Unreal Engine 5 - Lesson 5 - Animations

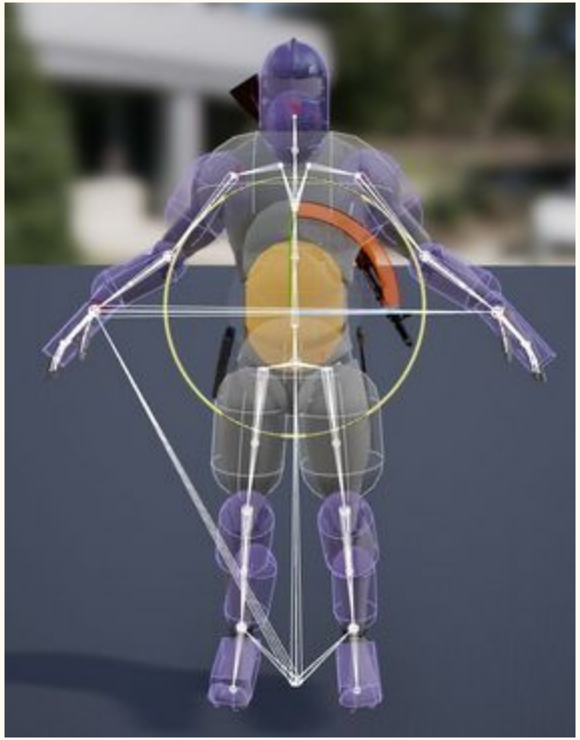
Nicolas Serf - Gameplay Programmer - Wolcen Studio
serf.nicolas@gmail.com

Summary

- Skeletal Mesh Animation System
- Animation Sequence
- Animation Montage
  - Timeline
  - Playing
- Slots
- Animation Editor
  - Skeleton Editor
    - Sockets
    - Notify
  - Skeletal Mesh Editor
  - Animation Sequence Editor
  - Animation Blueprint Editor
- State Machines
  - States
  - Transitions
- Blend Spaces
- Root Motion
- Much more...

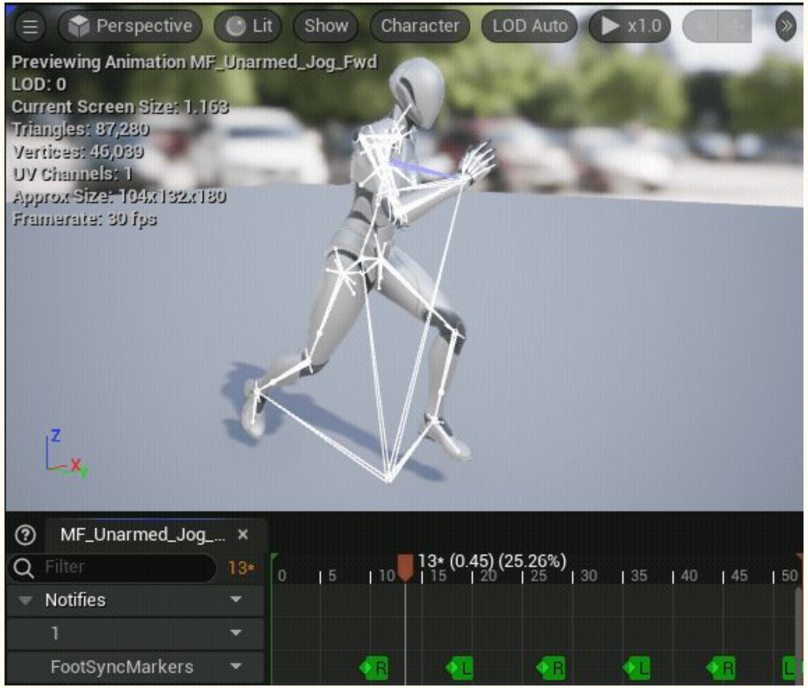
Skeletal Mesh Animation System

- Let's start by **Skeletal Mesh Animation System**
- It is the **most common standard** way of animation in Unreal when you are in a **3D environment** project with a **Skeletal Mesh** as your working ground
- A **Skeletal Mesh** is a **rigged mesh** that you can **manipulate** in order to create animations
- In addition, **Animation Blueprints** can be augmented to **Skeletal Meshes** to apply **logic** that **governs animation behavior** and **interactions** within levels.
- There is **multiple animation tools** provided by Unreal in order to works with Skeletal Meshes, we'll try to **highlight** mains one in this lesson
- Keep in mind that, again, it is more like an **introduction** about animation system in the project
  - You need to **dive more precisely** into it if you want to get a better **grasp** at the system
  - For some of you, in **small teams**, you may be **responsible** for **creating** and **maintaining** Animation Blueprint and animation system
  - For others that will be working in **bigger team / company**, it is most likely a **job** that will be handled by a **technical animator**



Animation Sequence

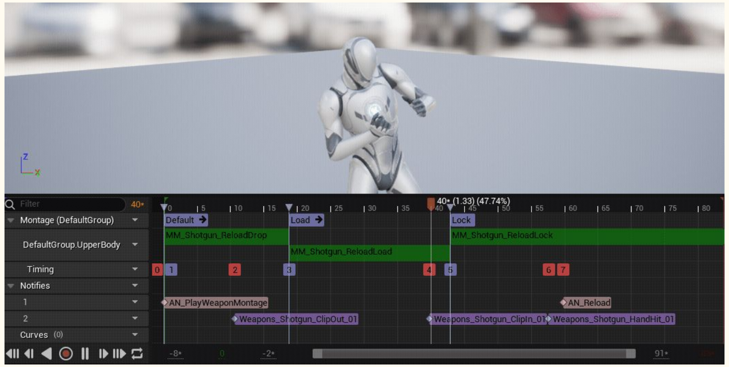
- Before diving into **multiples editor** windows and **system** that make the **animation system** what it is, let's start by the very basic which is **Animation Sequence**
- An **Animation Sequence** is an **animation asset** that contains **animation data** that can be played on a **Skeletal Mesh** to animate a character.
- An Animation Sequence contains **keyframes** that specify the **position**, **rotation**, and **scale** of the **Skeletal Mesh's Skeleton** at specific points in time. By **blending between keyframes** during sequential playback, the Skeleton's motion **animates** the Mesh.
- Animation Sequences are tied to **specific Skeletons**, which enables **animations** to be **shared across Skeletal Meshes** that use the same Skeleton.
- **Animation Sequences** are most often **created** in **external animation** and modeling software and are contained within an **FBX file**. You can **import Animation Sequences** into Unreal Engine during the **FBX import process** for use in your project.
- For each **Skeletal Mesh**, you will then be able to **select** the **skeleton** you would **like to use**, and it is **important** as the **animation** will be played on that **skeleton**
- There is still a lot about **Animation Sequence** like **compression, animation sharing**, etc... You can find more details here



```
Previewing Animation MF_Unarmed_Jog_Fwd
LOD: 0
Current Screen Size: 1.163
Triangles: 87,280
Vertices: 46,039
UV Channels: 1
Approx Size: 104x132x180
Framerate: 30 fps
```

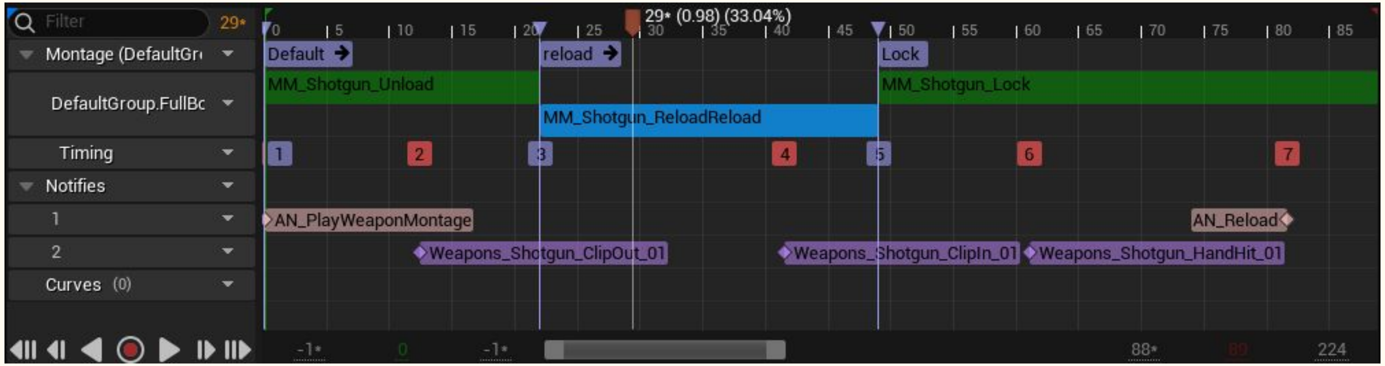
Animation Montage

- **Animation Montages** are like a extended feature-wise **Animation Sequence**
- You use **Animation Montages** when you want to **combine multiples Animation Sequence** into a **single asset** and control it through blueprint
- It is also used to **replicate Root Motion** in a **network environment**
- Coming from that principle of combining multiples Animation Sequence, you can also uses the notion of **Montage Sections**
  - It allows to **dynamically played back** in **any order** through logic at runtime by **section name**
  - You can control the **transitions between Montage Sections** in the **Montage Sections Panel** or you can set up more **dynamic transition** behaviors between Sections by using **Blueprints**.
- **Montages** being **assets**, you can create then by **right clicking** in content drawer and search for **"Animation Montage"**, you'll then be asked which **skeleton** you want to animate through this animation montage
- Alternatively, you can also create a **montage** by right clicking on an **Animation Sequence** which will **automatically populate** the montage timeline



Animation Montage - Timeline

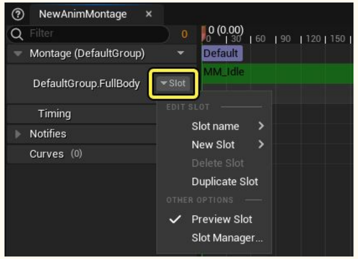
- The **Timeline**, like other editor timelines in Unreal Engine, is organized into **tracks**. These tracks contain **information** that **dictates** the behavior of the **animation playback** and are organized into **playback frames**.
- First entry on the timeline are **sections**
  - Indicated by their assigned name and a purple header, you can add, delete and move these sections to match your needs
  - To create a Montage Section **right click** within the top track, select **Create New Section**, and enter a name for this Montage Section
- Then, there is **montage tracks**
  - You can **add** and **manage** Animation Sequence **tracks**, organize tracks by **Slot Groups** or **Slots**, and see **Timing indicators**.
  - It is **important** to understand that timing we are seeing are **pure indicator**, you'll moves **notifies or sections** in order to move those **indicators**

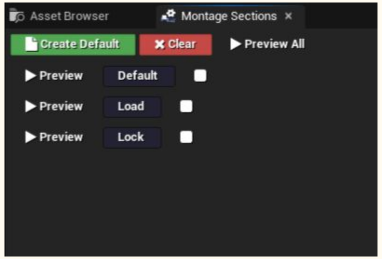


Animation Montage - Timeline

- On track, you have the notion of Slot
    - Within **Animation Montages**, Animation sequences are **organized** and played in **Slots**. A Slot **occupies** a **mesh** or **part of a mesh**, and when an animation is placed into the **Slot** from the animation Montage it can then be played on the Animation Blueprint in that Slot. A **Slot track** is a succession of **sequences** that can hold any number of **Animation Sequences**.
    - To play **Sequences** on **different parts of your Character**, animations can be divided into **Slot Groups**. These Slot Groups can be set up to play on **separate parts** of your character using the Anim Graph. For example you can use an **upper body Slot** and a **lower body slot**, to play separate animations on the upper body and lower body at the same time.

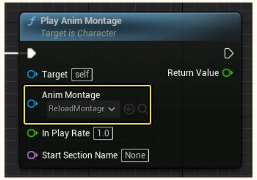
- On a **separated window**, but directly **linked** to the **timeline** you have the **montage sections**
    - You can **establish** the **default playback order** of your Montage Sections.
    - By **clicking** on **white boxes**, you can defines how the **playflow** of the animation is **designed**
- Finally, there is the **notifies** and the **curves**, we'll be speaking more about them later on the lesson as they are really **important** as a **prog** point of view



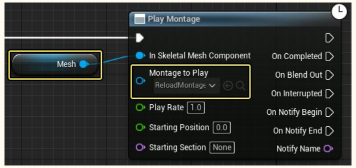


Animation Montage - Playing

- Finally, let's see how to play an **Animation Montage** from Blueprint
- After having **created** and **modified** your Animation Montage in the **Animation Sequence Editor**, you can now play the **Montage** from a Blueprint as long as your character **actor** is able to be **referenced**. You can use the **Play Anim Montage** node to playback an Animation Montage.



- Obviously, if you provides an **Anim Montage** which is not compatible with the **Skeleton** of the mesh, it'll not works
- Other options include changing the **In Play Rate** of the montage by a **factor** of the value assigned. For example, a value of **2** will play the montage back at **double speed**
- With this node you can also define which section to begin playback if you want to use a section other than Default as the starting section.
- For a more powerful and feature rich node, you can use the **Play Montage node**. With the Play Montage node you need to connect the Mesh you want to animate and assign the Montage to Play. It provides **various event** which may be **necessary**

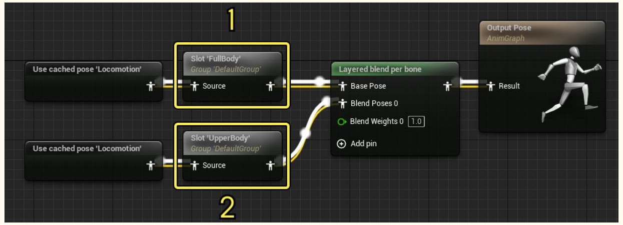


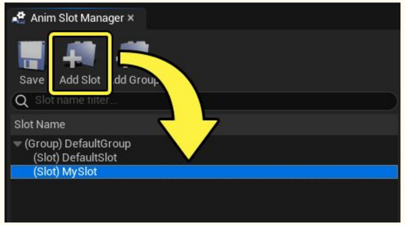
- As you can see, it is an **asynchronous node** and therefore, it can only be used in **Event Graph**
- In order for the **Montage to play**, you should set the **Animation** Mode to **Use Animation Blueprint** and connect the Animation Blueprint being used in the Anim Class property.

Slots

- **Slot concept** is **quite important** as it actually directly **drives where** and **how** the animation will be played
- We'll not go into too much **details** about it as it is most likely an **animator job**, but let's see how it works globally
- As you create more **complex Animation behavior** for your characters, it may be necessary to create **proxy areas** in your Animation Blueprint where you can insert **one-off animations**. This can be accomplished by using **Slots**, which are nodes that you can add to your Animation Blueprint at various points to layer and play animations into. **Slots** are mainly used together with **Animation Montages**, however they can also be used with **Sequencer**.
- **Slot** will be ultimately used in **Animation Blueprint**, and we'll explains that later on

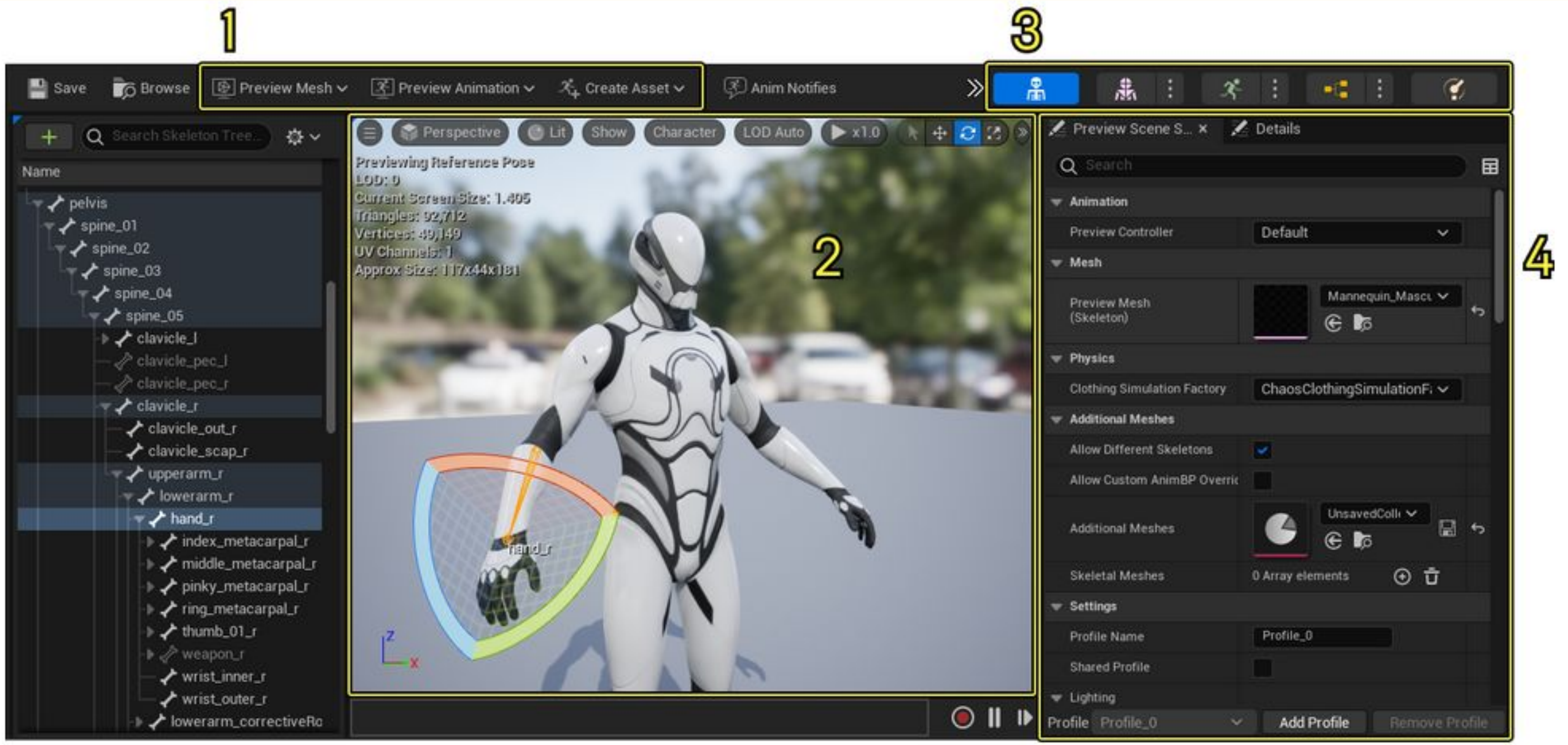
- You can **create** a slot from the **Anim Slot Manager window**
- By default, all **Skeletons** come with a **starting** Slot named **DefaultSlot**.
- You then have the slot group
  - Slot groups are **not** just **organizational**. When you play a montage that uses a **Slot** in the **same group** as one that's already **running**, it **stops** the **active** one. This **automatic behavior allows for montages to be interrupted**. For example, a weapon reloading montage could be interrupted to play an ability or melee attack montage. If both montages are playing animations on Slots within the same group, then the most recent montage will interrupt the previous.





(Group) DefaultGroup
(Slot) DefaultSlot
(Slot) MySlot

Animation Editor



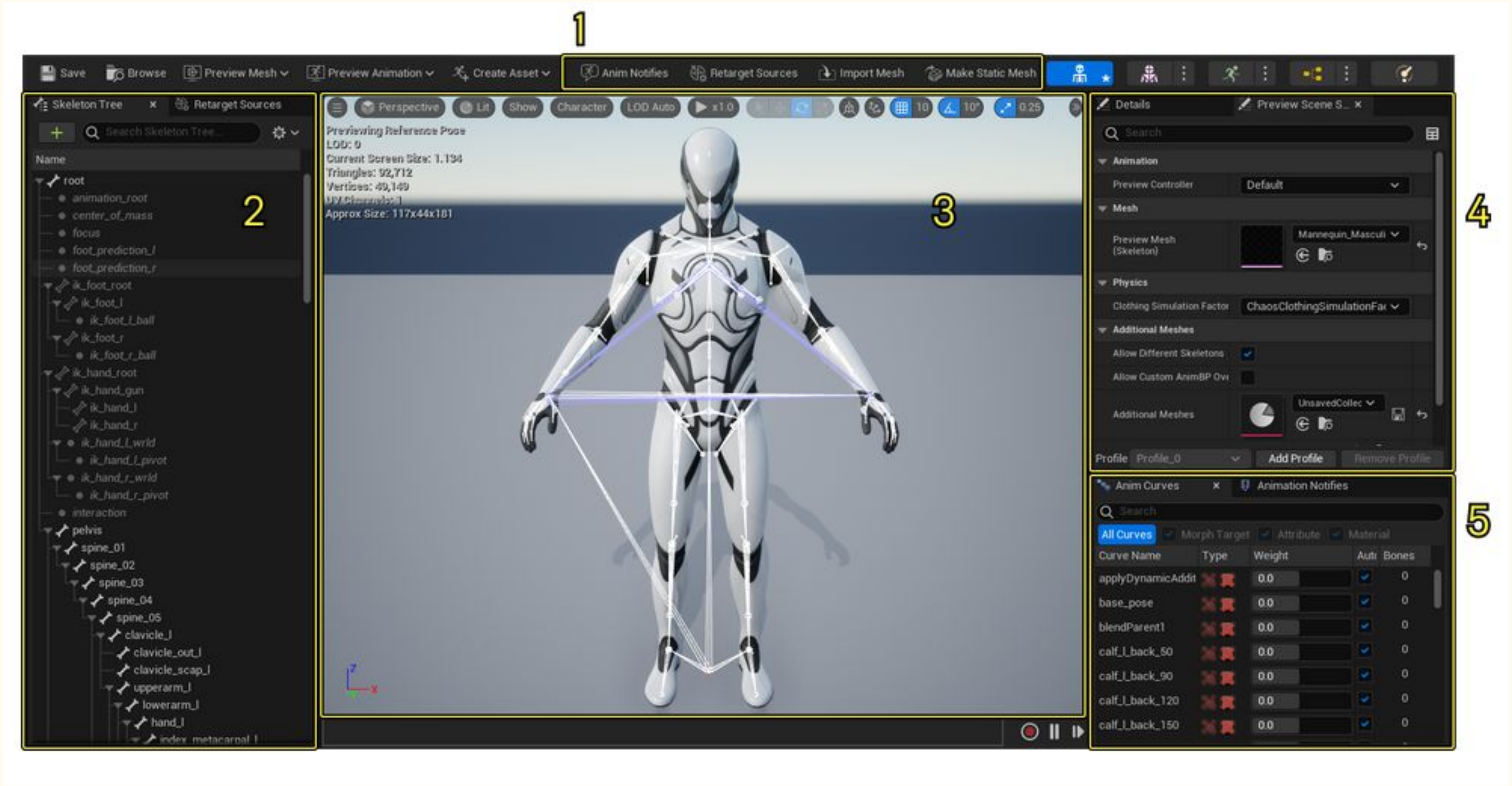
Previewing Reference Pose
LOD: 0
Current Screen Size: 1.405
Triangles: 92,712
Vertices: 49,149
UV Channels: 1
Approx Size: 117x44x181

Animation	
Preview Controller	Default
Mesh	
Preview Mesh (Skeleton)	Mannequin_Mascu
Physics	
Clothing Simulation Factory	ChaosClothingSimulationF
Additional Meshes	
Allow Different Skeletons	✓
Allow Custom AnimBP Overric	
Additional Meshes	UnsavedColl
Skeletal Meshes	0 Array elements
Settings	
Profile Name	Profile_0
Shared Profile	
Lighting	
Profile	Profile_0

Animation Editor

- The **Skeletal Mesh Animation System** in Unreal Engine is comprised of **specialized animation asset editors** that will allow you to works efficiently on various **Animations assets** related.
- With these **animation editors**, you can create **character animations**, interactions within the levels, and other **procedural behaviors**.
- When within the animation editor, you are provided with 4 main tab that you'll be using intensively
  - 1 : **Toolbar** - Like the classic toolbar, it provides various tools in order to for example **preview mesh**, **preview** an **animation**, and a complex **Create Asset** which will be allowing you to create various asset type based your current selection
  - 2 : **Viewport** - I'll not go into details, viewport allows to visualize what you are doing and some metrics which are always useful
  - 3 : **Editor Modes** : Probably the most important aspect of the animation editor has it allows you to switch between edition tab . We'll list every tab that are available and a quick description before going into more details about them
    - **Skeletal Editor** : This editor is used for working with **Skeleton Rigs** and provides visual control of **bone** and **joint** hierarchy **associated** with a Skeletal Mesh.
    - **Skeletal Mesh Editor** : The **Skeletal Mesh Editor** is where you can make **edits** to **meshes**, assign **Materials**, adjust Level of Detail (**LOD**), and test **Morph Target** functionality.
    - **Animation Sequence Editor** : If the **Skeletal Mesh** has associated **Animation Sequences**, you can **edit** and **preview** them, including augmentation tools such as, **Blendspaces**, **Morph Targets** and **Animation Notifies** here
    - **Animation Blueprint Editor** : Similar to **Unreal's Blueprint Editor**, The **Animation Blueprint editor** is a **visual scripting environment** for directing animation functionality and behaviors within the level. You can access this Mode after an **Animation Blueprint asset** has been created for your mesh.
    - **Physics Asset Editor** : The **Physics Asset Editor** is a dedicated **animation editor** you can use to manipulate the **Physics Asset** assigned to your **Skeletal Mesh**.
  - 4 : **Preview Scene Settings** - It allows you to **control various part of your previewer settings** which will be directly **visible** in the **viewport**

Animation Editor - Skeleton Editor



Previewing Reference Pose
LOD: 0
Current Screen Size: 1.134
Triangles: 92,712
Vertices: 49,149
UV Channels: 1
Approx Size: 117x44x181

- root
  - animation_root
  - center_of_mass
  - focus
  - foot_prediction_l
  - foot_prediction_r
  - ik_foot_root
    - ik_foot_l
      - ik_foot_l_ball
    - ik_foot_r
      - ik_foot_r_ball
  - ik_hand_root
    - ik_hand_gun
      - ik_hand_l
      - ik_hand_r
    - ik_hand_l_wrld
      - ik_hand_l_pivot
    - ik_hand_r_wrld
      - ik_hand_r_pivot
  - interaction
  - pelvis
    - spine_01
      - spine_02
        - spine_03
          - spine_04
            - spine_05
              - clavicle_l
                - clavicle_out_l
                - clavicle_scap_l
                - upperarm_l
                  - lowerarm_l
                    - hand_l
                      - index_metacarpal_l

Animation
Preview Controller: Default
Mesh
Preview Mesh (Skeleton): Mannequin_Masculi
Physics
Clothing Simulation Factor: ChaosClothingSimulationFac
Additional Meshes
Allow Different Skeletons: ✓
Allow Custom AnimBP Ove:
Additional Meshes: UnsavedCollec
Profile: Profile_0

Curve Name	Type	Weight	Auto	Bones
applyDynamicAddit		0.0	✓	0
base_pose		0.0	✓	0
blendParent1		0.0	✓	0
calf_l_back_50		0.0	✓	0
calf_l_back_90		0.0	✓	0
calf_l_back_120		0.0	✓	0
calf_l_back_150		0.0	✓	0

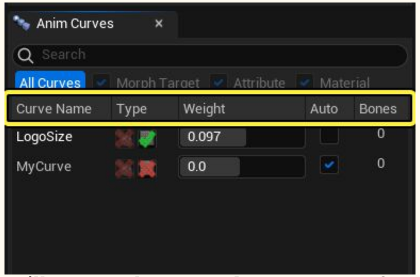
Animation Editor - Skeleton Editor

- **Skeleton Asset** are the **foundation** for all animation works with **Skeletal meshes.**
- To make it simple, a **Skeleton Asset** is the **bone representation** of a **mesh**, dictating which **bones** is **attached** with which **bones** in a **hierarchical order**. It ensure to make our arm move when our shoulder is moving
- The **Skeleton Editor** is the appropriate visual editor in order to find the **tools** and **properties** to make **changes to Skeleton Asset**
- In this editor you can **manipulate individual bones** and bone structures, attach Skeletal Mesh **Sockets**, and preview any **Animation Curves** and Animation **Notifies** associated with your skeleton. The Skeleton Editor is also where you can find the **Retargeting Manager**, a tool for managing meshes **associated** with the **current Skeleton Asset.**
- 1 : **Toolbar** - **Traditional** Toolbar with some **additions** related to Skeleton Editor like **opening the animation notifies, opening retargeting**, **importing a new mesh** for the skeleton or **create a static mesh based on the pose preview**
- 2 : **Skeleton Tree / Retarget sources** - **Skeleton tree** shows the **hierarchy** of **bones** and **sockets management** and also **options** for **retargeting**. Keep in mind that you can **select** a **bone** and **play** with it **transform** in order to **preview** it in **viewport**. The “+” also to create some additional option on the skeleton tree
    - **Socket** : More on next slides
    - **Virtual bones** : Auxiliary and additional bones children of existing ones but constrained to another. More about that here
    - **Add Time Blend Profile** : **Per-bone scales** that can be used in **transition** to **tweak weight** of specific bones relative to transition blend **time**
    - **Add Weight Blend Profile** : **Per-bone scales** that can be used in **transition** to **tweak weight** of specific bones relative to transition blend **weight**
    - **Add Blend Mask** : Used to specify the **alpha** of individual bones when **layered poses**

    Animation retargeting is a complex subject which you can find more informations here

Animation Editor - Skeleton Editor

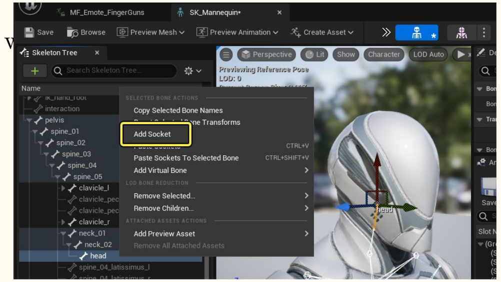
- 3 : **Viewport** - Not much to say about **viewport** as its **behavior** and **intended purpose** is kind of **always** the **same**
- 4 : **Preview / Details** - Preview just like all kind of **preview** allows to **modify preview settings** such as **selected animation**, **applied Skeletal meshes**, **viewport lighting**, etc...
  Details panel will also be really **familiar** for anymore because it **display informations** based on your **selection**. You'll be able to **tweaks transform** and **some bones** or **socket parameters** through it
- 5 : **Anim curves / Animation notifies** - You can find a lot of uses about curves and some complex setup, so we'll not go into details about them, but you can find more information about them here
  - Basically, you'll be using a **curves** when you want to **animate additional properties** and **values** in **sync** with that animation
  - **Animation Curves** also referred as **anim curves** or **curves**, which **carries float-type** values on which you can add **keyframe** within the Animation Sequence. If you need to works with curves, I highly recommend to check Unreal documentation about **Curve Editor**
  - It can be useful for example to modify **morph targets properties** or **Material parameters**
  - You can also **import** a curve from an **external tool**
  - You'll have access to an Anim Curves window on which you can parametrize the curves like
    - Curve name
    - Curve Type (Morph targets or Materials)
    - Weight : **If not auto**, **play** with it
    - Auto : Is the **weight determined** by the **sequence curve**
    - Bones : **Number** of bones connected to the curve (**parametrable from details**)
  - We'll not dive into **Morph Targets** which are complex but for **Materials**, you'll simply needs to **define** a **parameter** with the **appropriate names based on the curve**, and it'll be directly **connected**
    - Care, it will modify **every parameter** of every **Materials** on the mesh
  - You can obviously **retrieve** the **value** of a **curve** from within the **Animation Blueprint**
  - We'll see in more details **Notify** as they are quite important



Curve Name	Type	Weight	Auto	Bones
LogoSize		0.097		0
MyCurve		0.0	✓	0

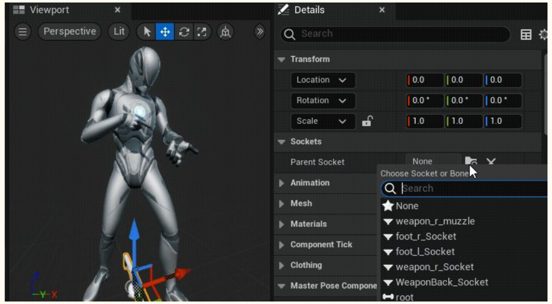
Animation Editor - Skeleton Editor - Sockets

- Let’s start with **Socket**
- Even if socket **creation** start in an more **centric artist window** which is the **Skeleton Editor,** it is most likely that they’ll **not really touch Sockets** after being placed.
- Indeed **socket** will be **placed** on a **bone**, as a **leaf** for the most cases and they’ll be **animated indirectly** by the animation of the parent bones
- However, as a **programmer** you’ll make an **heavy uses** of **socket**, primarily to **spawn Actor**, **Scene Component** or **VFX** at **specific socket**, these are some examples
    - Spawning the VFX of the gun firing at the **Muzzle Socket**
    - Spawning the **weapon** at the **storage socket**, think about a 2 handed sword which is nicely place on the back of the player
- Even if you are the **primary user** of the **socket**, be sure to always **communicate** with **designers and artists** if you are willing to **modify** a socket **transform**
    - A **socket**, even being purely an **attachment point** may provoke some **disagreement** in art department because you moves it and it is now **hiding** another element you were not aware of
    - **Socket** are something uses by **multiples department** for example **customization system**, etc…
- In order to **create a socket,** simply **right click** the **bone** you want to create a socket on, and **“Add Socket”**
- **Mesh Socket** follows the same idea, but they are a bit different
    - When you are sharing **Skeleton** with different **meshes,** you may v
      are **exclusive** for one **Skeletal Mesh**
    - **Mesh Socket** allows that by making a **Socket exist** on the **Skeletal Mesh** instead of **Skeleton**
    - To create a **Mesh Socket**, right click on an existing socket and **“Create Mesh Socket”**



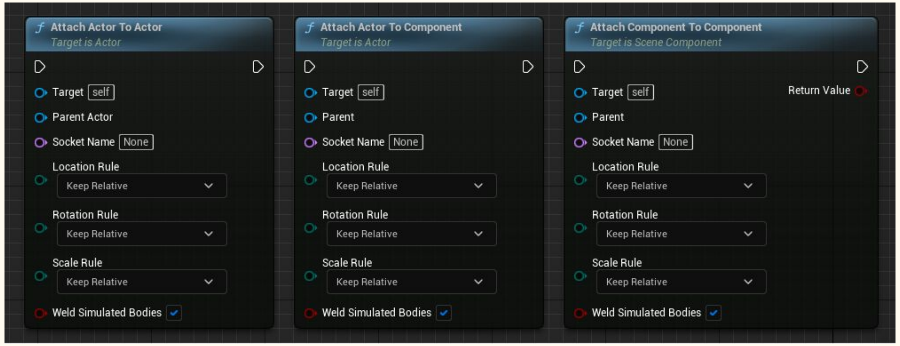
Animation Editor - Skeleton Editor - Sockets

- To conclude about sockets, we'll see the attachment from Blueprint which comes with 2 aspects
  - **Editor Attachment** : From the blueprint editor, by clicking on a **Scene Component**, you can **specify** the **Parent socket to be used**



Transform
Location 0.0 0.0 0.0
Rotation 0.0° 0.0° 0.0°
Scale 1.0 1.0 1.0
Sockets
Parent Socket None
Choose Socket or Bone
None
weapon_r_muzzle
foot_r_Socket
foot_l_Socket
weapon_r_Socket
WeaponBack_Socket
root

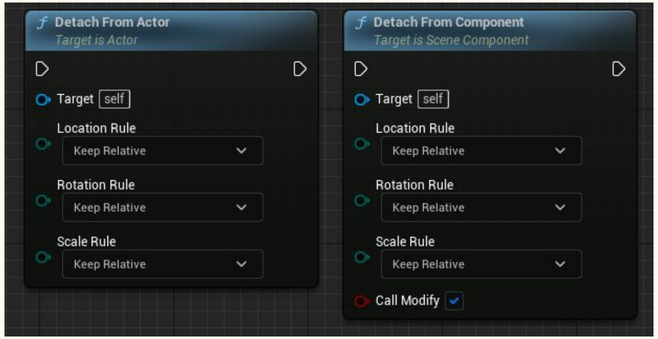
  - **Dynamic Attachment** : We'll show **blueprint node**, but obviously, **equivalence** exists in **C++**



Attach Actor To Actor
Target is Actor
Target self
Parent Actor
Socket Name None
Location Rule: Keep Relative
Rotation Rule: Keep Relative
Scale Rule: Keep Relative
Weld Simulated Bodies ✓

Attach Actor To Component
Target is Actor
Target self
Parent
Socket Name None
Location Rule: Keep Relative
Rotation Rule: Keep Relative
Scale Rule: Keep Relative
Weld Simulated Bodies ✓

Attach Component To Component
Target is Scene Component
Target self
Return Value
Parent
Socket Name None
Location Rule: Keep Relative
Rotation Rule: Keep Relative
Scale Rule: Keep Relative
Weld Simulated Bodies ✓

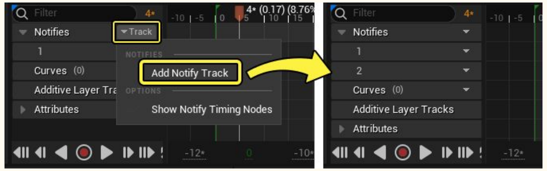


Detach From Actor
Target is Actor
Target self
Location Rule: Keep Relative
Rotation Rule: Keep Relative
Scale Rule: Keep Relative

Detach From Component
Target is Scene Component
Target self
Location Rule: Keep Relative
Rotation Rule: Keep Relative
Scale Rule: Keep Relative
Call Modify ✓

Animation Editor - Skeleton Editor - Notify

- **Animation Notifies** can became quite **complex** with a lot of uses like **Cloth simulation** etc… You'll simply give an **hint** of what they are, and **how** you can use them
- **Animation Notifications** also called **Animation Notifies** or just **Notifies**, provide a way for you to create **repeatable events synchronized** to **Animation Sequences.**
- There is multiples usages for that, for example
  - **Sounds** (such as footsteps for walk or run animations)
  - **Spawning particles**, and other types
- **Animation Notifies** are commonly **accessed** and **created** within **Animation Sequences.** To get started, open an **Animation Sequence Asset,** and locate the **Notifies track** in the **timeline.**

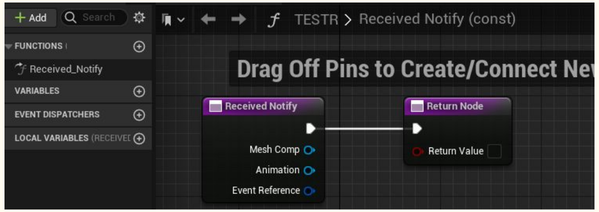


- There is **two types of Notify** which will display different information in the detail panel and visually on the timeline
  - **Notify** : They are **instant notify** which have. You'll be able to move the moment it **happens** through the **timeline**
  - **Notify State** : They offer a **frame** way to have **notifies.** You'll be able to move the **start** and **end** moment it **happens** through the **timeline**
- By defaults, Unreal comes with **multiples notify** like **Sound playing**, **Cloth Simulation**, **Particle Effects**, **Niagara Effects**, etc…

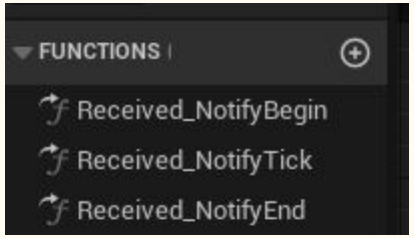
- **Skeleton Notifies** are used as **Notify Events** within Animation Blueprints, either in the **Event Graph** or **Transition Graph**. To add a Skeleton Notify Event, **right-click** in the **Event** or **Transition Graph** of your Animation Blueprint and select the **Notify** from the **Add Anim Notify Event** menu. This adds the event node to the Graph which is executed when the **Notify** is called from the **animation** it resides in

Animation Editor - Skeleton Editor - Creating Notify

- You can obviously **create custom notify** which will be available in the **notify list** when you right click on the **timeline**
- In order to do so, you can **create** a **blueprint class** and inherit either from **AnimNotify** or **AnimNotifyState**
- For **AnimNotify** you can override one main function which is **Received Notify** and will be called basically when the notify will be **reach** in the animation sequence

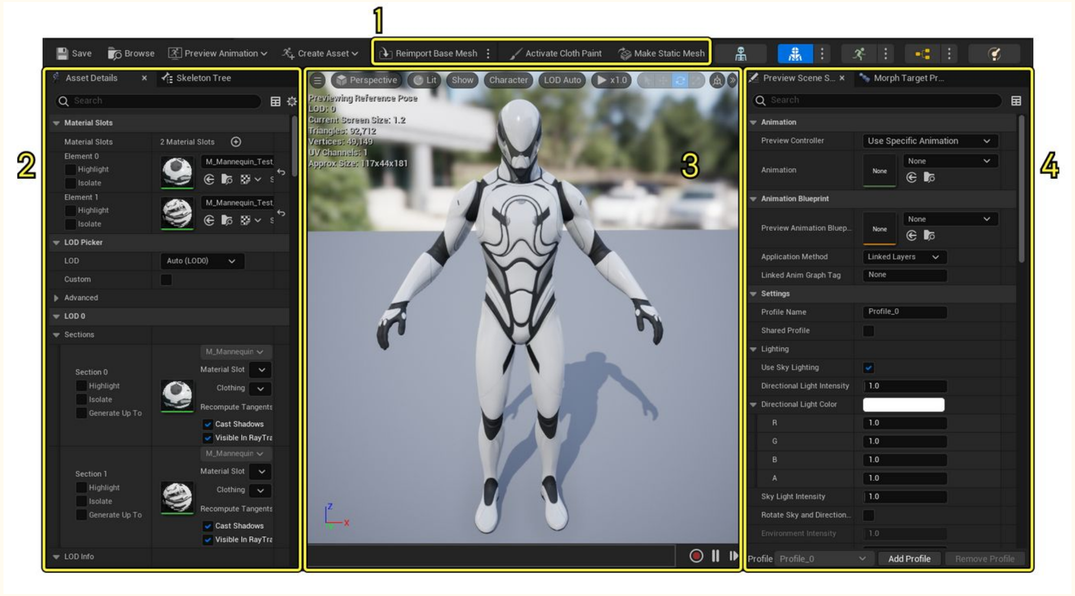


- For **AnimNotifyState** you can override three main functions which are
  - **Received_NotifyBegin** : Called when the notify frame starts
  - **Received_NotifyTick** : Called each frame while your animation sequence is within the frame
  - **Received_NotifyEnd** : Called when the notify frame ends



Animation Editor - Skeletal Mesh Editor

- This section will be quite **simple…** as we'll not speak at all about it because it is a **centric artist window** and we have other important things to talk about
- The **Skeletal Mesh Editor Mode** is where you can find the **tools** to manipulate and **preview Skeletal Mesh assets**. It is similar to the **Static Mesh Editor**. In the **Skeletal Mesh Editor** you can make changes to the **polygonal mesh** by assigning **Materials**, adding **clothing elements**, setting up **LODs** (Level of Detail), and previewing **any Morph Targets** applied to the mesh, paint, etc…



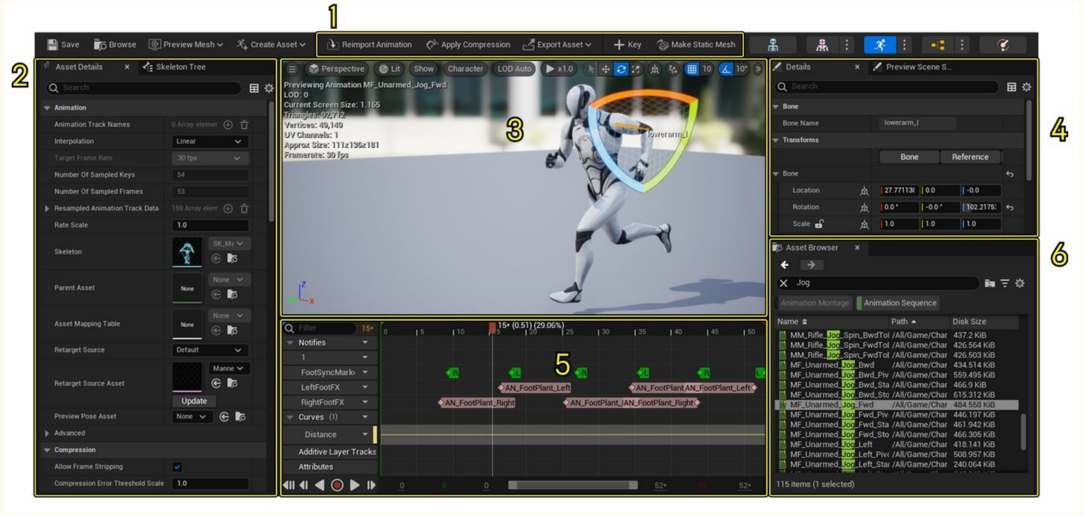
Previewing Reference Pose
LOD: 0
Current Screen Size: 1.2
Triangles: 92,712
Vertices: 49,149
UV Channels: 1
Approx Size: 117x44x181

Material Slots: 2 Material Slots
Element 0: M_Mannequin_Test
Element 1: M_Mannequin_Test
LOD: Auto (LOD0)
Section 0 / Section 1 — Material Slot: M_Mannequin

Preview Controller: Use Specific Animation
Animation: None
Preview Animation Bluep...: None
Application Method: Linked Layers
Linked Anim Graph Tag: None
Profile Name: Profile_0
Use Sky Lighting: ✓
Directional Light Intensity: 1.0
Directional Light Color — R: 1.0, G: 1.0, B: 1.0, A: 1.0
Sky Light Intensity: 1.0
Environment Intensity: 1.0
Profile: Profile_0

Animation Editor - Animation Sequence Editor

- Just like the previous slide, we'll not go into details about the Animation Sequence Editor because… well the **Animation Sequence Editor** provides **access** to the **various animation-centric** assets available for **Skeletal Meshes** in Unreal Engine. In the Animation Sequence Editor, you can edit and **preview animation Sequences**, **Montages**, **Curves**, and more.
- We've already covered various element and therefore there is no need to present it
- In you want more **informations** about all the buttons available in this window, you can refer to Unreal Documentation
- **Asset Editor (5)** will differs based on what you are **selecting**, just like the **details panel (4)**
- The **Asset Browser** will always **browse** asset **linked** to the **current Skeleton Asset**



Previewing Animation MF_Unarmed_Jog_Fwd
LOD: 0
Current Screen Size: 1.165
Triangles: 92,712
Vertices: 49,149
UV Channels: 1
Approx Size: 111x136x181
Framerate: 30 fps

Animation Track Names	0 Array elements
Interpolation	Linear
Target Frame Rate	30 fps
Number Of Sampled Keys	54
Number Of Sampled Frames	53
Resampled Animation Track Data	159 Array elements
Rate Scale	1.0
Parent Asset	None
Asset Mapping Table	None
Retarget Source	Default
Preview Pose Asset	None
Allow Frame Stripping	✓
Compression Error Threshold Scale	1.0

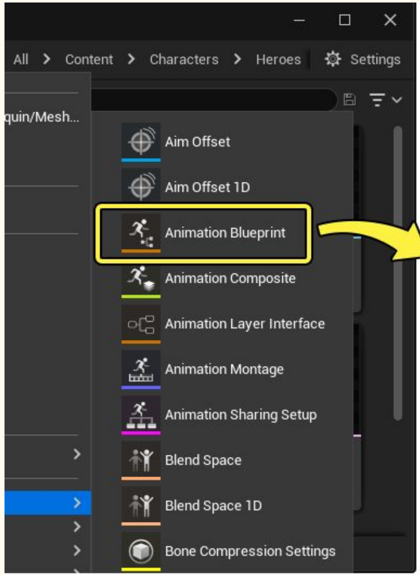
Bone Name	lowerarm_l		
Location	27.77113	0.0	-0.0
Rotation	0.0°	-0.0°	102.2175
Scale	1.0	1.0	1.0

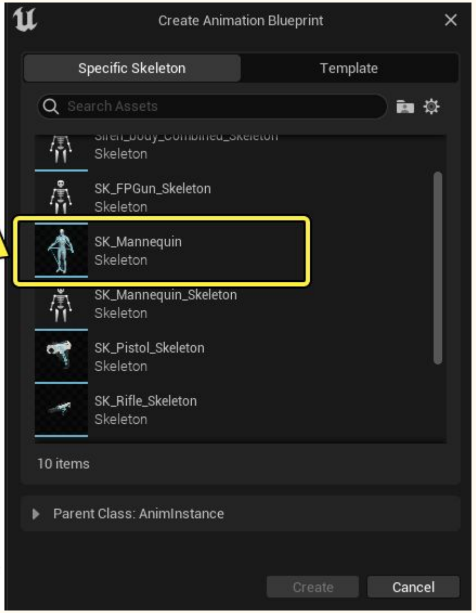
Name	Path	Disk Size
MM_Rifle_Jog_Spin_BwdToI	/All/Game/Char	437.2 KiB
MM_Rifle_Jog_Spin_FwdToI	/All/Game/Char	426.564 KiB
MM_Rifle_Jog_Spin_FwdToI	/All/Game/Char	426.503 KiB
MF_Unarmed_Jog_Bwd	/All/Game/Char	434.514 KiB
MF_Unarmed_Jog_Bwd_Piv	/All/Game/Char	559.495 KiB
MF_Unarmed_Jog_Bwd_Sta	/All/Game/Char	466.9 KiB
MF_Unarmed_Jog_Bwd_Sto	/All/Game/Char	615.312 KiB
MF_Unarmed_Jog_Fwd	/All/Game/Char	484.558 KiB
MF_Unarmed_Jog_Fwd_Piv	/All/Game/Char	446.197 KiB
MF_Unarmed_Jog_Fwd_Sta	/All/Game/Char	461.942 KiB
MF_Unarmed_Jog_Fwd_Sto	/All/Game/Char	466.305 KiB
MF_Unarmed_Jog_Left	/All/Game/Char	418.141 KiB
MF_Unarmed_Jog_Left_Piv	/All/Game/Char	508.957 KiB
MF_Unarmed_Jog_Left_Sta	/All/Game/Char	240.064 KiB

115 items (1 selected)

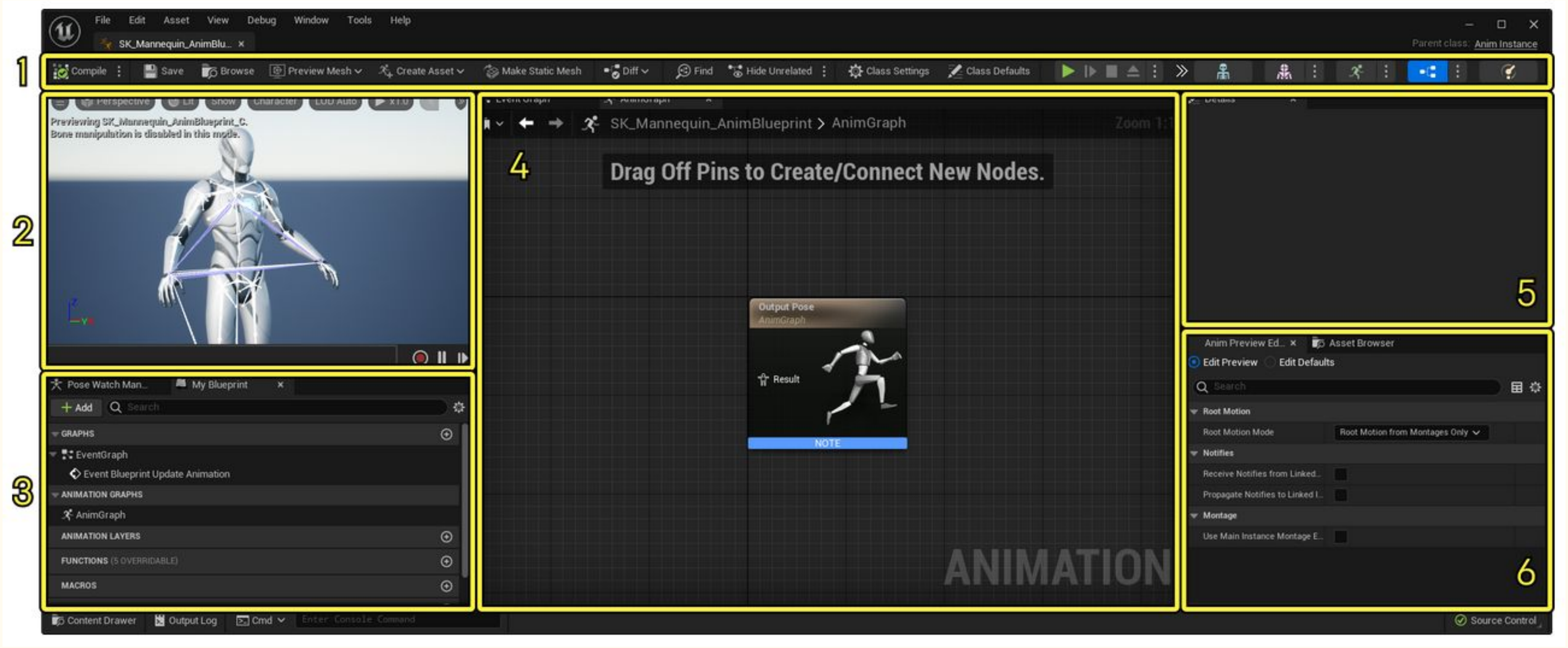
Animation Editor - Animation Blueprint Editor

- **Animation Blueprints** are **specialized Blueprints** that control the **animation** of a **Skeletal Mesh** during simulation or gameplay. Graphs are **edited** inside of the **Animation Blueprint Editor**, where you can **blend animation**, **control the bones** of a Skeleton, or **create logic** that will define the **final animation** pose for a Skeletal Mesh to use per frame.
- The very first step is obviously to **create** an **animation blueprint** which is an **asset**. Therefore, you can create this one from **right clicking** on the **content drawer.** You'll be ask to select the **skeleton** on which it will be **targeted**
- You'll then be able to **create** all the **logic** you want in the **blueprint editor** but there is a final step that you need to do in order to have your animation blueprint **control** your **skeletal mesh animation**
  - Click on the **Character** which needs to have its **animation setup**
  - In **Animation category**, for **Animation Mode** select **Use Animation Blueprint** and for **anim class**, you can input the **animation blueprint you just created**



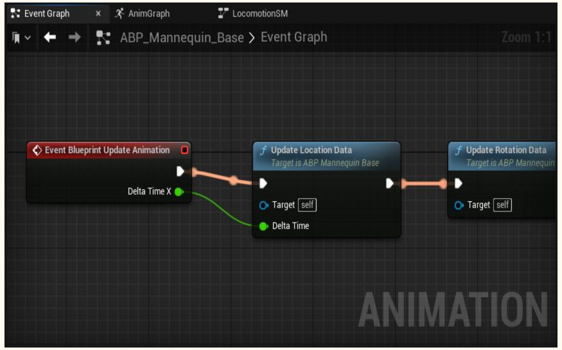


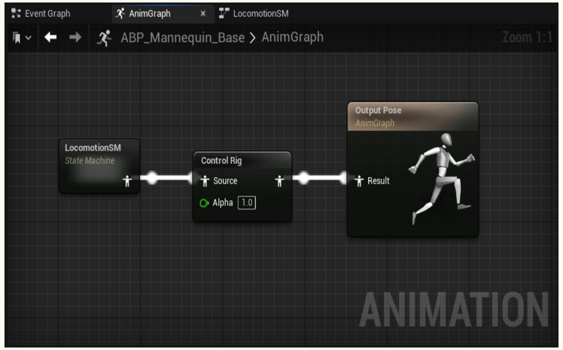
Animation Editor - Animation Blueprint Editor

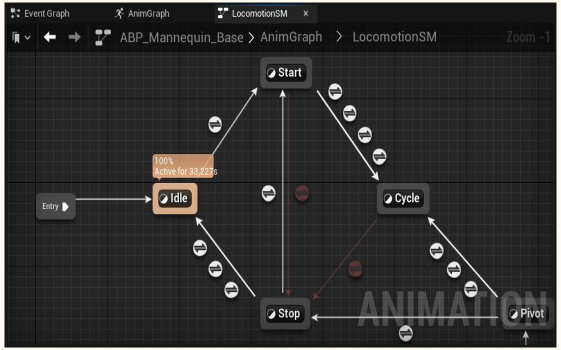


Animation Editor - Animation Blueprint Editor

- 1 : **Toolbar** - Contains various element which allows to compile, make a diff, find a specific event, variable, etc... your settings, testing, etc...
- 2 : **Viewport** - Not much to say about it. It allows to visualize the animation
- 3 - **My Blueprint** - It contains the traditional stuff that you are using in
- 4 - **Graph** - Graph is a bit more complex than the classic blueprint editor because there is multiple graph inside it
  - **Event Graph** : This is the one you are **used to**, you construct your **Blueprint-based logic** to define node **properties** which **control** the other **graphs**
  - **Anim Graph** : You construct your **pose-based logic** which evaluate the **final pose** of the **Skeletal Mesh** for the current frame
  - **State Machines** : You construct **state-based logic**, which is typically used for **locomotion.** These **state machine** will be called in the **Anim Graph**
- 5 - **Details** - Not much to say about it, it is the traditional details panel
- 6 - **Anim Preview Editor** - The Anim Preview Editor is where you can make changes to your **variables** (including Class Defaults), which will update the **Skeletal** Mesh in the viewport



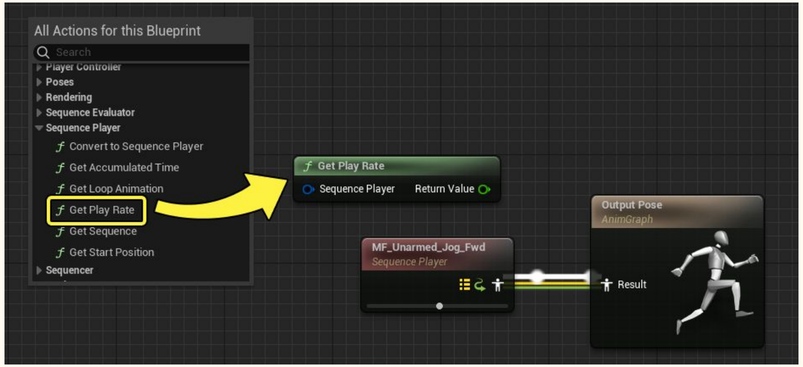




Animation Editor - Graphing in animation blueprint

- You'll see a bit more in detail the **general workflow** when **graphing** for **animation**
- We'll **not talk about State Machines** which will come in **another slide**, but mainly focusing on **Event Graph** and **Anim Graph**

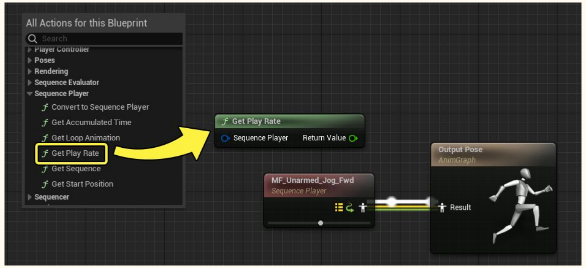
- The **AnimGraph** is where you create **animation-specific logic** for your character. Typically, this includes **creating nodes** which **control blending**, **transforming bones**, **locomotion**, and other similar **animation effects.**
- Inside the AnimGraph, you can use values **calculated** from the **EventGraph**, or **functions**, and then connect those **variables** as **inputs** for your **AnimGraph nodes**, such as **Blend Spaces**. The combined effects of your nodes and their values are connected to the **Output Pose**, which is where the **final pose** of the character is defined for **every frame**.
- Just like regular blueprint editing, you can access the **contextual menu** by **right clicking** and select what you want
- Obviously, **Anim Graph** being different, there is **Pose information** which is the **most essential** with the **Output pose** which will control how the character is animated



Animation Editor - Graphing in animation blueprint

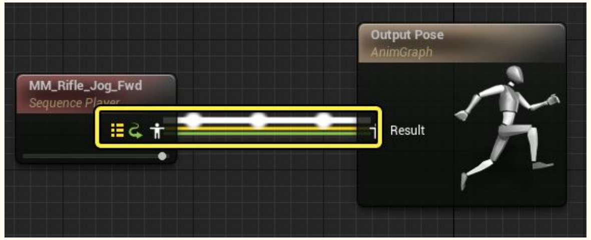
- You'll see a bit more in detail the **general workflow** when **graphing** for **animation**
- We'll **not talk about State Machines** which will come in **another slide**, but mainly focusing on **Event Graph** and **Anim Graph**

- The **AnimGraph** is where you create **animation-specific logic** for your character. Typically, this includes **creating nodes** which **control blending**, **transforming bones**, **locomotion**, and other similar **animation effects**.
- Inside the AnimGraph, you can use values **calculated** from the **EventGraph**, or **functions**, and then connect those **variables** as **inputs** for your **AnimGraph nodes**, such as **Blend Spaces**. The combined effects of your nodes and their values are connected to the **Output Pose**, which is where the **final pose** of the character is defined for **every frame**.
- Just like regular blueprint editing, you can access the **contextual menu** by **right clicking** and select what you want
- Obviously, **Anim Graph** being different, there is **Pose information** which is the **most essential** with the **Output pose** which will control how the character is animated
- We'll not have time to dive into nodes that are available in the anim graph, which is more the works of technical animator, but in smaller team, you may need to do them by yourself, here is the official documentation about node reference



Animation Editor - Graphing in animation blueprint

- **Pose connections**, which are denoted with the **Pose icon**, have their execution shown as **pulsing links** along their connection lines.
- In the AnimGraph, the flow of **execution represents poses** being **passed** from **one** node to **another**. Some nodes, such as **Blends**, have **multiple inputs** and **make** a decision internally on which **input** is currently part of the **flow** of **execution**. This determination is usually dependent on some external input, like the value passed to a data pin.
- **Poses** and **nodes** can also contain **several internal attributes**, which are represented by **parallel execution lines** between the connected pins and icons on the node. This information conveys **additional attributes** that are being sent along with the animation pose.



Attribute	Icon	Description
Curves	∫	Passes Anim Curve data.
Attributes	☰	Passes Animation Attribute data.
Sync	🕒	Passes Sync Group data.
Root Motion	↪	Passes Root Motion data.
Inertial Blending	●	Passes Inertialization data. This indicator only appears when the Inertialization node is requested, typically when a blend occurs.

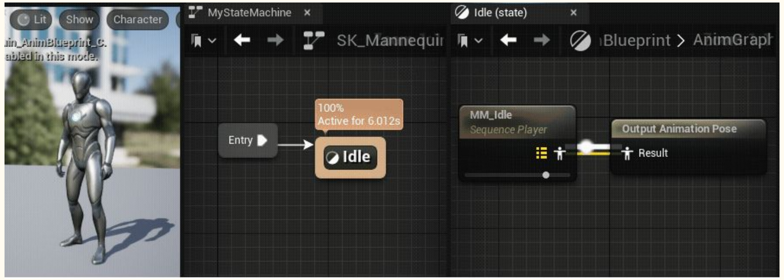
State Machines

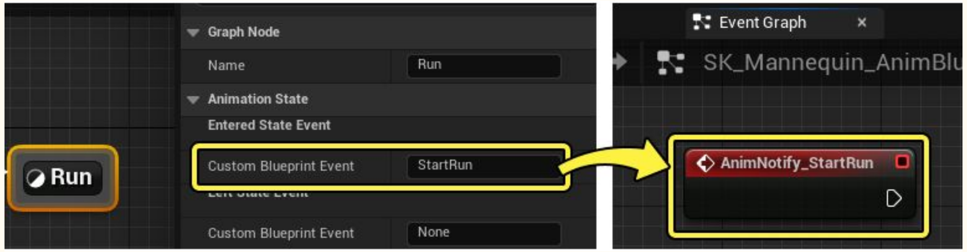
- **State Machines** are **modular systems** you can build in **Animation Blueprints** in order to define certain **animations** that can play, and when they are **allowed** to play.
- Primarily, this type of system is used to **correlate animations** to **movement states** on your **characters**, such as **idling**, **walking**, **running**, and **jumping**.
- With **State Machines**, you will be able to **create states**, define **animations** to play in those **states**, and create various **types** of **transitions** to control when to switch to other states. This makes it easier to create **complex animation blending** without having to use an overly complicated Anim Graph.
- State Machine are **not asset** but are directly **created** from the **Anim Graph**. State Machines are **subgraphs** within the Anim Graph, therefore you can see the **State Machine** graph within the **My Blueprint panel.**
- To create one, **right-click** in the **Anim Graph** and select **State Machines > Add New State Machine**. Connect it to the **Output Pose**.

- There is some mandatory nodes in a state machine
  - **Entry Point** : All State Machines **begin** with an **entry point**, which is typically used to **define** the **default state.** In most common locomotion setups, this would be the **character idle state**.
- From the entry point, you'll then be able to **create states**
  - To create the **default state**, **click** and **drag** the **entry output pin and release the mouse**, which will expose the **context menu**. Select **Add State**. This will create the **new state** and **connect it to the entry output**, making this state active by default.
  - You can also **right click** in the **graph** to open the **context menu**
  - States are **organized sub-sections** within a **State Machine** that can **transition to** and **from** each other regularly. States themselves contain their own **Anim Graph layer**, and can contain any kind of animation logic. For example, an **idle state** may just contain a character's idle animation, whereas a weapon state may contain **additional logic** for shooting and aiming. Whatever logic is used, the purpose of a **state** is to produce a **final animation** or **pose unique** to that state.
  - For **simple animation usage**, you can simply **drag & drop** an **animation** into the **state** to assign it

State Machines - States

- A state being a **complex element**, you can **double click** on it to open its **internal layer graph**
- Like **Anim Graphs**, states contain a final **Output Pose** node to connect your animation logic to. When the state is active, this logic will execute. When a **different state** is active, this logic will no longer execute.



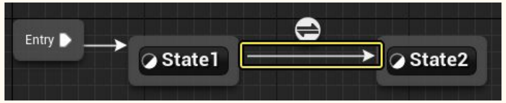


Graph Node
Name: Run
Animation State
Entered State Event
Custom Blueprint Event: StartRun
Left State Event
Custom Blueprint Event: None

- In the **details panel**, when you are **selecting** an asset, you'll have access to **state properties**
  - **Name** : Simply the state name
  - **Entered State Event** : Creates a **Skeleton Notify** with the name used in the **Custom Blueprint Event field**. This notify will execute when the **state becomes active** and **starts to transition**
  - **Left State Event** : Creates a **Skeleton Notify** with the name used in the **Custom Blueprint Event field**. This notify will execute when starting to **blend to another state.**
  - **Fully Blended State Event** : Creates a **Skeleton Notify** with the name used in the **Custom Blueprint Event field**. This notify will execute when this state is **fully blended to.**
  - **Always Reset on Entry :** Enable it to make the state fully reset which will most likely
    - Sequence players will restart at the animation start time.
    - Properties will initialize at their default values.

State Machines - Transitions

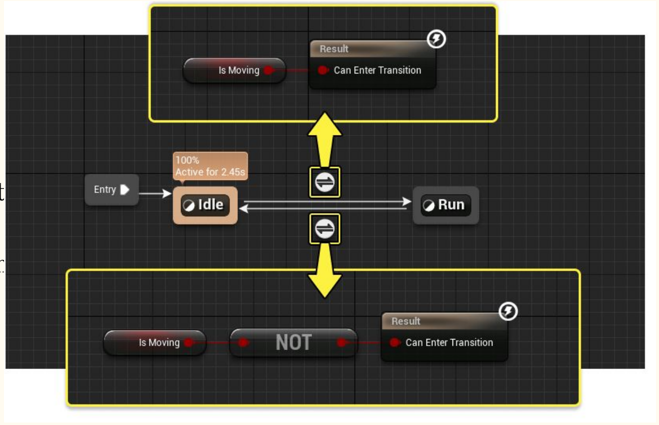
- To control which **states** can **blend** to **another**, you can create **transitions**, which are **links** between states that define the **structure** of your **State Machine**.



- To **create** a **transition**, **drag** from a **state border** to **another state**
- Transitions are **single-direction**, so if two states are intended to transition **back** and **forth**, you need to create a **transition** for **each direction**.
- Transition, just like **state**, can be **complex behavior** having as an exit node if the transition can be operated or not
- If you **double click** on a transition, it will **open** it up and you'll be able to **code** your **transition condition**

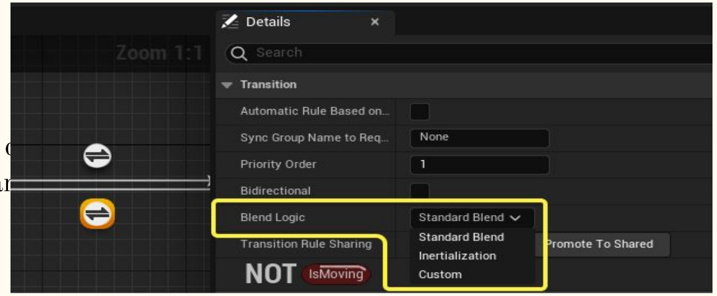
- You can have access to a **variety** of **functions** related the **transition graph**
- You can also have access to a number of **Animation Notify Functions**
- You can find the reference of this function

- While **transitioning,** if another **state** becomes **active,** t and transition to that new state instead. When this certain **Animation Notifies to the interruption.** This m the **interruption occurs.**
    - To set up transition **interruption** notify behavior, the **Transition Interrupt** properties in the **details panel.**



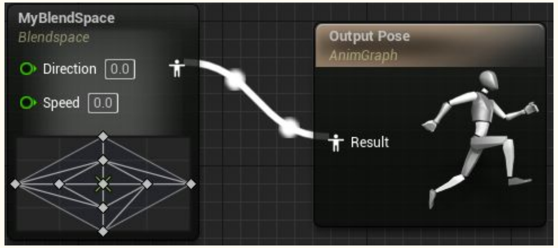
State Machines - Transitions

- **Transition** also means **blending** between **2 animations**
- There are **three main types** of **state transition blending** you can use when deciding how you want states to transition. You can **choose** any of these **types** by selecting the **transition** and locating the **Blend Logic property in the Details panel**.
  - **Standard Blend** : **Standard Blend** is the **default transition option** and contains settings for **duration**, **curve**, and other **basic controls**.
    - Transition Crossfade Sharing : You can choose to **share** this **blending transition** between **severals transition**
    - Duration : The **length of time**, in seconds, that the **transition takes**.
    - Mode : The **curve type** to use when blending with this transition. Holding Ctrl + Alt on each of the options will display a preview of the curve shape.
    - Custom Blend Curve : If **Mode is set to Custom**, then this is where you specify a custom created **Curve Asset** to use as the **curve shape**, when blending with this transition.
    - Blend Profile : You can optionally specify a **Blend Profile** here, if you want certain **bones** to **blend faster** than others during this transition.
  - **Inertialization** : Inertia is quite **situational** as it is **not working** on every **transition** based on the animation used
    - If you are using **inertialization** as a **blend type**, you must also ensure that the **Inertialization node** is used in the Anim Graph. It must be placed **after** your **State Machine evaluates**
    - Keeping your blend duration short, less than 0.4 seconds is best.
    - When the poses are extremely different, do not use inertialization.
    - Refer to [Unreal documentation]() for more details
  - **Custom** : **Custom** **blends** are blends you c own **Anim** **Graph** **layer**, with its **duration** ar **standard blend settings**

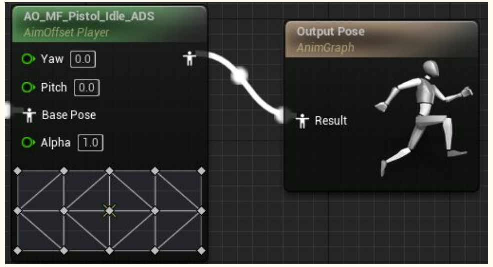

Details
Search
Transition
Automatic Rule Based on...
Sync Group Name to Req... None
Priority Order 1
Bidirectional
Blend Logic Standard Blend
Transition Rule Sharing
Standard Blend
Inertialization
Custom
Promote To Shared

Blend spaces

- **Blend Spaces** are assets that allow **multiple animations** or **poses** to be blended by **plotting** them onto either a **one** or **two-dimensional graph**. This graph can then be **referenced** within **Animation Blueprints** where the **blending** can be **controlled** by gameplay input or other variables.
- **Blend Spaces** are asset and can be **created** from the **content drawer** and there is multiple type of blendspace
  - **Blend Space / Blend Space 1D** : Base **variety** of Blend Space, which provides all the **main functionality** of blending animations along the graph. They are intended to be **referenced** within **Animation Blueprints** as a **base layer**, with secondary animations proceeding from it.

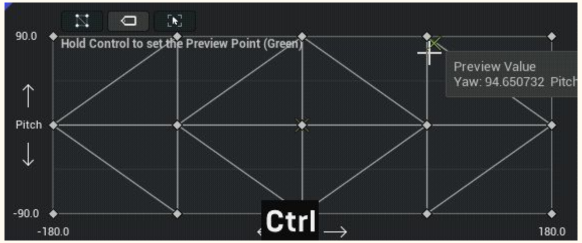


  - **Aim Offset / Aim Offset 1D** : Aim Offsets are **Blend Spaces** meant to **contain mesh-space additive** animations as their samples. Typically, these are used to **create weapon** or other **look-at aiming blend spaces**. Aim Offset Animation Blueprint nodes are intended to receive an **input pose** along with the **normal axis inputs**. To check more about Aim Offset, check Unreal Documentation



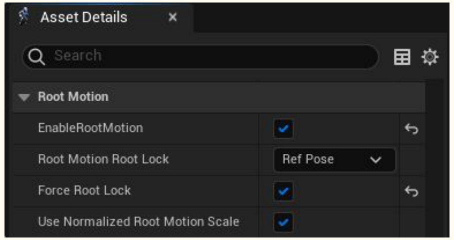
Blend spaces

- Both **Blend Spaces** and **Aim Offsets** also support **single-axis (1D)** variants. Typically these Blend Spaces are used when you only need a **single axis** of blending. In the case of **Blend Space 1D** or **Aim Offset 1D**, the graph only provides a **horizontal axis.**
- **Blends Spaces** are based on **axis**, which mean you'll need to define that from the **Asset Details**. In Vertical / Horizontal axis category, you can define
  - Name
  - Minimum Value
  - Maximum Value
  - Obviously, this values will **differs** based on what you **need**, such as **-90** to **90**, or **-180** to **180** if you are creating an **Aim Offset** and want your grid to match **rotation values.**
- You'll then needs to **populate** your **blend space**. In order to do so, from the asset browser window, you can **drag & drop** an **animation** into the **blend space**
- **Holding Shift** will cause the **animations** to **snap** to the **grid points** which can be useful for alignment
- Once you have a **completed graph**, you can **reference** and **manipulate** the **Blend Space** in **Animation Blueprints**. To add your Blend Space, **right-click** in the AnimGraph and locate your **Blend Space**. You can also **drag it** from the **Content Browser** into the graph.
- If you want to learn more about to use blend space from within Animation Blueprint, you can check Unreal Documentation



Root Motion

- With **Root Motion animations**, you can drive **character movement** with animation data to create more **grounded movement** within levels.
- Within a level, a character is composed of **many Components**. A character's movement is often driven by the **character's Movement Component**, with **animation playback** layered on top to communicate the visual feedback of motion.
- Animations are driven by the **Skeletal Mesh's Skeleton**, which is composed of **bones**. The Root Bone is the Skeleton's foundational bone; unlike other bones, the **Root Bone** is **not built to represent an element** of the Skeletal Mesh, like a leg or arm, but rather, is a reference point for the entire skeletal structure. Some animations do not have any animation data on the Root Bone, it remains **stationary** and **grounds** the **skeleton** and the animation to a single point, while others animate the **Root Bone** to follow the animation's displacement in 3D space.
  - In the absence of **any character movement**, animations with a **stationary Root Bone** will play in place with no actual movement or displacement.
  - Some other animation may have **movement assigned** to the **Root Bone**
- However, an animation **containing animation** data on the **Root Bone** does not **affect** the character movement by **default**, it must first be **enabled** with the **Root Motion property**.
- Each **individual Animation Sequence or Montage** must be **toggled** to **Enable Root Motion**. This property is found and modified within the **Asset Details panel**
- After enabling **Root Motion** within an individual **Animation Sequence's** parameters, you can determine from the **animation blueprint** how you want the root motion to be applied on the **Class Defaults settings**



Root Motion

EnableRootMotion	✓
Root Motion Root Lock	Ref Pose
Force Root Lock	✓
Use Normalized Root Motion Scale	✓

A much more...

- The advantage of **Unreal's Animation system** is that it is so **develop** that there is a lot of **feature** that are kind of **ready** to **use** or **implement** in order to **leverage** your **animation process** and **quality** for games
- Unfortunately, some topic are really specific and would require a lesson alone to get an idea of what is possible
- You'll maybe never works on some topics, but it is always good to know that they exists like
    - **Pose Warping** : You can enable Pose Warping to dynamically adjustment a character's animated poses to align with character movement, using Root Motion. With Pose Warping, you need fewer individual animations to reach the same level of animated-movement coverage as before. This reduces the dependency on animation-dictated gameplay, and allows animation and gameplay tuning to evolve side by side during development.
    - **Motion Warping** : With Motion Warping, you can dynamically warp windows of a character's animation to align with Root Motion or to align with assigned Warp Targets. With this feature, you can rely less on the manual creation and fine tuning of animations to fit specific object interactions, and apply logic to bend base animations to fit pre-established conditions.
    - **IK** : The IK Rig system provides a method of interactively creating Solvers that perform pose editing for your skeletal meshes. The resulting IK Rig Asset can then be embedded into any animation systems, such as Animation Blueprints to dynamically modify the pose-based solver parameters
    - **Distance Matching** : With Distance Matching, Animation Sequences can be driven by a calculated distance variable rather than time-based linear playback. This document provides an overview of Distance Matching, and a workflow example demonstrating the implementation process.

Time to.... highlight a concept

Input System & Enhanced Input System

Practice

- Practice will be **quite particular** this time because it may **require** that you have **animation** in order to **test** and include them into the general practice and your follow-through project and they needs to be compatible with
- In order to do so, we'll be working on a importing a mixamo character, a set of animation and replicating an existing animation blueprint from Unreal directly in your follow-through project which will be representing the guard
- **General**
    - Be sure to **look closely** on how **template animation blueprint** is working
    - If there is some subjects that interest you, you can look at it more closely but do not waste too much time for now
- **Follow-through project**
    - Import a mesh and a set of animation which will be representing your AI guard
        - You can make the AI guard as complex as you want, but you'll need at least **Idle**, **Walk** and **Run**
        - Create an animation blueprint which will be representing your AI guard
            - You can take as a reference the one that is used on the Player Character
            - Blend spaces would be a good idea for movement
            - State Machine would be a good idea for locomotion
        - Assign the animation blueprint to the IA Guard mesh
    - Important a animation for attack which will be used by the guard when he's in range of attack
        - Create an animation montage based on the animation sequence you'll be importing
        - Play that animation montage somewhere, we'll trigger this properly later
        - From that animation montage, find a way to send an event at a specific moment which should kill the player if he is in range