SoulsController – Documentation

A Dark souls inspired character using animation driven actions.

This document should give all the necessary information to work with the *Soulscontroller* project. However, Unreal engine 5 knowledge is expected as a pre-requisite to working with the project and I recommend checking out their documentation. If you want to use the character rig to animate, general animation skills with blender are required.

READ ME – Before attempting to add content or create separate logic I recommend taking a look at both the showcase and demo level. Please mess around within the blueprints and animation notify system to gain a deeper understanding of when and what is able to be adjusted without breaking.

Content

1. Revision history

If any changes will be made post-release of this document, this will be documented here.

Name	Date	Changed description	Version
Tom Baas	18/01/2023	Reiterated 5.4, added a full new step on how to make an event with a specific function. Added step counters on all.	1.4
Tom Baas	18/01/2023	Reiterated on 5.2. Added step counter a tip and another image.	1.3
Tom Baas	18/01/2023	Reiterated on the conditional attack setup in 5.3. also added steps amount in 5.3	1.2
Tom Baas	18/01/2023	Made the data names of 5.4 bold to emphasize them.	1.11
Tom Baas	17/01/2023	Added interactive table of content due to feedback.	1.1
Tom Baas	16/01/2023	Finished 1st version.	1.0

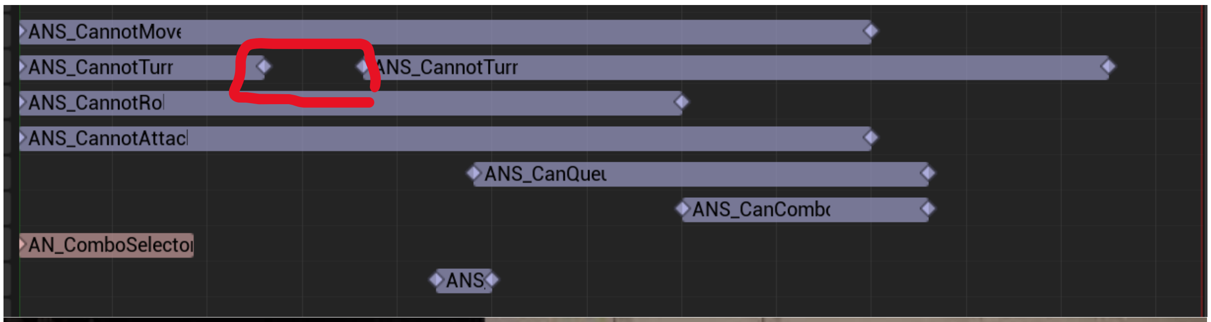
2. What does *SoulsController* contain?

2.1 Dark Souls character using animation driven actions

This blueprint character includes:

- The ability to move (using Unreals Character movement component)
- An animated character.
- Animation driven actions (Uses animation notify states)
- A resource system (stamina. Is visually presented in the WBP_Hud blueprint)

This character uses root-motion for actions but doesn't use root-motion for movement, thus it is. The actions revolve around having set animation states which can tick on and off depending on the animation state notifies inside of the animation montage. For example, like Dark souls, the player can only turn during a specific part of the windup of an attack. I've recreated this and it looks a bit like this:



[Video shows that the turning, during the attack, can only happen when the specific animation notify state called ANS_Cannotturn isn't active.]

All action animations are built around this structure. Every action uses all of the main animation state notifies but can fluctuate depending on the action how many secondary notifies they use. These **main animation state notifies** are:

- ANS_CannotMove	[Cannot move while this state is active.]
- ANS_CannotTurn	[Cannot turn while this state is active.]
- ANS_CannontRoll	[Cannot roll while this state is active.]
- ANS_CannotAttack	[Cannot attack while this state is active.]

The **secondary animation state notifies** are:

- ANS_CanQueue	[Can queue a combo during this state.]
- ANS_CanCombo	[Can start a combo during this state.]
- ANS_ComboSelector	[Sets the selected next combo attack should be.]
- ANS_AttackCondition	[Sets state for a conditional attack e.g. roll attack.]
- ANS_HitDetection	[Starts and ends hit detection during this state.]

When and where the montage starts isn't fairly important, but if the notifies are setup properly the system should be very robust.

2.2 Weapon system

All the animations, damage and combat-related stamina values are held within the weapon. The weapon is a child actor component on the skeletal mesh of the character.

the main adjustable variables are put into "**profiles**" these are either structs or maps holding the necessary data. Currently there are three profiles in the weapon blueprint. These profiles mean that depending on the weapon that the player is holding, he can have a widely different animation set with also differently setup notify tracks, which change even more.

These profiles are:

- Damage Profile	[Holds all damage values per type of attack.]
- Animation Profile	[Holds all the animations the character uses.]
- Stamina Profile	[Holds all stamina values per type of attack or action.]

The weapon **hitbox** is generated on the weapon mesh between two sockets called Tip and base. *More on this in section 5.2.*

2.3 Character rig

I'm going to be honest; I am not a professional rigger, and I am not good at doing 3D modeling. However, I think I created a human character rig that is decent at doing root motion and weapon animation.

The reason for this, is because the feet control bones aren't in the same hierarchy as the root bone. Which means that when the root moves, the feet don't follow. This allows for good quality animations where the feet actually look like they step on the ground instead of sliding and floating. I'm getting away with not having the control bones in the hierarchy because they aren't being exported. *More on this in section 5.3.*

The rig is also good at doing animations with weapons. This is because the weapon has its own bone, to which the hand control bones are parented to. This means that animations are able to be animated from the perspective of the weapon and not the hands. The hands stay cleanly attacks and it's easy to animate the track of the weapon.

2.4 Notable functions and macros

The project has some notable functions and macros that get used quite often and I recommend using them / changing however you like. These are all present inside of the character. These are:

Functions

- Perform attack	[Uses enums to reduce stam, get attack and play montage.]
- Get anim instance	[Simple shortcut to get the anim instance]

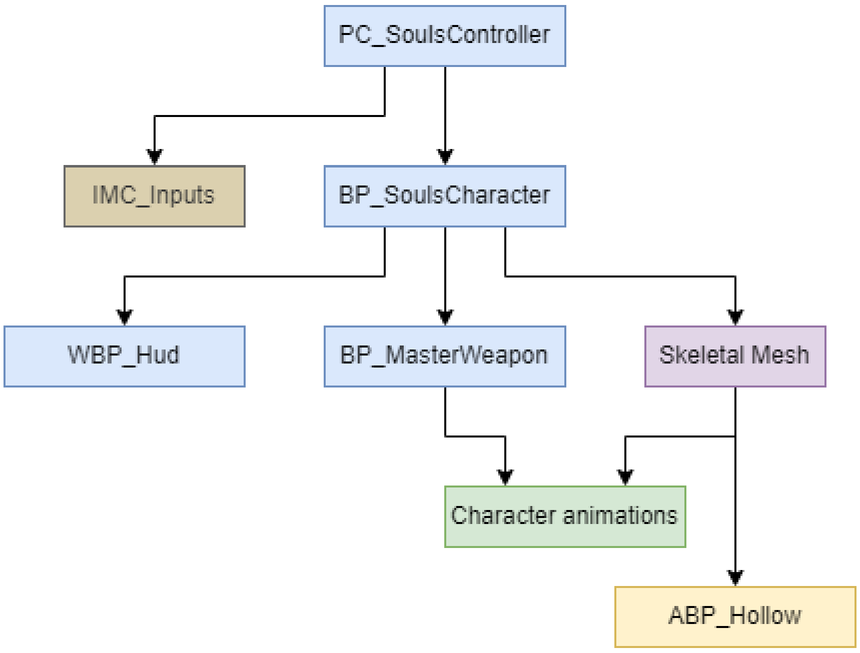
Macros

- Reduce stamina	[reduces stamina and performs all stamina system code.]
- Set buffer	[performs all the code to set and cancel a buffer.]

3. System structure

To give some insight in to how to use the project, I made a little system structure diagram. This follows the usual structure that a character would go for except for the fact that the animations are variables within the weapon. Which means that if the weapons changes the animations change.

This structure also means that it is possible to exchange the skeletal mesh into a different one if you just use different animations and apply all the animation state notifies.

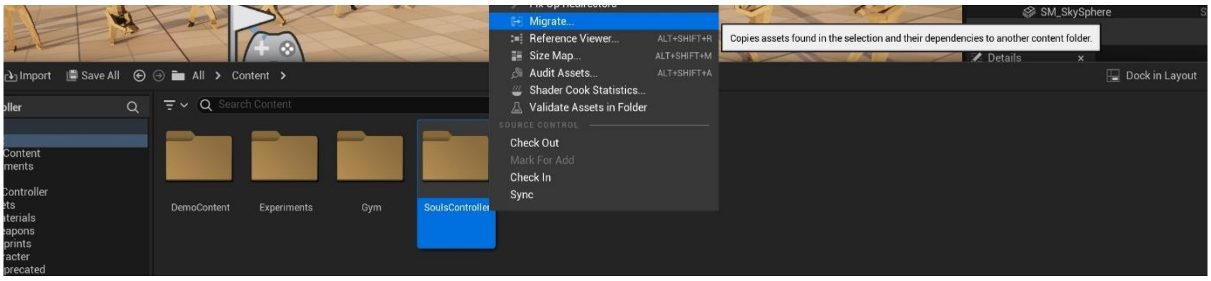


4. Importing into a new project (4 Steps)

Importing the project onto another project is very easy. I will go through the process in steps.

Step 1

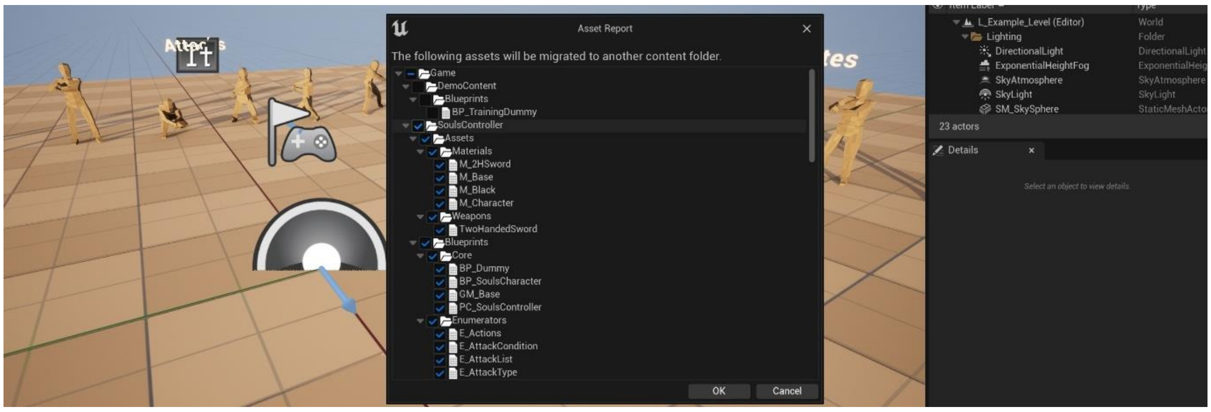
At first you will have to open up both the project you want migrate the character to and the project you want to migrate it to. Then, within the SoulsController project, go into the */Content/* folder and right click on */Content/SoulsController/* folder and select migrate.



Copies assets found in the selection and their dependencies to another content folder.

Step 2

When the migration menu pops up, deselect the game folder and select ONLY the */Soulscontroller/* folder. If it looks the same as the picture, click “OK”.

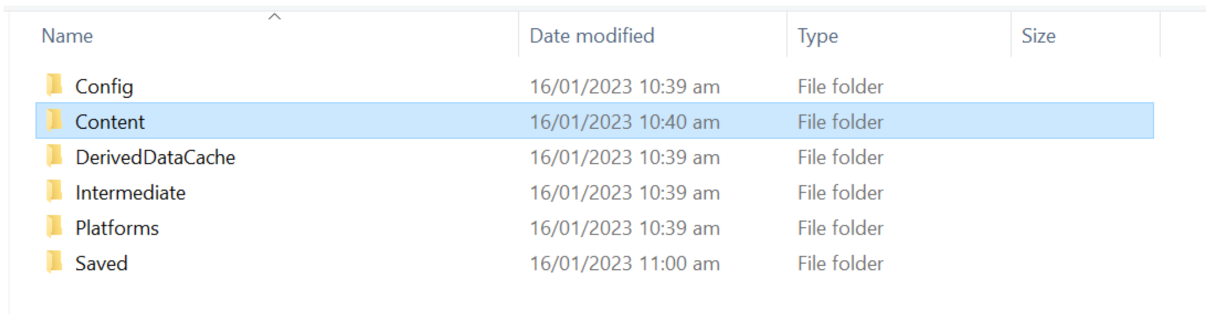


```
Asset Report
The following assets will be migrated to another content folder.
Game
  DemoContent
    Blueprints
      BP_TrainingDummy
  SoulsController
    Assets
      Materials
        M_2HSword
        M_Base
        M_Black
        M_Character
      Weapons
        TwoHandedSword
    Blueprints
      Core
        BP_Dummy
        BP_SoulsCharacter
        GM_Base
        PC_SoulsController
      Enumerators
        E_Actions
        E_AttackCondition
        E_AttackList
        E_AttackType
OK    Cancel
```

```
L_Example_Level (Editor)    World
  Lighting                  Folder
    DirectionalLight        DirectionalLight
    ExponentialHeightFog    ExponentialHeig
    SkyAtmosphere           SkyAtmosphere
    SkyLight                SkyLight
    SM_SkySphere            StaticMeshActo
23 actors
Details
Select an object to view details.
```

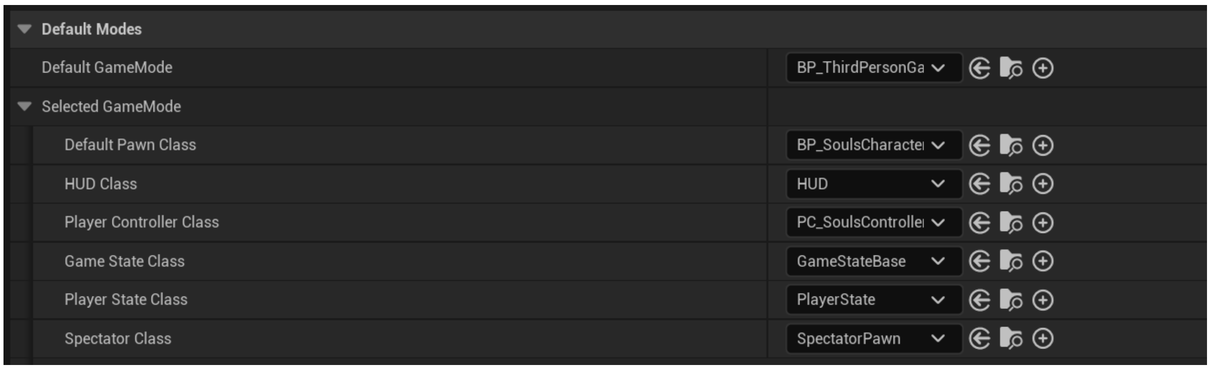
If you want to migrate any of the demo / gym related content, feel free to!

Step 3
It will open up the content browser. Here you should locate the project that you want to import it to. When you've located it select content and press "select folder".



Name	Date modified	Type	Size
Config	16/01/2023 10:39 am	File folder	
Content	16/01/2023 10:40 am	File folder	
DerivedDataCache	16/01/2023 10:39 am	File folder	
Intermediate	16/01/2023 10:39 am	File folder	
Platforms	16/01/2023 10:39 am	File folder	
Saved	16/01/2023 11:00 am	File folder	

Step 4
It should now be imported within the project. Now to use it as your character, you have to select it within the gamemode. You can go to your project settings > Maps & modes and select both the PC_SoulsController as your selected player controller and BP_SoulsCharacter as your selected pawn. It should look like this.



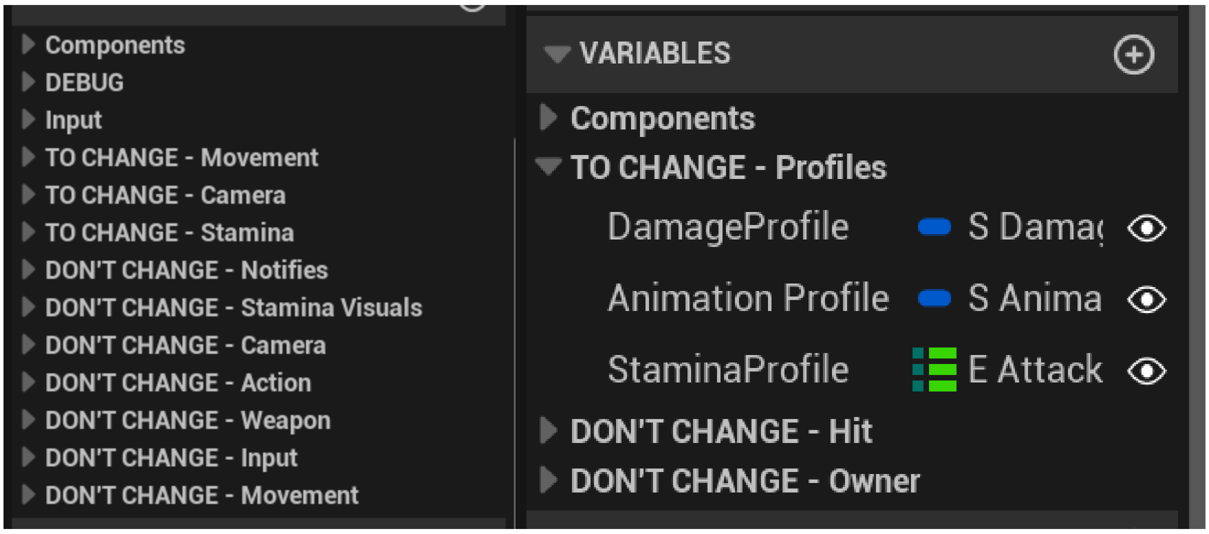
Default Modes	
Default GameMode	BP_ThirdPersonGa
Selected GameMode	
Default Pawn Class	BP_SoulsCharacte
HUD Class	HUD
Player Controller Class	PC_SoulsControlle
Game State Class	GameStateBase
Player State Class	PlayerState
Spectator Class	SpectatorPawn

Congratulations! You've now correctly imported the souls character unto your project!

5. Creating Content

5.1 Adjustable variables

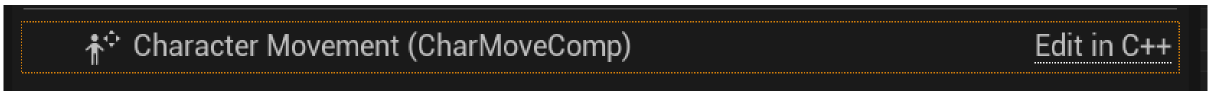
The variables are set up in way that it is easy to identify what variables are able / allowed to be changed due to the category structure that I've applied. In the character, it looks like this.



[These pictures show the categories of the variables that can be changed in the character (left) and the weapon (right).]

Damage, damage type, stamina, hitbox radius and combat animations are stored with profiles, within the weapon blueprint.

It is also important to know that if you want to change the movement in detail, it is important to do this within the character movement component.

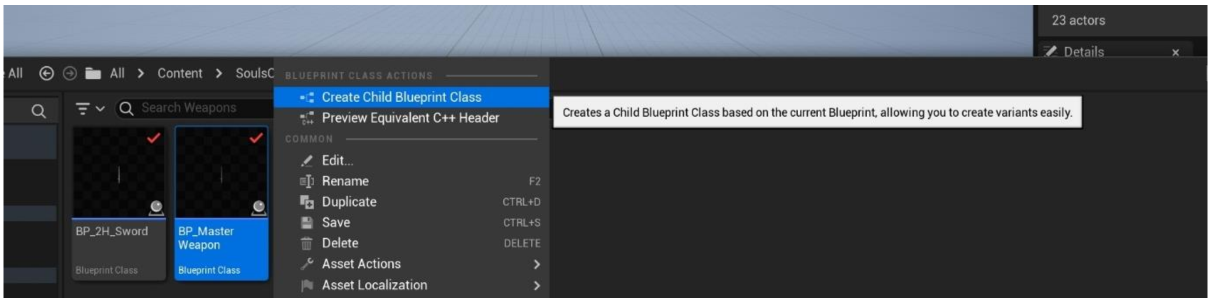


5.2 Adding new weapons (4 steps)

Adding a new weapon blueprint is fairly easy. Currently there are only animations for using two-handed swords. If you want to use any other weapon than the standard two-handed sword, you will have to supply both the static mesh and the animations for the weapons yourself.

Step 1
First thing you do, you create a child of the BP_MasterWeapon blueprint located here:

- */SoulsController/Content/SoulsController/Blueprints/Weapons/BP_MasterWeapon.uasset*

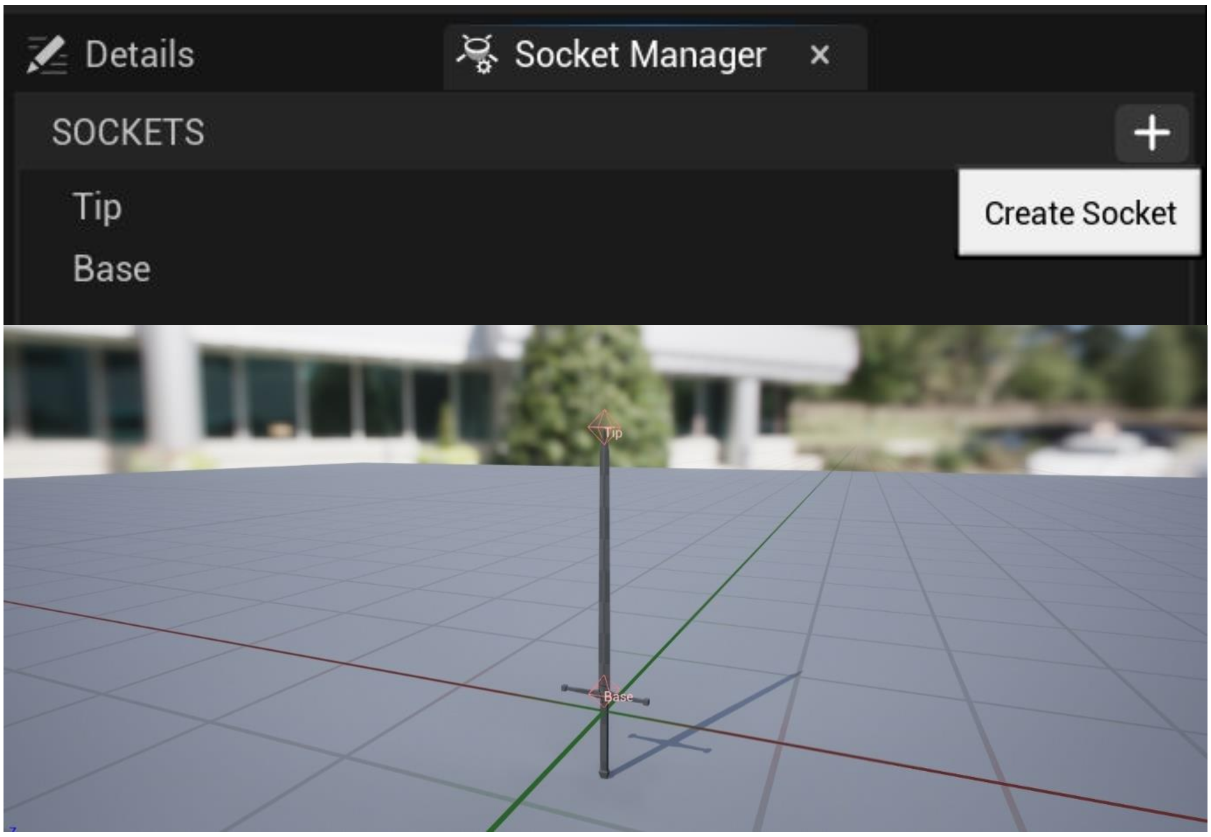


Creates a Child Blueprint Class based on the current Blueprint, allowing you to create variants easily.

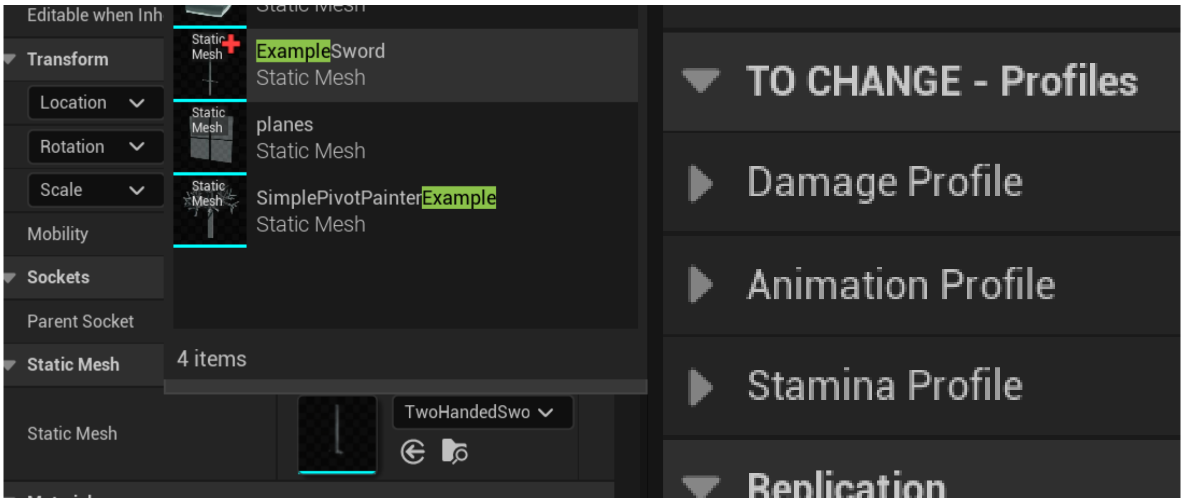
Step 2
You will have to supply your own weapon model, and before you can use it, you need to create two sockets. These are called “Base” and “Tip”. It should look a little bit like the next image.

Tip – If you don't have your own weapon you can use a copy of the base two-handed sword. It is located here:

- */SoulsController/Assets/Weapons/TwoHandedSword.uasset*

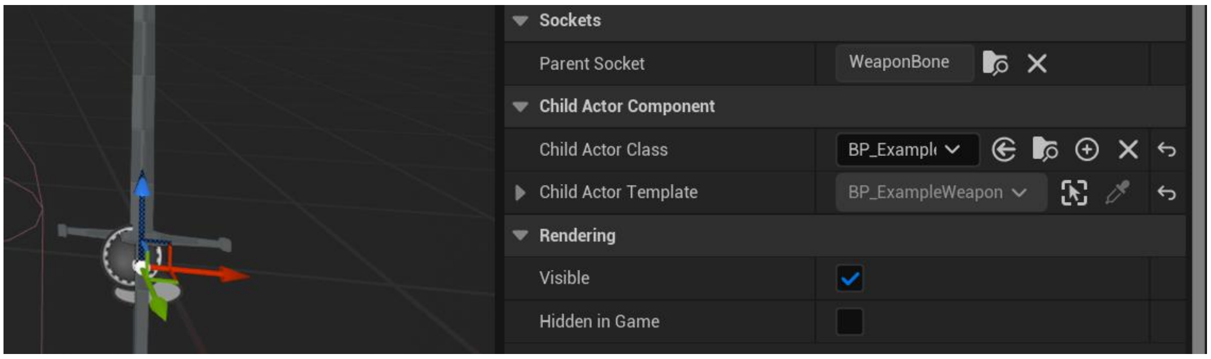


Step 3
Now just replace the static mesh "Weapon Mesh" to your desired model and change the variables to fit your weapon. If you have a new animation set, don't forget to adjust the animation profile inside of the **class defaults** to use your new animations.



[Static mesh of weapon (left), variables to change (right)]

Step 4
To "equip" the character to use this weapon, go into the character blueprint and select the Weapon_Actor component, then go to the right side of the screen and select your own weapon class as the Child Actor Class object.


Sockets

Parent Socket: WeaponBone

Child Actor Component

Child Actor Class: BP_Exampl

Child Actor Template: BP_ExampleWeapon

Rendering

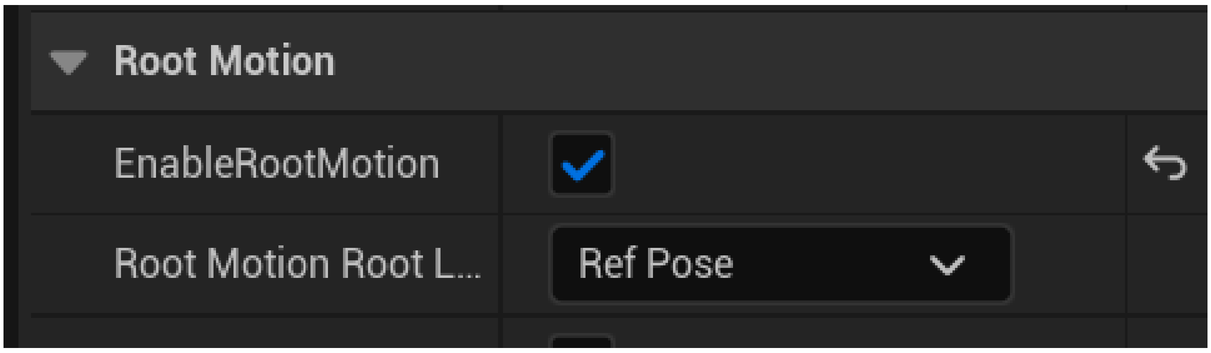
Visible: ✓

Hidden in Game:

Done! Now you can go and play with your new weapon in the game!

5.3 Setting up new animations (5 steps)
I won't be giving a tutorial on how to animate with the rig, but I will show you how you can set up new animations with the animation notify state system.

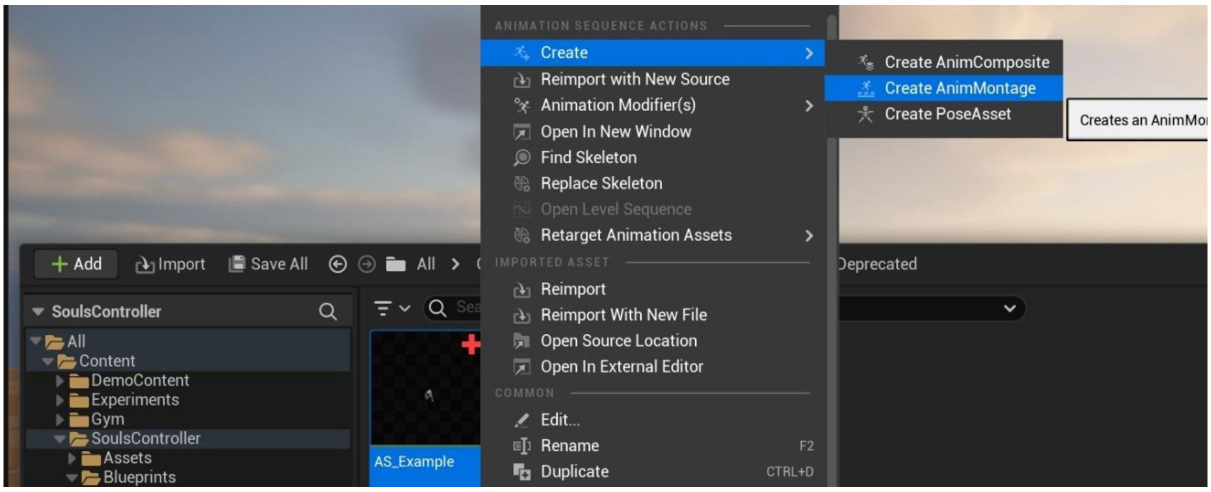
Step 1
First thing you do, is set EnableRootMotion to true inside of the animation sequence.


Root Motion

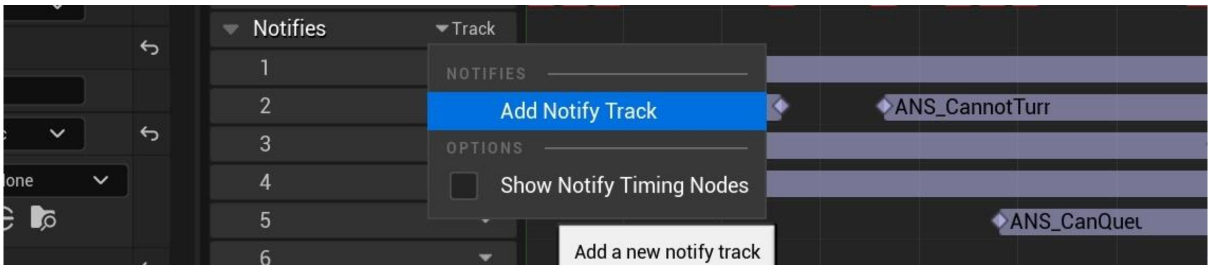
EnableRootMotion: ✓

Root Motion Root L...: Ref Pose

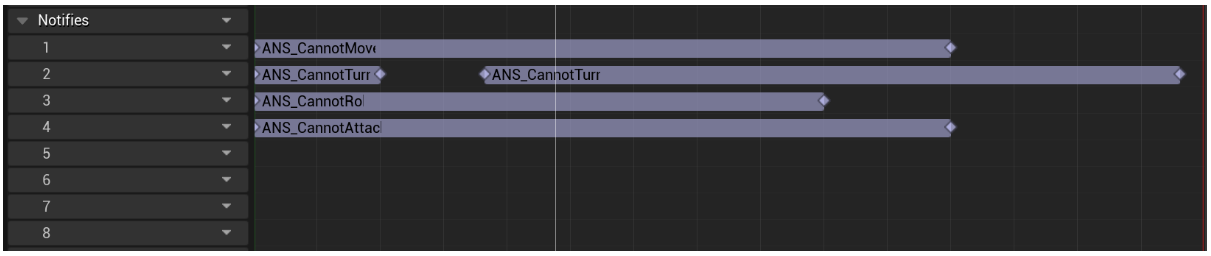
Step 2
Secondly, you create a montage of said animation sequence.



Step 3
Open up the animation montage and add 8 notify tracks to hold animation notify states.



Step 4
Then, I would start with adding the **main animation notify states** first. These are explained within section 2.1. These are general and should fit within any action that the player is able to perform. It will look a bit like this.



You should adjust these states depending on what purpose the animation has. If it’s a forgiving attack, you should allow the player to roll quite early.

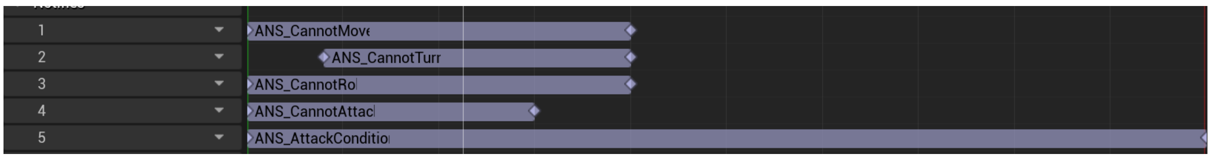
Note – Don’t forget to turn off “set rotation desired” to **false** in the second CannotTurn notify state. Otherwise, it will rotate towards desired rotation randomly at the start of the second notify.

Tip – Right click on an empty space in the notify track to enable snapping to frames!

Step 5
Depending on the attack, you will have to put in different animation notify states. to make a certain kind of animation I will give a specific explanation.

You now have to choose between these three different actions:

Conditional action
This is an action which could have a specific type of attack after it has been completed. Take either the backstep or roll as an example. To create a conditional attack, you will have to add the ANS_AttackCondition to a notify track. This track has to span over the entire animation.

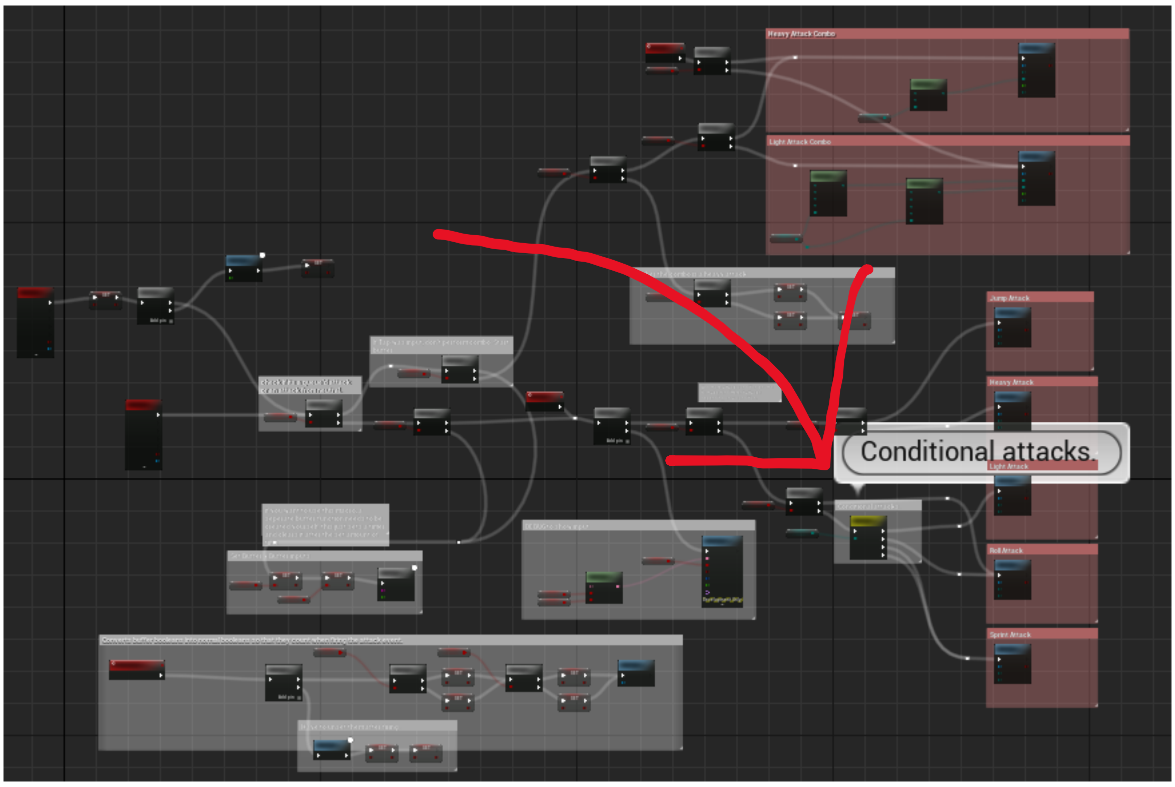


1	ANS_CannotMove
2	ANS_CannotTurr
3	ANS_CannotRo
4	ANS_CannotAttacl
5	ANS_AttackConditio

Now select the attack condition notify and select the conditional state you want this animation to have!

If you want to add a new conditional state, add this In the E_Attackcondition enumerator here:
- /SoulsController/Blueprints/Enumerators/E_AttackCondition.uasset

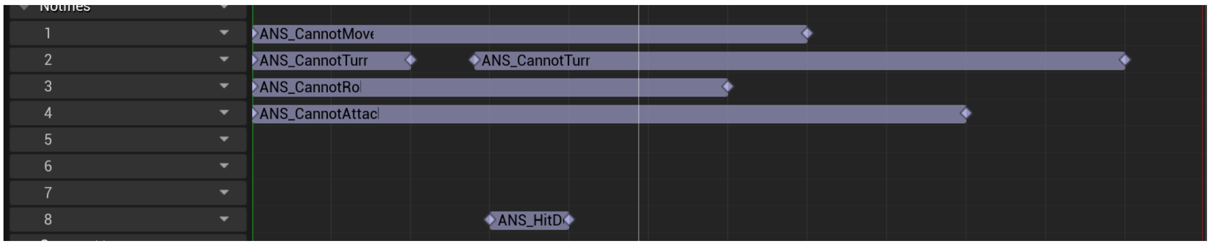
The place that you will have to create the conditional attack is inside of the character blueprint, within the Attack Graph.



Attack without combo

If you want to setup an attack that doesn't make use of combos, you basically only need to add the ANS_HitDetection state to the part of the animation you want to detect hits on.

Do note that it uses some instance editable variables to select what damage it will do and what knockback strength it has.



If you want to add new types of damage you will have to add an enumerator on the E_AttackType enumerator. You will also have to edit the damage on the damage profile of the weapon *don't forget*. The enumerator is located here:

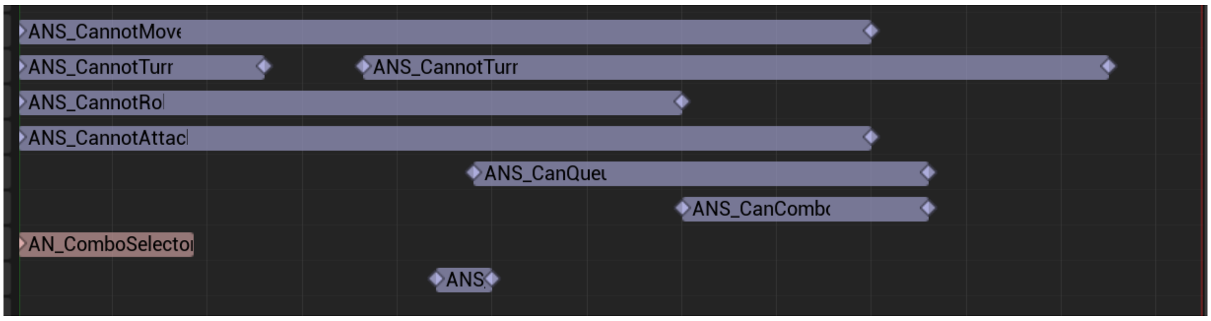
- /SoulsController/Blueprints/Enumerators/E_AttackType.uasset

Attack with combo

To setup an attack with a combo you will have to put a lot of animation notifies into the animation. You will have to add these 4 on top of the main 4:

- ANS_ComboSelector	[Has instance editable variables]
- ANS_HitDetection	[Has instance editable variables]
- ANS_CanQueue	
- ANS_CanCombo	

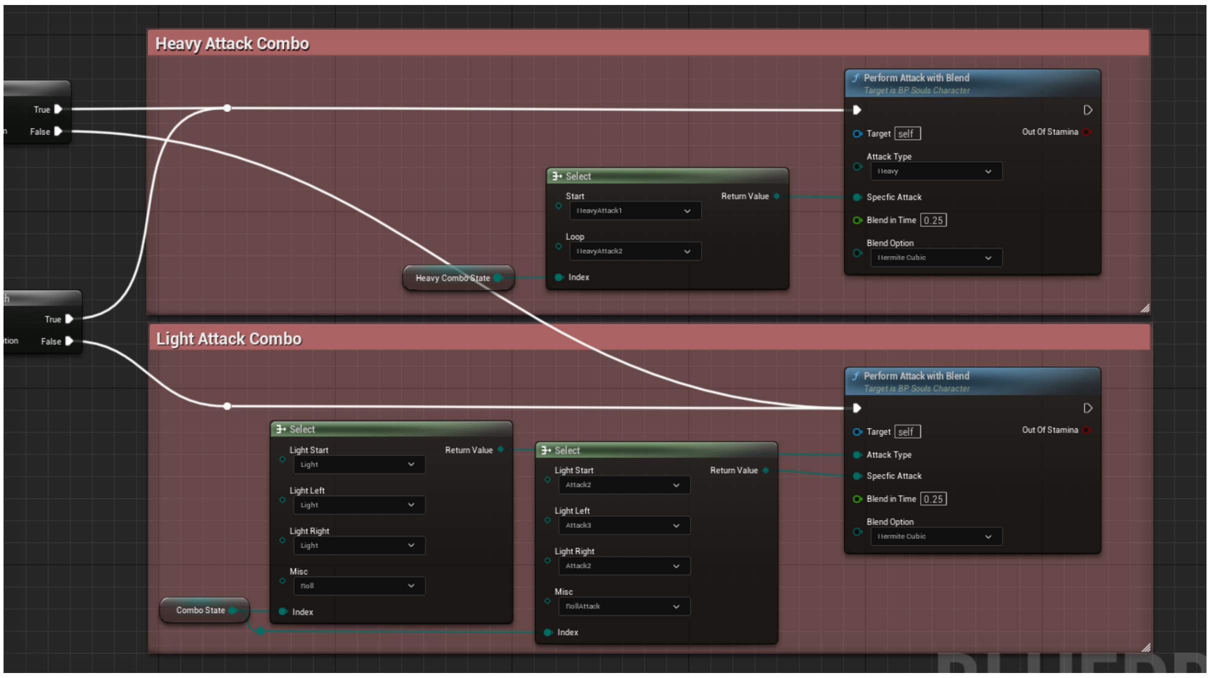
If you want another explanation, I recommend checking out section 2.1.



When setting this up, don't forget to set up the correct combo state on the combo selector notify. If you want to have the possibility of more enumerators and combos, you can do so by adding an enumerator here:

- /SoulsController/Blueprints/Enumerators/E_LightCombo.uasset
- /SoulsController/Blueprints/Enumerators/E_HeavyCombo.uasset

If you add a new enumerator here, you will also have to delve into the code to make sure these select nodes are properly set up.



Note – Adding a new enumerator here is going to have an effect on a lot of the other systems and enumerators and I recommend reading section 5.4

5.4 Creating a new action or attack type (3-5 steps)

I'm not going to explain where to start the montage in code, but I do think it is important to understand the underlying systems regarding actions and attacks.

Currently, there are 7 different places where you might have to add stuff. These are:

S_AnimationProfile: /SoulsController/Blueprints/Structures/S_AnimationProfile.uasset
S_DamageProfile: /SoulsController/Blueprints/Structures/S_DamageProfile.uasset
E_AttackList: /SoulsController/Blueprints/Enumerators/E_AttackList.uasset
E_AttackType: /SoulsController/Blueprints/Enumerators/E_AttackType.uasset
E_CharacterStaminaCosts: /SoulsController/Blueprints/Enumerators/E_CharacterStaminaCosts.uasset
StaminaProfile: [A map variable within the weapon blueprint.]
CharacterStaminaProfile: [A map variable within the character blueprint.]

Creating an action

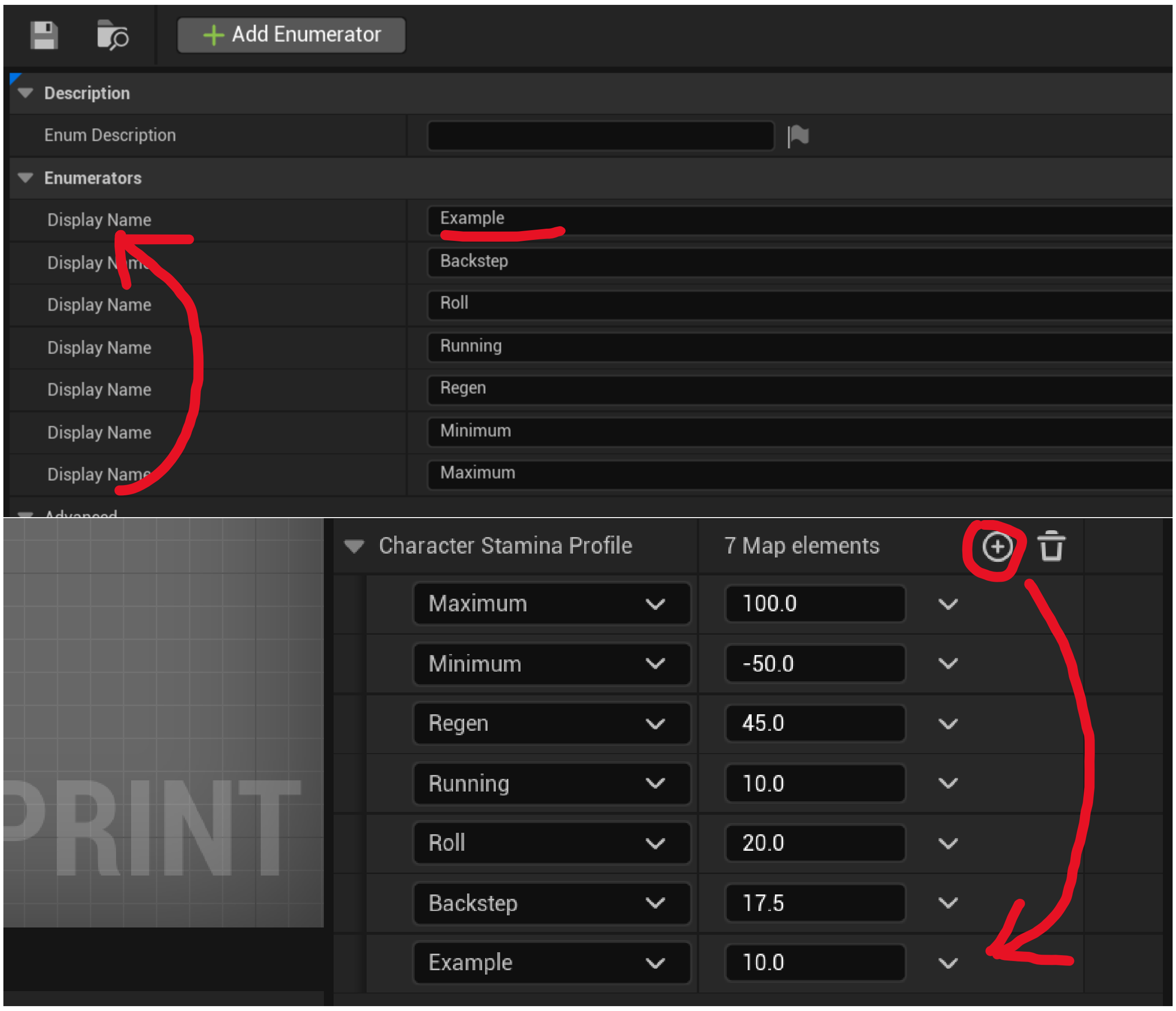
Step 1

A new action (or attack) has to be added to S_AnimationProfile. After which it is possible to pull the animation off of AnimationProfile variable within the character. You have to add a **Anim Montage Object reference**.

Step 2

Add your action's stamina Cost in E_CharacterStaminaCosts and a new mapped element in the CharacterStaminaProfile in the character blueprint. It should look like this:

Note – Put the new enumerators at the top so that when adding them as a mapped element, they won't be blocked due to the default value already being in the map.



Display Name	Example
Display Name	Backstep
Display Name	Roll
Display Name	Running
Display Name	Regen
Display Name	Minimum
Display Name	Maximum

Maximum	100.0
Minimum	-50.0
Regen	45.0
Running	10.0
Roll	20.0
Backstep	17.5
Example	10.0

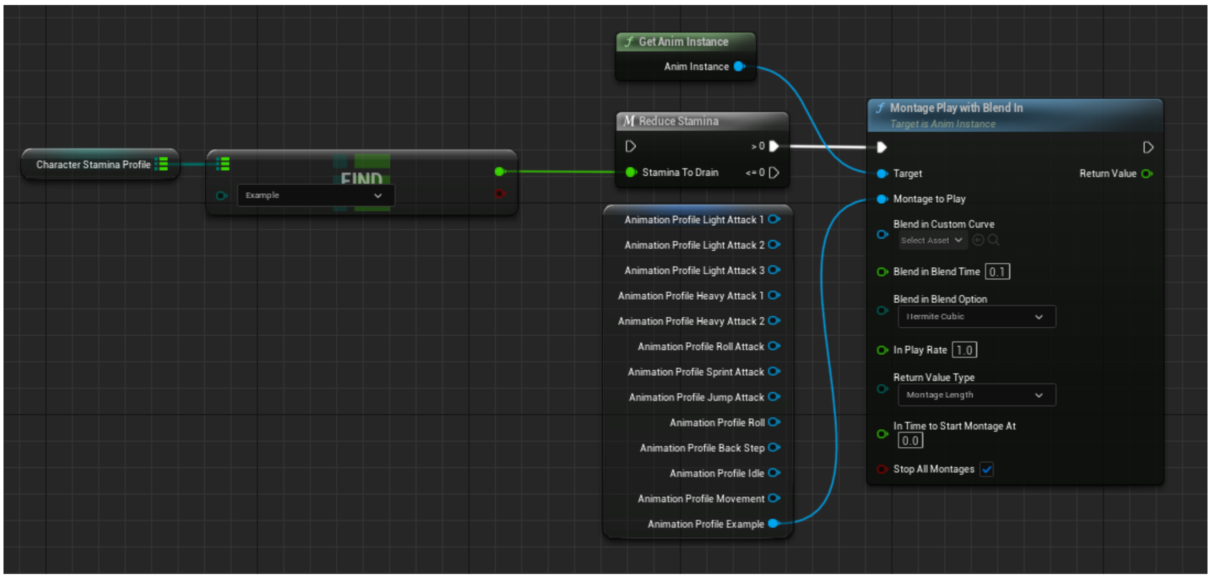
Step 3
You have to add your animation to the animation profile within the weapon you plan to use. This **S_AnimationProfile** is located within the weapon. The master weapon is located here:

- /SoulsController/Blueprints/Weapons/BP_MasterWeapon.uasset

Step 4
Now you have to set up the necessary code to perform the action within the character. An action would require a Reduce Stamina node, Montage play (With blend in, if desired) which pulls its animation from **AnimationProfile** within the character. To get the stamina value for your action, you have to get the **CharacterStaminaProfile** map variable and find the specific value with your action enumerator.

Note - You have to split the struct to be able to pull the specific animation from the AnimationProfile variable.

The action play event should look a little bit like this:


Get Anim Instance
Anim Instance
Reduce Stamina
Stamina To Drain
Character Stamina Profile
FIND
Example
Animation Profile Light Attack 1
Animation Profile Light Attack 2
Animation Profile Light Attack 3
Animation Profile Heavy Attack 1
Animation Profile Heavy Attack 2
Animation Profile Roll Attack
Animation Profile Sprint Attack
Animation Profile Jump Attack
Animation Profile Roll
Animation Profile Back Step
Animation Profile Idle
Animation Profile Movement
Animation Profile Example
Montage Play with Blend In
Target is Anim Instance
Target
Return Value
Montage to Play
Blend in Custom Curve
Select Asset
Blend in Blend Time 0.1
Blend in Blend Option
Hermite Cubic
In Play Rate 1.0
Return Value Type
Montage Length
In Time to Start Montage At 0.0
Stop All Montages

Note – Don’t forget to recompile all of the blueprints that make use of the character stamina profile. Reload the widget blueprint!!!

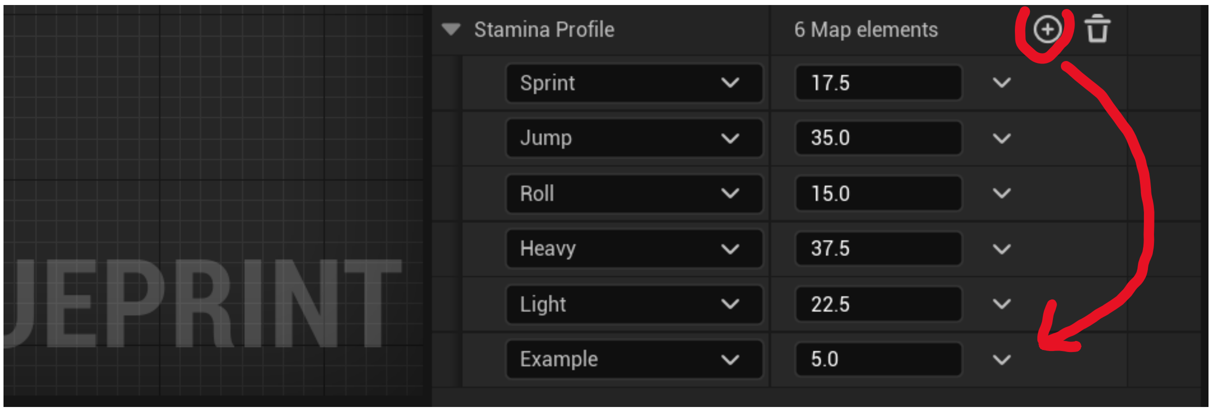
Creating an attack type

Step 1

A new action (or attack) has to be added to S_AnimationProfile and the E_AttackList. After which it is possible to pull the animation off of AnimationProfile variable within the character. You have to add a **Anim Montage Object reference**.

Step 2 (Only if attack is new type of attack)

Add your attack to the E_AttackType Enumerator, Add your attack to S_DamageProfile and the StaminaProfile as a mapped element.

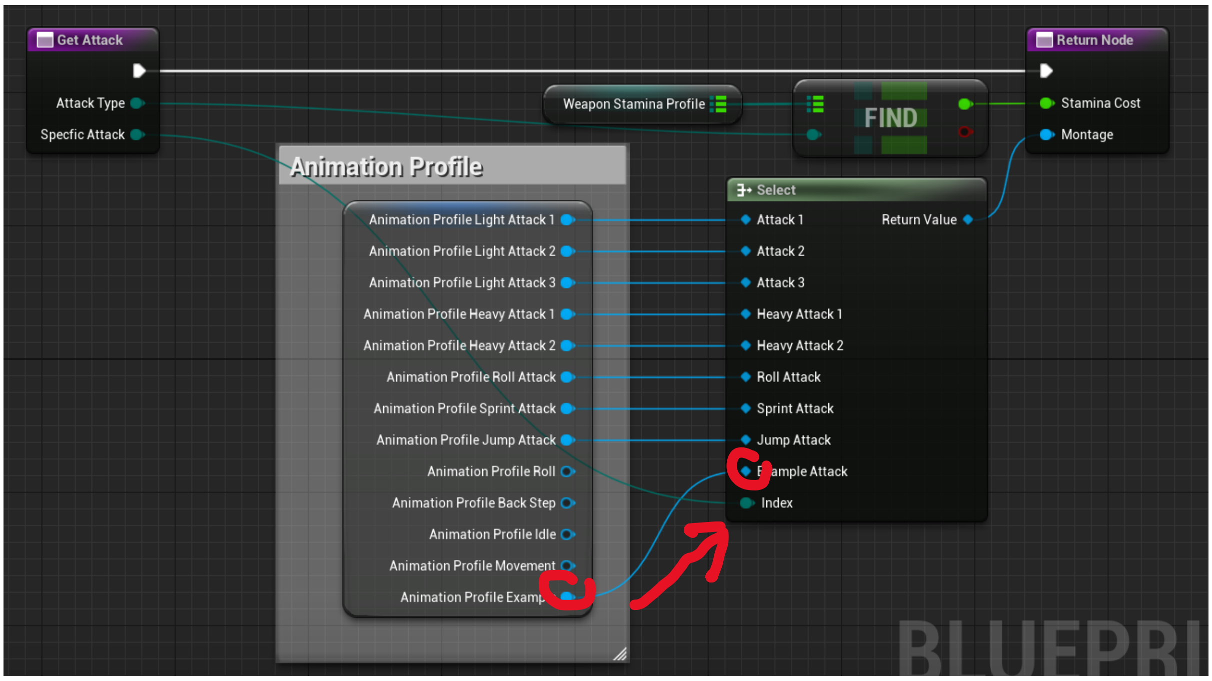


Stamina Profile	6 Map elements
Sprint	17.5
Jump	35.0
Roll	15.0
Heavy	37.5
Light	22.5
Example	5.0

Note – Put the new enumerators at the top so that when adding them as a mapped element, they won't be blocked due to the default value already being in the map.

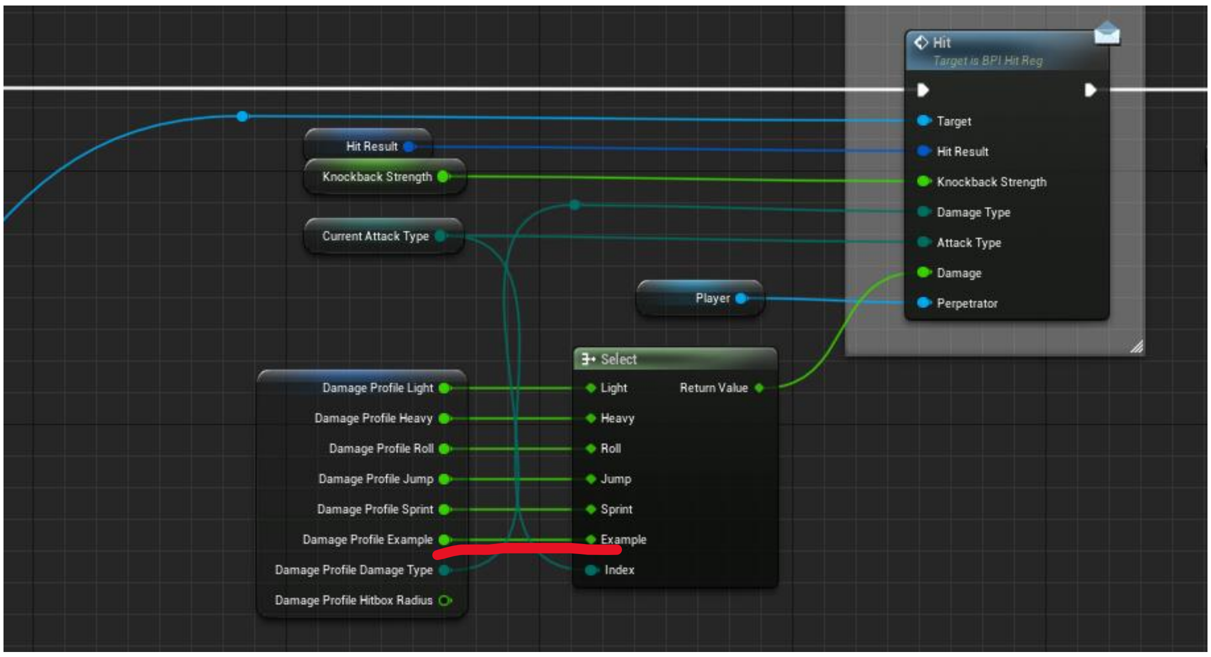
Step 3

In the BP_SoulsCharacter there is a function called GetAttack, connect your new output from the AnimationProfile struct to the right index on the SpecificAttack index. It should look like this example:

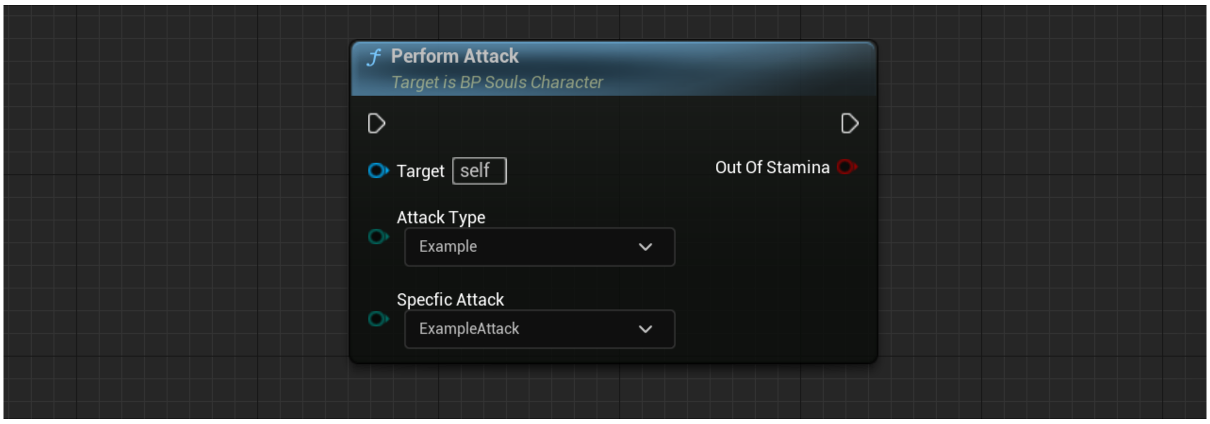


Note – If your attack doesn't immediately appear, reload the blueprint. The change might've not loaded in properly yet.

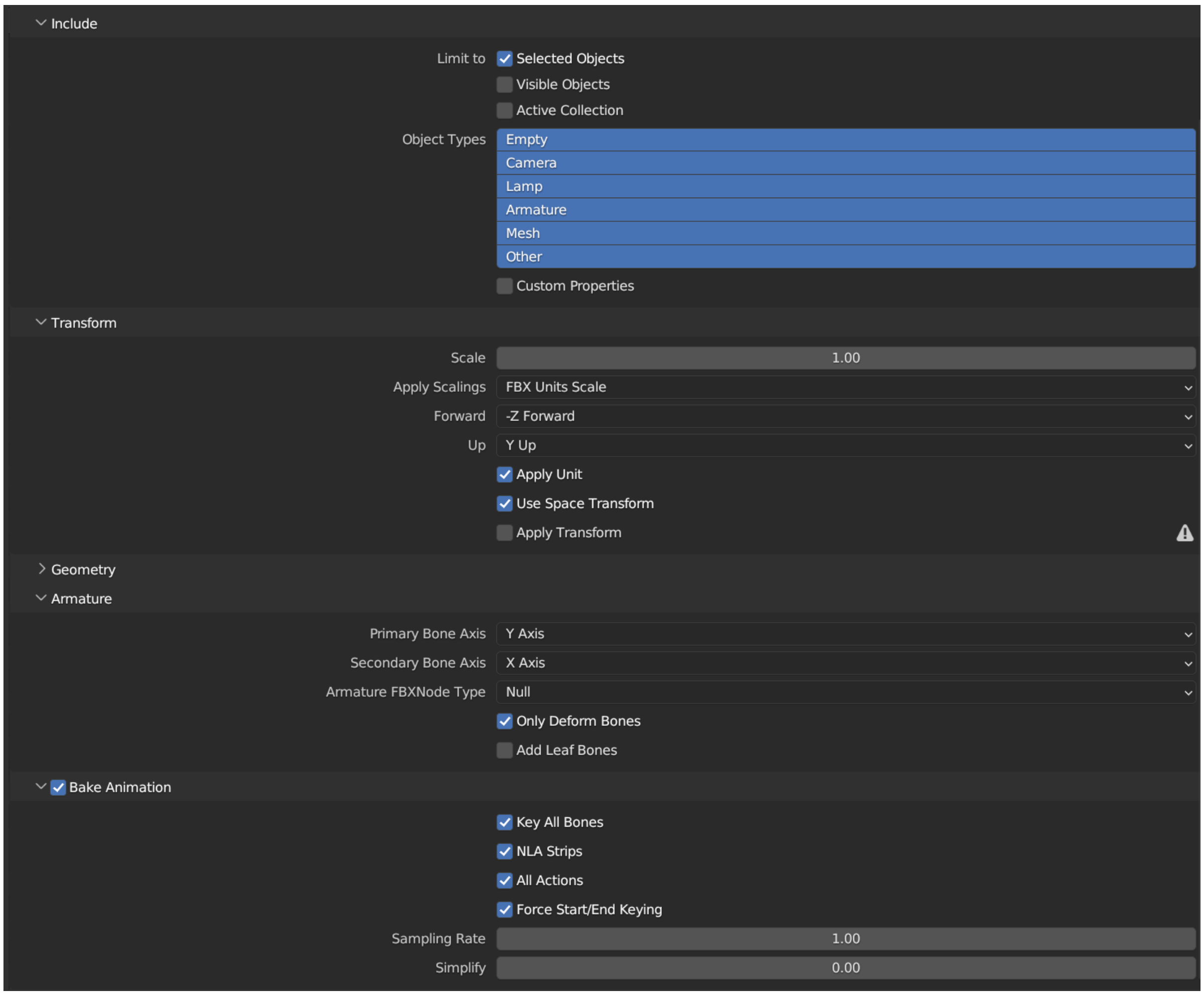
Step 4
In the BP_MasterWeapon there is function called Trace, inside of this function you should connect your attack DamageProfile value to the correct CurrentAttackType index. It should look a bit like this:



Step 5
It is now possible to use the PerformAttack or PerformAttackWithBlend node to select your attack and play it at a desired time! Hurray!



5.5 Blender export settings
If you are going to be creating your own animations with the blender model created, you will need my export settings. Here they are:


Include

Limit to: [✓] Selected Objects, [ ] Visible Objects, [ ] Active Collection

Object Types: Empty, Camera, Lamp, Armature, Mesh, Other

[ ] Custom Properties

Transform

Scale: 1.00

Apply Scalings: FBX Units Scale

Forward: -Z Forward

Up: Y Up

[✓] Apply Unit

[✓] Use Space Transform

[ ] Apply Transform

Geometry

Armature

Primary Bone Axis: Y Axis

Secondary Bone Axis: X Axis

Armature FBXNode Type: Null

[✓] Only Deform Bones

[ ] Add Leaf Bones

[✓] Bake Animation

[✓] Key All Bones

[✓] NLA Strips

[✓] All Actions

[✓] Force Start/End Keying

Sampling Rate: 1.00

Simplify: 0.00

Important things to note are: **Only Deform Bones, Simplify = 0.00, Apply scalings = FBX Units Scale.**

Goodluck animating!

6. Known issues

When I encounter issues / bugs I will report on them here. This will hopefully get reiterated upon further in development.

Montage play delayed anim state begin

This bug has occurred a couple times where a montage play node gets fired, but the animation state notifies begin events don't get fired within the same tick of playing the montage. This means that the animation gets canceled due to the montage stop node on move input (because the ANS_CannotMove doesn't play, this montage stop is allowed to execute)

I'm suspecting this is an engine bug, but it requires some further research and testing to clearly know the cause. A bug report will be made in the future.

Many Enumerators

The big reason why it might be harder for beginners to use this feature is due to a large amount of enumerators being used in many places. Further iteration onto the project will try and solve this issue by replacing the enumerators with a better system.