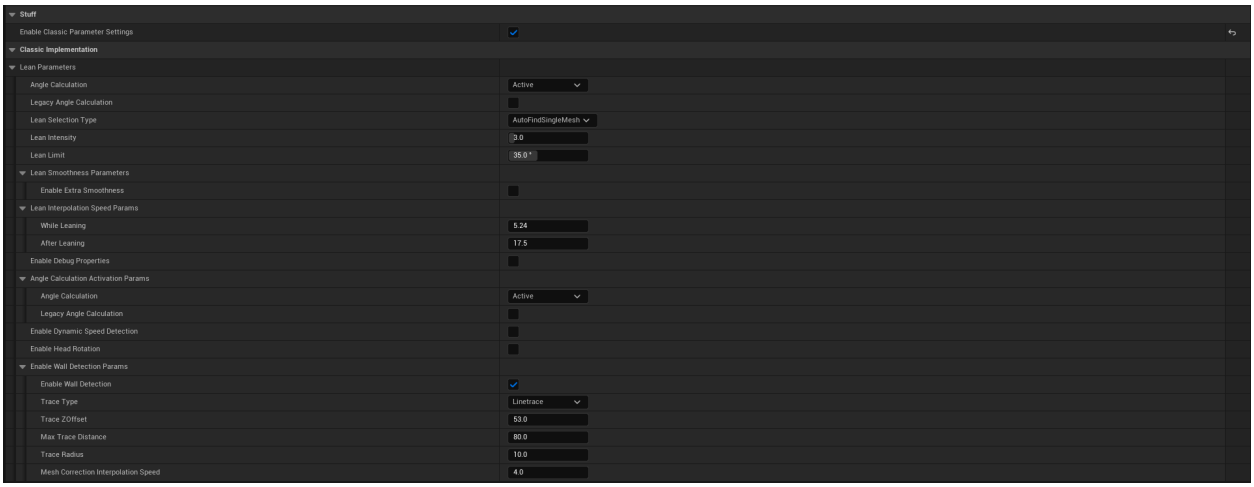
You then will be greeted by this small Tab where you can choose which **Data Asset Instance** you want to pick.

In this case we want to select the **Lean Link** Data Asset Instance. *[Image 08]*

Image 08



Then, after selecting the proper Class you'll face pretty much the exact same Interface you've already seen in the Blueprint Details Panel.



Everything looks the same...

You may ask, why does it even exist ?😜
My Answer is... to decrease the Development Time and hassle of the Developer **(you)**

With it you just have to set those Values **once**.
Then you can set the **Custom Lean Script Link** Value as the created Lean Link Asset and done ! *[Image 09]*

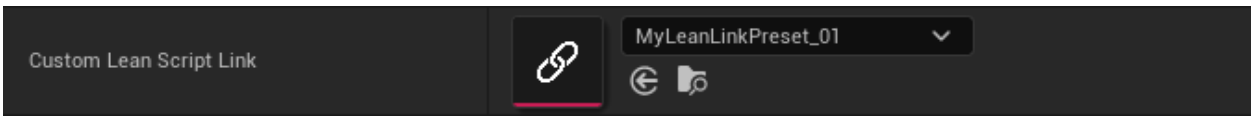
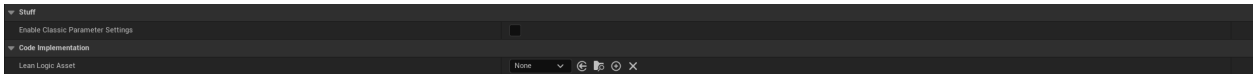


Image 09

But there is something else you can do with this Asset...



When disabling the Enable **Classic Parameter Settings** Boolean you will see that each Parameter is removed and replaced by a single **Class Reference**.

In it you can set a Lean Logic Class.
This Lean Logic Class will allow you to Code certain Logic for your Character.

Inside the **Blueprint** | Documentation

Angle Calculation Parameters

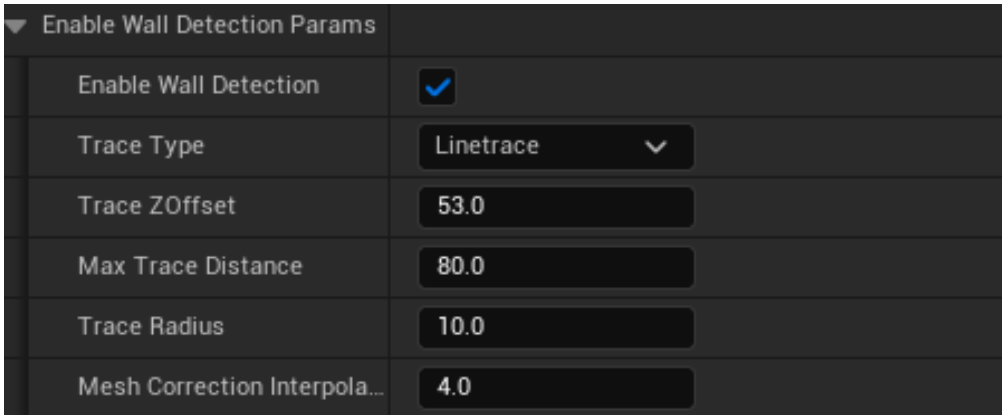
By enabling this Setting you will allow your Character to use the **Angle Calculation Rotation System**.

This was recently **replaced** by a newer Alternative but I was not sure if this would satisfy you all.

That's why I implemented the functionality to simulate the old Rotation System. **(Legacy Angle Calculation** Boolean)

Inside the **Blueprint** | Documentation

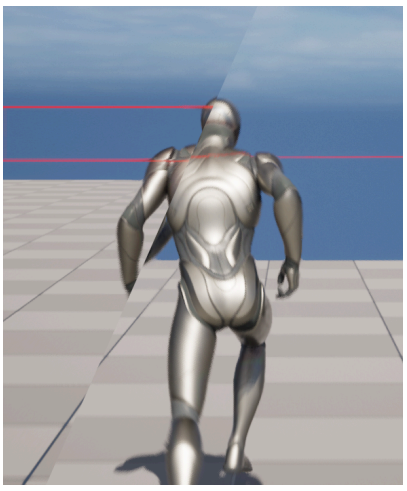
Wall Detection System



Wall Detection was pretty much the largest Part of Version 1.6.0
And also the first **Teaser** of 1.6.0 contains this System

Trace Type (Hover over that Property to get more detailed Information)
You can choose if your Character should have more accurate or more realistic Leaning.

TraceZOffset
This is the Z Location of the Trace Start Location. You can alter it. By default its Value is 53

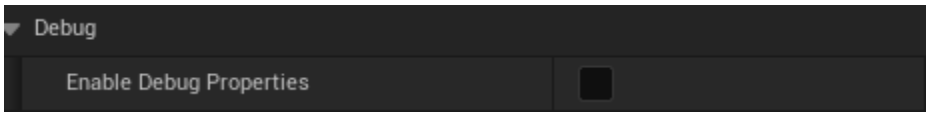


Max Trace Distance
The maximum Trace Distance from the center to the right/left Side

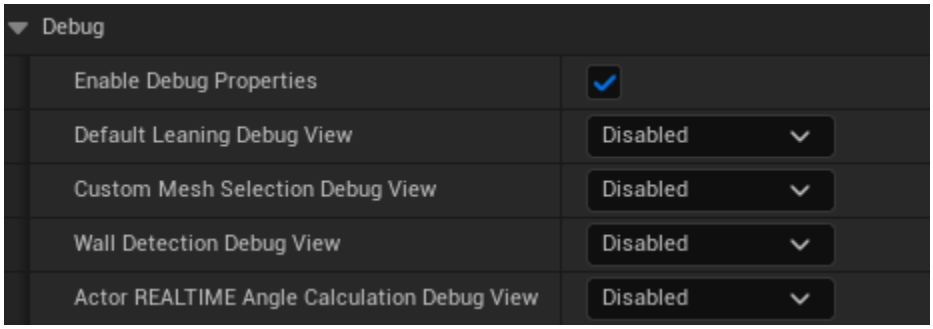
Trace Radius
[Only usable when enabling **Box Trace** in the Trace Type property]
This will in/decrease the Radius of the Box Trace

Mesh Correction Interpolation Speed
If hitting a Wall the Character should quickly Interpolate to the nearest free space > Control the Speed of it.

Debug Parameters



Enable this Boolean to get access to all the Debug Properties.

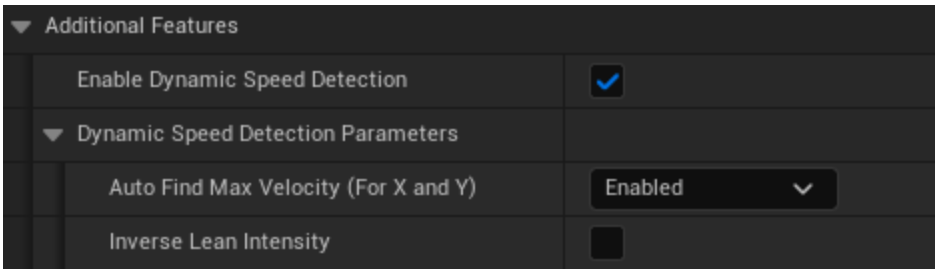


Each of these properties will show everything that is related to what it says in the Debug Section

E.g. Wall Detection Debug View -> This will show the Line/BoxTraces of the Character and all Prints related to this System

Additional Features | Documentation

Dynamic Speed Detection Parameters



This System allows you to dynamically in/decrease the Lean Intensity based on the Velocity of the Character.

You can either set the Max Velocity or let the Code find it.

By enabling this Boolean you will change some Values in the Details Panel. [Image 10]

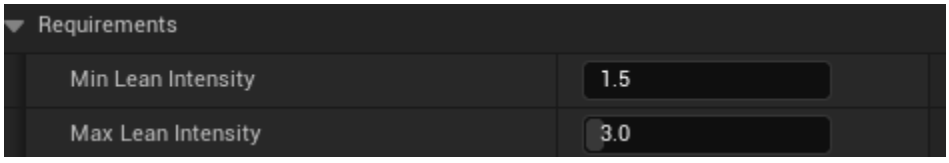


Image 10

Min and Max Lean Intensity Interval.
You can select how fast your Character should lean when moving really slow and when reaching its Max Velocity.

By enabling the Inverse **Lean Intensity Boolean** the Lean Intensity Interval will just change the Start and Finish Value

Inverse Lean Intensity = false;

Starts from 1.5 and Ends at 3.0

Inverse Lean Intensity = true;

Starts from 3.0 and Ends at 1.5

Additional Features | Documentation

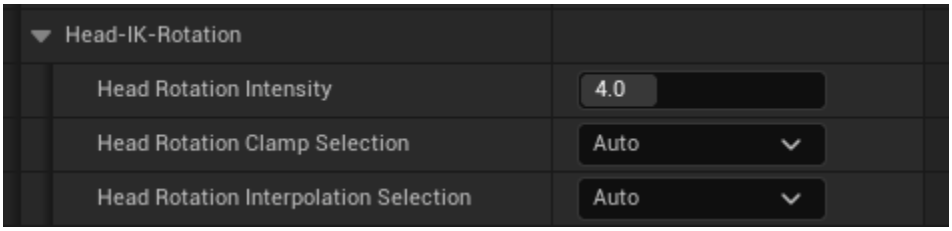
Head Inverse Kinematics Rotation

Unfortunately this System will take a bit more Steps to properly implement it.

😭

This will be done in Steps to make it more clear....

Enable the **Head Rotation** Boolean



Head Rotation Intensity

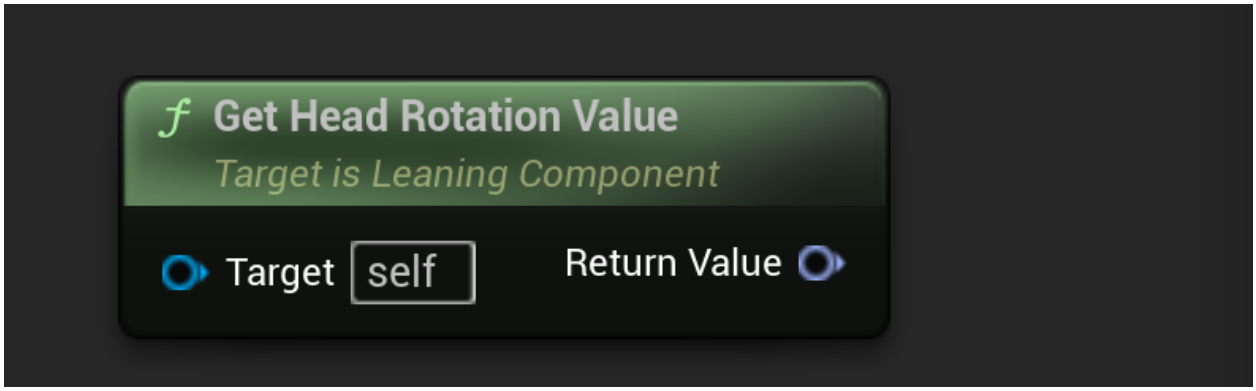
The Intensity of the Head Rotating

Head Rotation Clamp Selection

The Max/Min Rotation where the Head is allowed to rotate

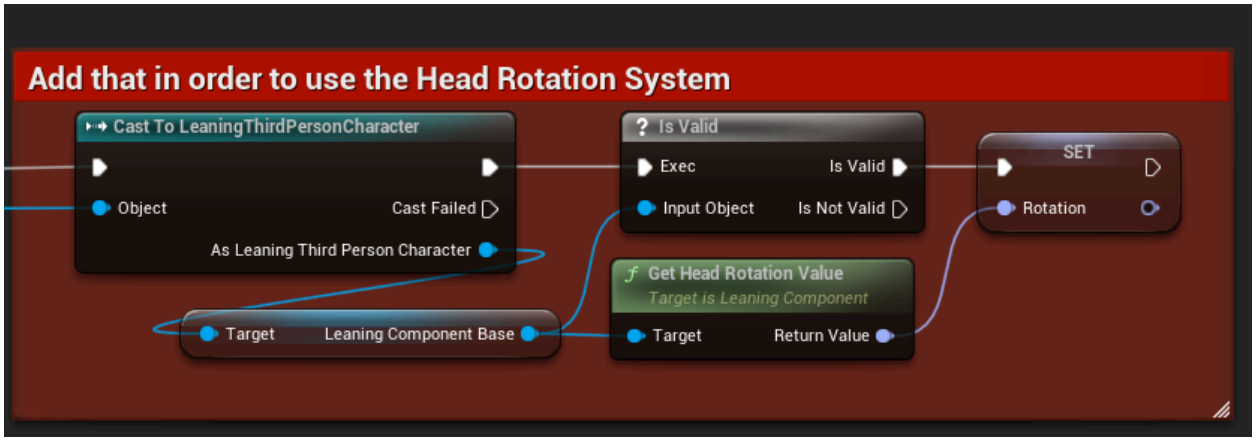
Head Rotation Interpolation Selection

Head Rotation Interpolation -> Selection (while Leaning, After Leaning)



You now can get access to this Function 😃

However you can only really use it inside the Animation Blueprint, due to the fact that this was made to alter the Rotation of the Head and unfortunately there is no such Function in Regular Blueprints *(from my knowledge at least)*



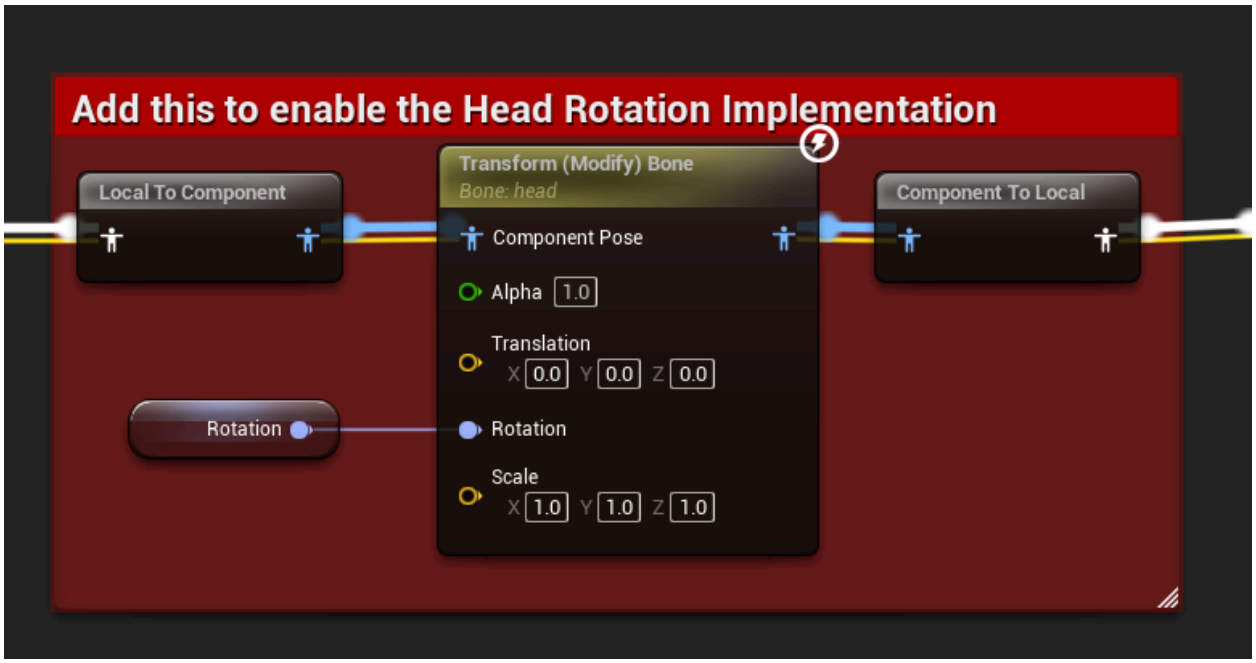
Go to your Animation Blueprint (to the Event Graph) and add this Code.

Please adapt the Cast to your Character

[You can use an Interface to slightly improve Performance]

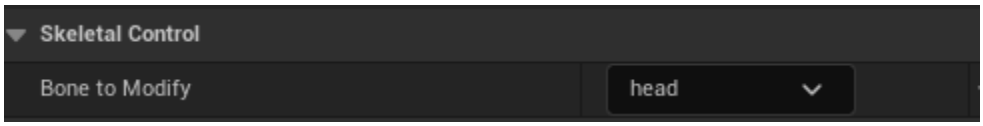
You now **successfully** implemented the Rotator for the **Head Rotation**

Then go to the Anim Graph and add the **Transform (Modify) Bone** Node.
(Tip > do this at the very last of your AnimBP in order not to interfere with the rest of your Code)

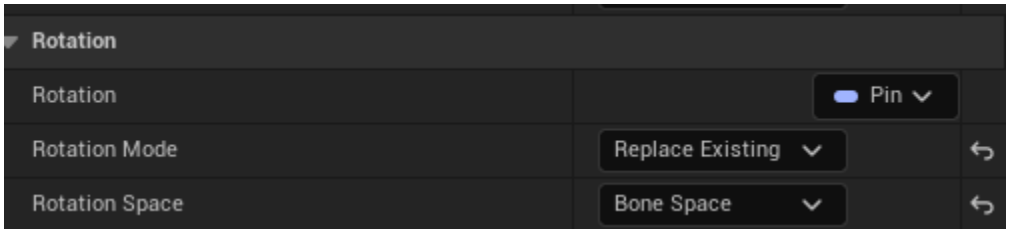


This should then look something like this.

Inside the Node , select your Target Bone > in this case it is the **Head Bone**



Then you only need to set these Values...



...and you're done. Your Head should now rotate when you're moving ! 🥳

Lean Link Asset | Documentation

Lean Link **Preset Generation**

This Additional Feature was made to kill the issue when altering the Values in the Editor Viewport.

You can now just **alter** the Lean Component in the Viewport, test it quickly and then if you're satisfied you can generate a **new Lean Link Asset** which then contains all of your altered Values.

In **Unreal Engine 5** you can find the Button in the Tools Bar in the Editor Viewport
In **Unreal Engine 4** it is in the Windows Bar (also in the Editor Viewport)

You may then face a Window where you have to select the **Data Asset Instance**.
> You then have to select the **Lean Link Data Asset Instance**

Lean Logic Class | Documentation

C++ Version Call Lean Logic Function

The Lean Logic Class was made to maintain the readability of your **Blueprint/C++ Code**.
So, this Class is able to do pretty much anything a normal Function can do, but with some extra Functions directly from Chopstig.

You can simply create a new **Blueprint Class** of Type > **Lean Logic Asset** *[Image 11]*

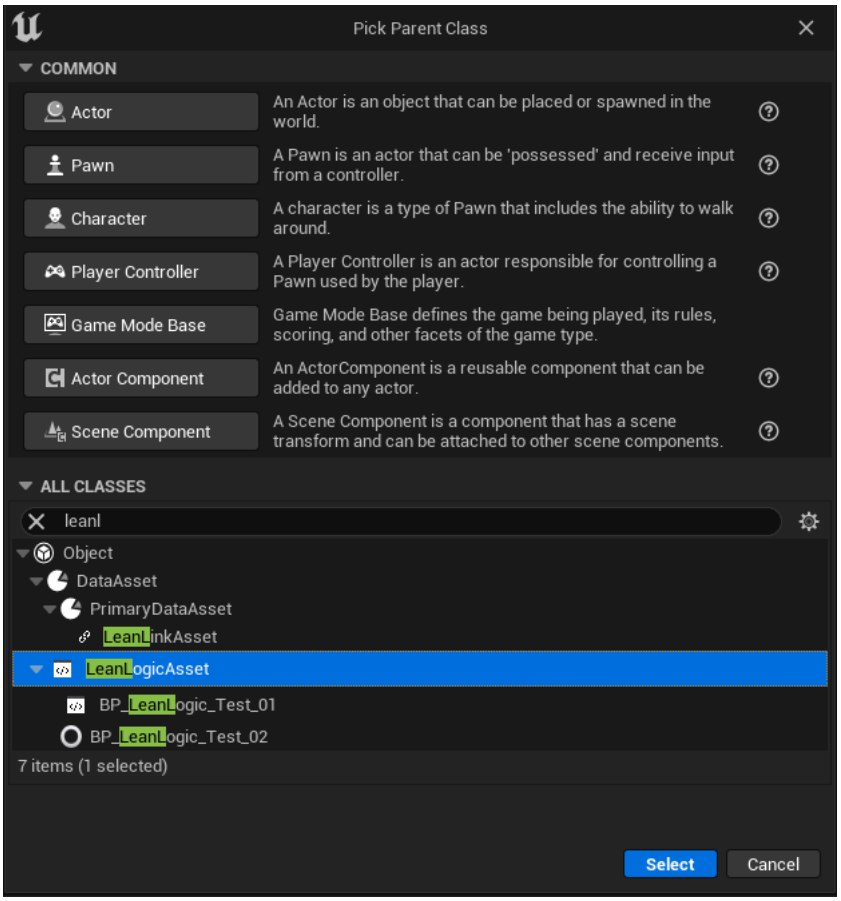
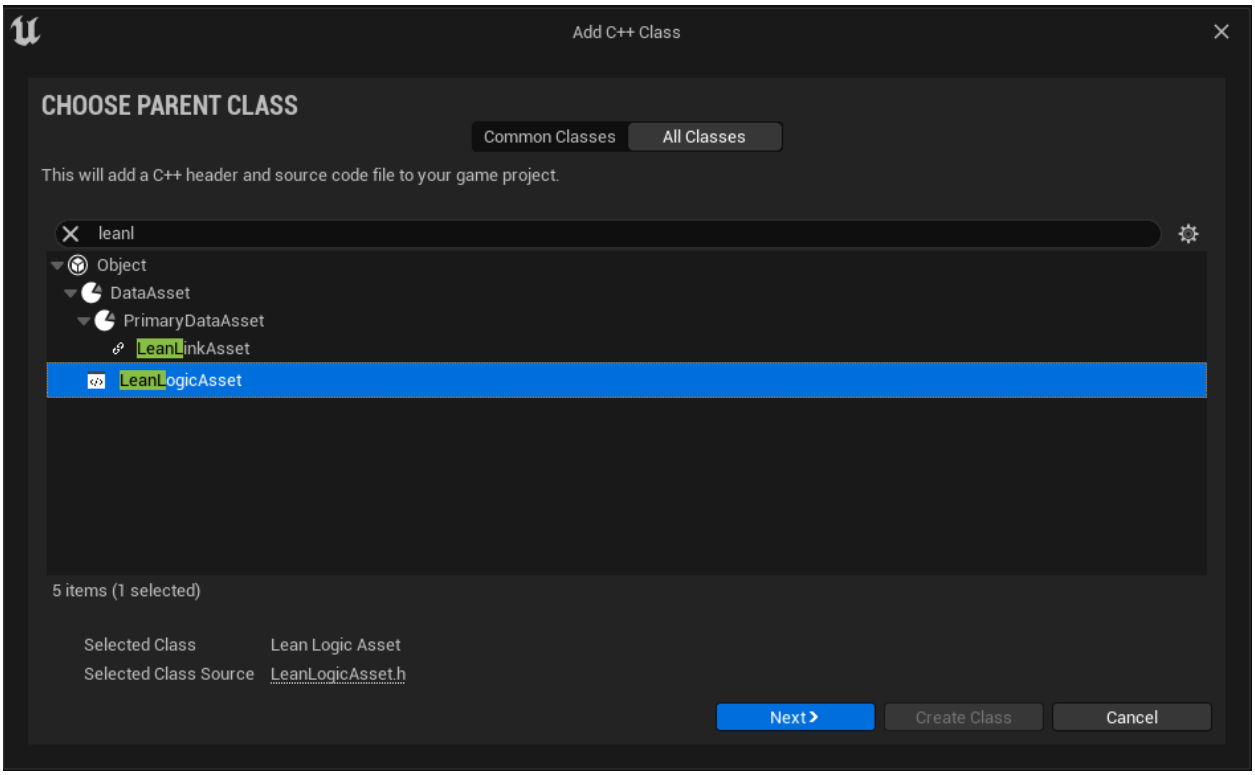


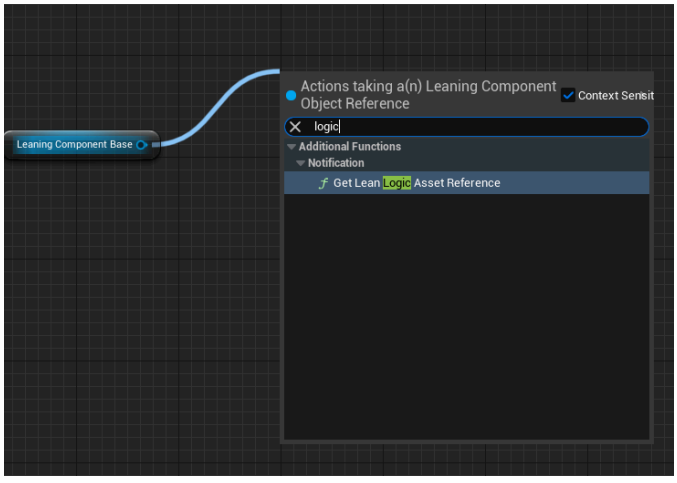
Image 11

For C++ it is the exact same Process of creating a simple Class based off the Lean Logic Asset *[Image 12]*



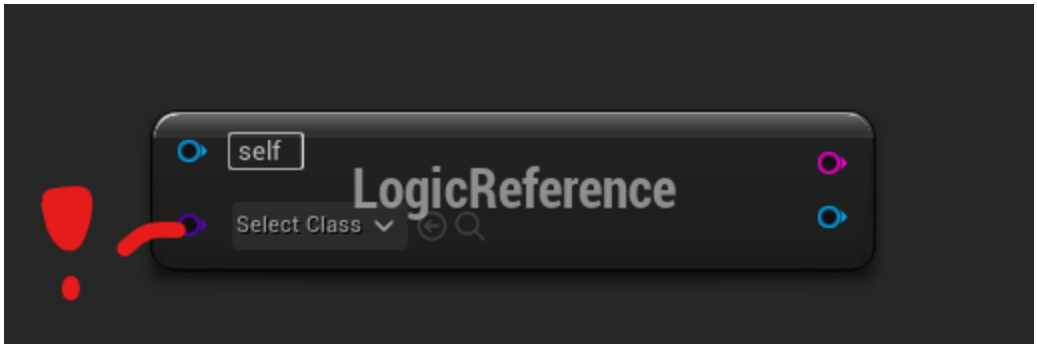
Calling a Function inside the Lean Logic Class is really simple.

1. Create a Custom Event inside the Lean Logic Blueprint Class
2. Go to your Character Blueprint or wherever you get access to the Leaning Component
3. Drag of the Leaning Component Reference and call the Function : **Get Lean Logic Asset Reference** *[Image 13]*



In the **Get Lean Logic Asset Reference** Function you need to specify the Lean Logic Class.

Insert the exact same **Lean Logic Class** that is used in the Lean Link Asset which got linked up to your Character Blueprint/C++ Class



The **Output** is then dynamically adapted to the Class that you have selected.
Drag out of it and then you should be able to Call your Custom Event.

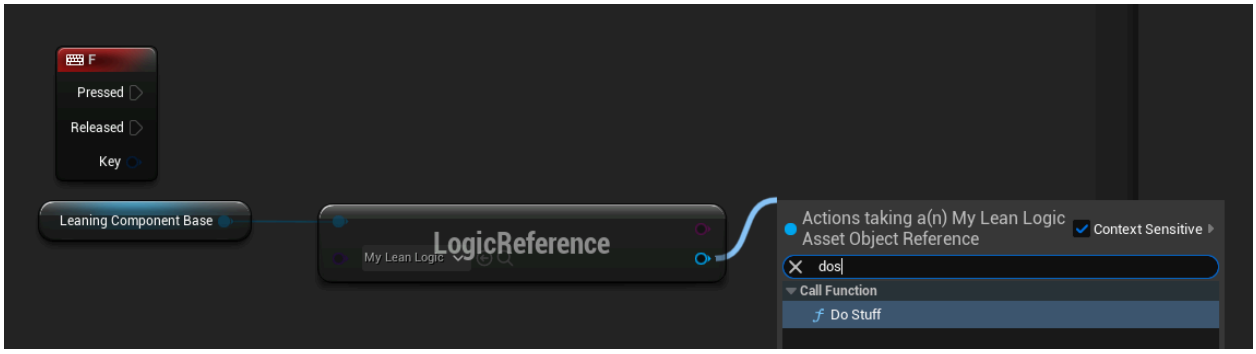
The complete procedure in C++ is **nearly** the same as if you would do it in Blueprints.

1. Create a C++ Class based on [..]
2. Create a Basic Function with the **BlueprintCallable** Macro

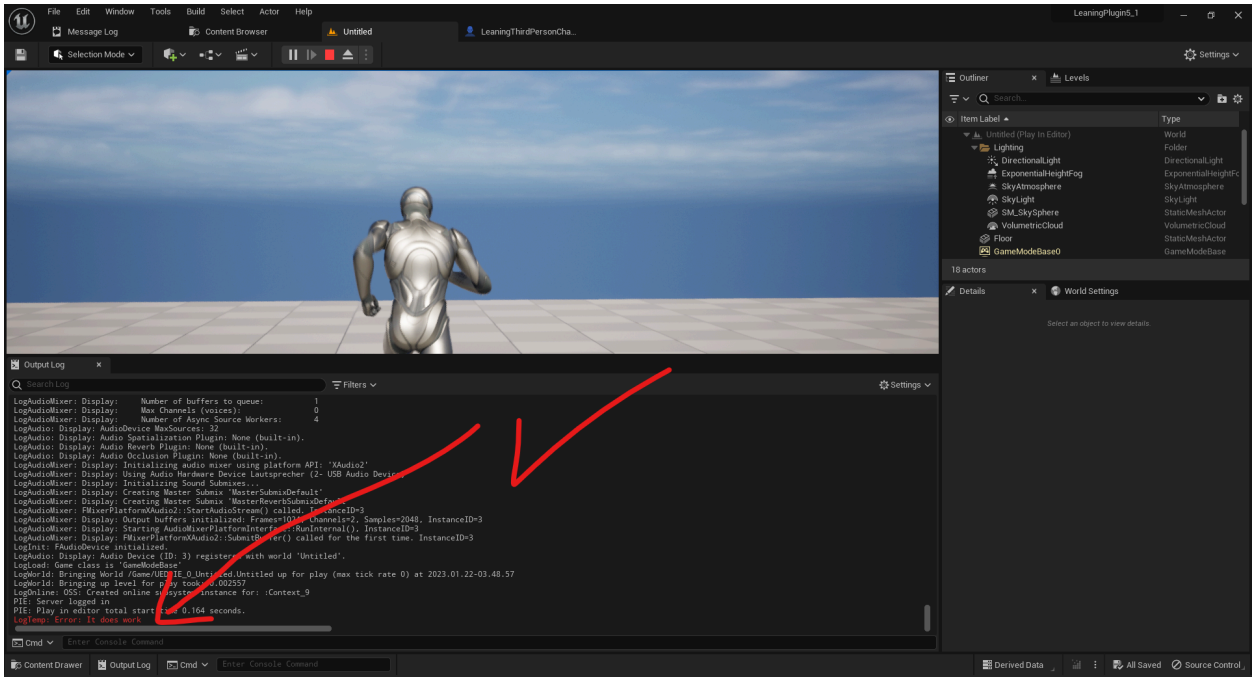
```
UCLASS()
0 derived blueprint classes
class LEANINGPLUGIN5_1_API UMyLeanLogicAsset : public ULeanLogicAsset
{
    GENERATED_BODY()

public:
    UFUNCTION(BlueprintCallable)
        void DoStuff(double JustSomeValues)
    {
        UE_LOG(LogTemp, Error, TEXT("It does work"));
    }
};
```

3. Get the Lean Logic Reference Function and do what is mentioned here

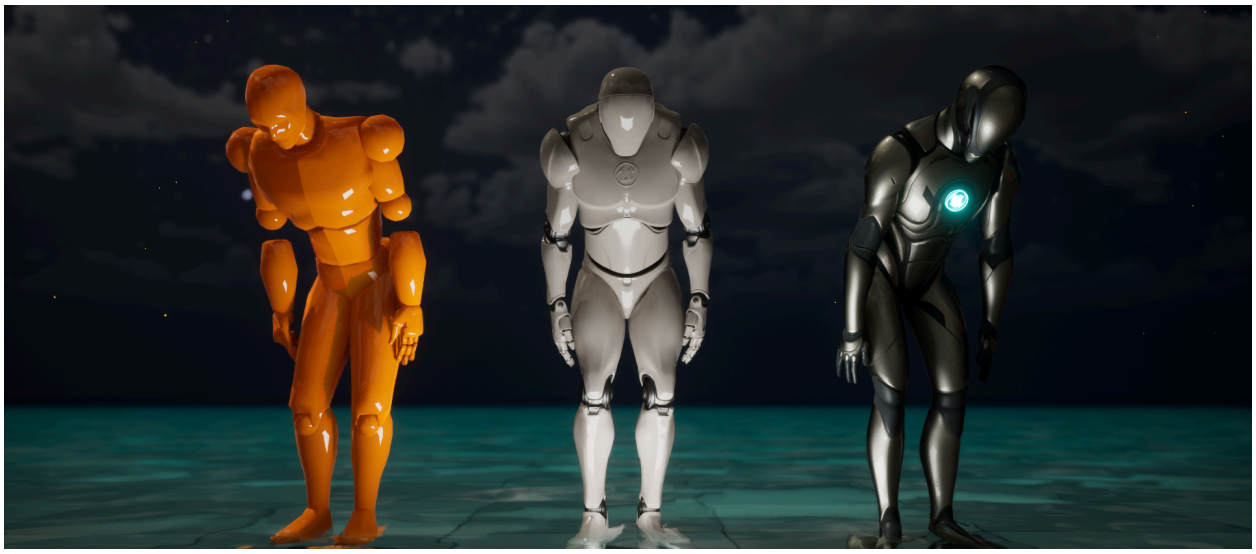
4. After compiling the Code you now should be able to Call the Function inside your Blueprint Class *[If you want to call this Function in C++ you then do not need to compile the Code]*



Then run the Game and test if the Function actually gets called...



And it fortunately does work perfectly. 😁



Thank you for everything and good luck on your next Project 😁

Cheers 🥂
Chopstig from Silvergull Studio