



synty®

Sword Combat

Keyframe Animation pack

User Guide

Version: 1.0.0

www.syntystore.com

Table of Contents

Table of Contents	2
1. Installation and set up	4
Requirements	4
Unity Asset Store Installation	4
Synty Store Installation	4
2. Animation Pack Components	5
Key Features	5
List of Animations	5
Animation Naming Conventions	6
3. Prop Bone	7
Synty Characters	7
Prop Bone Binder Tool	9
Setting up your character	9
Prop Bone Binder Tool Troubleshooting	12
Removing the Prop Bone Binder Tool	13
Prop Bone with Non-POLYGON Rigs	13
Using Animations Without Prop Bones	13
4. Quick Start	14
Gallery Scene	14
Build your own	14
Applying Animations to Characters	15
Creating a new Avatar	15
Modifying existing Avatar	16
Integrating Synty Sword Combat with Synty Base Locomotion	16
Adding a sword to existing Synty Base Locomotion Animations	18
5. Combat Animations	21
Attack Animations	21
Heavy and Light Attacks	22
Attack Combos	22
Leaping Attack	23
Blocking and Parrying	24
Sheathing/Unsheathing the sword	24
6. Tips for working with Sword Combat	26
Animation Files	26
Optimizing Animation Performance	26
Mecanim Humanoid Character Avatar	27
Adjusting Avatar Properties	28
7. Naming conventions	29
8. Terms of use	29
9. Glossary	31

Introduction

Welcome to SwordCombat, a specialised animation pack designed for seamless integration across multiple platforms and smooth implementation into your chosen development environment.

Featuring meticulously crafted sword combat animations, this pack will accelerate your workflow by providing a cohesive collection of motions to elevate humanoid character performances to new heights.

Witness your characters come to life with immersive fluidity as they engage in exhilarating duels, and get ready to unleash the power of sword combat in your creations like never before.

From all of us at Synty,
We thank you for the support!

1. Installation and set up

Requirements

Animation component files

- Unity supports fbx imports across all current LTS versions

Unity Asset Store Installation

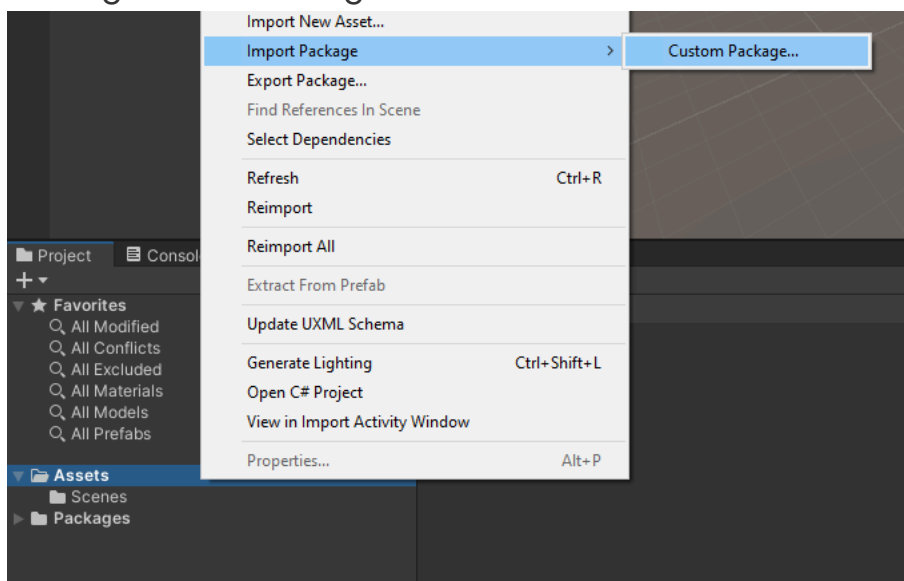
If you purchased Sword Combat from the Unity Asset Store, you can download the latest package using the Unity Package Manager, as follows:

1. Open Unity Package Manager from the top menu 'Window' > 'Package Manager'
2. Change the Packages drop down to 'My Assets'
3. Look for or search 'Synty Sword Combat' and click 'Install'

Synty Store Installation

If you purchased Sword Combat from www.syntystore.com, you will need to do the following:

1. Open your Project window (from top menu select 'Window' > 'General' > 'Project')
2. Right Click in the 'Assets' directory and select 'Import Package' > 'Custom Package...' from the right click menu



3. Navigate to where you downloaded the .unitypackage file and click 'Open'
4. You will be presented with a window to import the package, click 'Import'.

2. Animation Pack Components

The Animation Pack features 105 purpose built animations, an ideal base for building a seamless humanoid sword combat system. It includes such essential movements as attacking, blocking, hit reactions, deaths, dodges and more.

Key Features

- **Animation Sets**
 - 105 Sword Combat animations, including RootMotion versions, designed for a complete combat system with smooth blending and transitions.
- **Humanoid Character Avatar**
 - These animations integrate with Unity's Mecanim system, providing a foundation for utilising animations across different characters easily.
- **Gallery Scene**
 - Scene of all the animations as separate assets for users to view clearly as individual animations.

List of Animations

Attack A_Attack_HeavyCombo01A_ReturnToIdle_RootMotion_Sword.fbx A_Attack_HeavyCombo01A_ReturnToIdle_Sword.fbx A_Attack_HeavyCombo01A_RootMotion_Sword.fbx A_Attack_HeavyCombo01A_Sword.fbx A_Attack_HeavyCombo01B_ReturnToIdle_RootMotion_Sword.fbx A_Attack_HeavyCombo01B_ReturnToIdle_Sword.fbx A_Attack_HeavyCombo01B_RootMotion_Sword.fbx A_Attack_HeavyCombo01B_Sword.fbx A_Attack_HeavyCombo01C_ReturnToIdle_RootMotion_Sword.fbx A_Attack_HeavyCombo01C_ReturnToIdle_Sword.fbx A_Attack_HeavyCombo01C_RootMotion_Sword.fbx A_Attack_HeavyCombo01C_Sword.fbx A_Attack_HeavyFlourish01_ReturnToIdle_RootMotion_Sword.fbx A_Attack_HeavyFlourish01_ReturnToIdle_Sword.fbx A_Attack_HeavyFlourish01_RootMotion_Sword.fbx A_Attack_HeavyFlourish01_Sword.fbx A_Attack_HeavyStab01_ReturnToIdle_RootMotion_Sword.fbx A_Attack_HeavyStab01_ReturnToIdle_Sword.fbx A_Attack_HeavyStab01_RootMotion_Sword.fbx A_Attack_HeavyStab01_Sword.fbx A_Attack_LightCombo01A_ReturnToIdle_RootMotion_Sword.fbx A_Attack_LightCombo01A_ReturnToIdle_Sword.fbx A_Attack_LightCombo01A_RootMotion_Sword.fbx	Death A_Death_B_01_Pose_Sword.fbx A_Death_B_01_Sword.fbx A_Death_F_01_Pose_Sword.fbx A_Death_F_01_Sword.fbx A_Death_L_01_Pose_Sword.fbx A_Death_L_01_Sword.fbx A_Death_R_01_Pose_Sword.fbx A_Death_R_01_Sword.fbx Dodge A_Dodge_B_RootMotion_Sword.fbx A_Dodge_B_Sword.fbx A_Dodge_F_RootMotion_Sword.fbx A_Dodge_F_Sword.fbx A_Dodge_L_RootMotion_Sword.fbx A_Dodge_L_Sword.fbx A_Dodge_R_RootMotion_Sword.fbx A_Dodge_R_Sword.fbx Hit A_Hit_B_React_Sword.fbx A_Hit_B_Stagger_RootMotion_Sword.fbx A_Hit_B_Stagger_Sword.fbx A_Hit_F_React_Sword.fbx A_Hit_F_Stagger_RootMotion_Sword.fbx A_Hit_F_Stagger_Sword.fbx
---	--

<div>A_Attack_LightCombo01A_Sword.fbx</div> <div>A_Attack_LightCombo01B_ReturnToIdle_RootMotion_Sword.fbx</div> <div>A_Attack_LightCombo01B_ReturnToIdle_Sword.fbx</div> <div>A_Attack_LightCombo01B_RootMotion_Sword.fbx</div> <div>A_Attack_LightCombo01B_Sword.fbx</div> <div>A_Attack_LightCombo01C_ReturnToIdle_RootMotion_Sword.fbx</div> <div>A_Attack_LightCombo01C_ReturnToIdle_Sword.fbx</div> <div>A_Attack_LightCombo01C_RootMotion_Sword.fbx</div> <div>A_Attack_LightCombo01C_Sword.fbx</div> <div>A_Attack_LightFencing01_ReturnToIdle_RootMotion_Sword.fbx</div> <div>A_Attack_LightFencing01_ReturnToIdle_Sword.fbx</div> <div>A_Attack_LightFencing01_RootMotion_Sword.fbx</div> <div>A_Attack_LightFencing01_Sword.fbx</div> <div>A_Attack_LightLeaping01_ReturnToIdle_RootMotion_Sword.fbx</div> <div>A_Attack_LightLeaping01_ReturnToIdle_Sword.fbx</div> <div>A_Attack_LightLeaping01_RootMotionHorizontal_Sword.fbx</div> <div>A_Attack_LightLeaping01_RootMotionVertical_Sword.fbx</div> <div>A_Attack_LightLeaping01_RootMotion_Sword.fbx</div> <div>A_Attack_LightLeaping01_Sword.fbx</div> <div>Block</div> <div>A_Block_Begin_Sword.fbx</div> <div>A_Block_End_Sword.fbx</div> <div>A_Block_Loop_Sword.fbx</div> <div>A_Parry_Break_RootMotion_Sword.fbx</div> <div>A_Parry_Break_Sword.fbx</div> <div>A_Parry_F_CounterShove_RootMotion_Sword.fbx</div> <div>A_Parry_F_CounterShove_Sword.fbx</div> <div>A_Parry_F_PommelStrike_RootMotion_Sword.fbx</div> <div>A_Parry_F_PommelStrike_Sword.fbx</div> <div>A_Parry_F_ReturnToBlock_Sword.fbx</div> <div>A_Parry_F_Sword.fbx</div> <div>A_Parry_L_Sword.fbx</div> <div>A_Parry_R_Sword.fbx</div>	<div>A_Hit_L_React_Sword.fbx</div> <div>A_Hit_L_Stagger_RootMotion_Sword.fbx</div> <div>A_Hit_L_Stagger_Sword.fbx</div> <div>A_Hit_R_React_Sword.fbx</div> <div>A_Hit_R_Stagger_RootMotion_Sword.fbx</div> <div>A_Hit_R_Stagger_Sword.fbx</div> <div>A_KnockDown_Begin_Sword.fbx</div> <div>A_KnockDown_Begin_RootMotion_Sword.fbx</div> <div>A_KnockDown_End_Sword.fbx</div> <div>A_KnockDown_Loop_Sword.fbx</div> <div>A_Stun_Begin_Sword.fbx</div> <div>A_Stun_End_Sword.fbx</div> <div>A_Stun_Loop_Sword.fbx</div> <div>Idle</div> <div>A_Idle_Base_Sword.fbx</div> <div>A_Idle_EnergeticStance01_Sword.fbx</div> <div>A_Idle_Flourish01_Sword.fbx</div> <div>A_Idle_Menacing01_Begin_RootMotion_Sword.fbx</div> <div>A_Idle_Menacing01_Begin_Sword.fbx</div> <div>A_Idle_Menacing01_End_RootMotion_Sword.fbx</div> <div>A_Idle_Menacing01_End_Sword.fbx</div> <div>A_Idle_Menacing01_RootMotion_Sword.fbx</div> <div>A_Idle_Menacing01_Sword.fbx</div> <div>A_Draw_Sword_Masc.fbx</div> <div>A_Draw_Sword_Femn.fbx</div> <div>A_Idle_Sheathed_Sword_Masc.fbx</div> <div>A_Idle_Sheathed_Sword_Femn.fbx</div> <div>A_Sheathe_Sword_Masc.fbx</div> <div>A_Sheathe_Sword_Femn.fbx</div>
---	---

Animation Naming Conventions

Example: A_Attack_LightFencing01_ReturnToIdle_RootMotion_Sword.fbx

<Filetype> _ <AnimationType> _ <Direction> _ <Description> _
<Section> _ <RootMotion> _ <AnimationSet>

- **Filetype:** The broad-scope prefix. Here ‘A’ stands for ‘Animation’
- **AnimationType:** The general category of animation. Here this is an ‘Attack’ animation
- **Direction:** If required, this tag outlines a direction of movement (e.g. ‘F’ for ‘Front’ or ‘L’ for ‘Left’)
- **Description:** Describes the animation with more specificity. Here the attack is a ‘light attack’ (faster, one handed), and in a ‘fencing’ style of combat, hence ‘LightFencing01’. The numbering accounts for the

possibility of more than one 'LightFencing' attack.

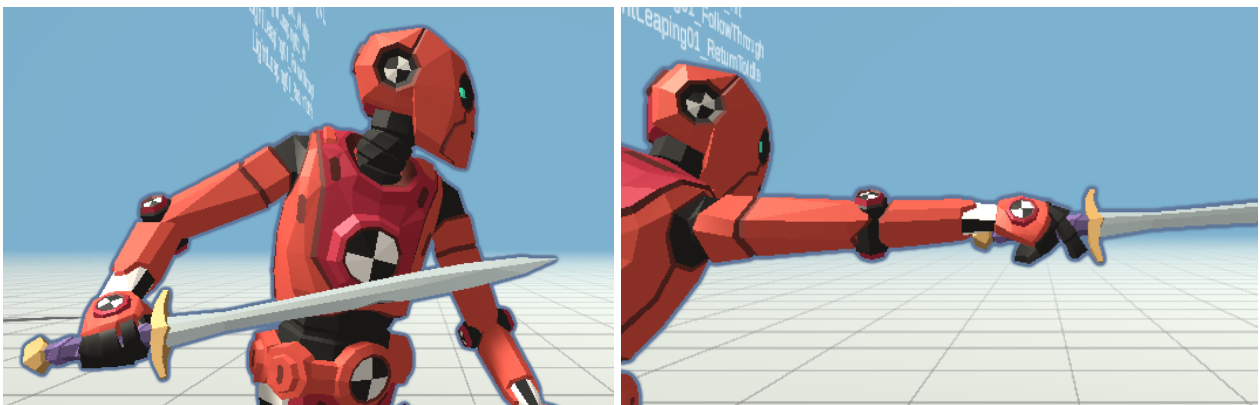
- **Section:** If required, the description can get more granular. Here, this is specifically the 'ReturnToldle' part of the attack move.
- **RootMotion:** If this is the root-motion version of an animation, it is indicated with an extra tag in the name.
- **AnimationSet:** Here 'Sword' indicates that the animation belongs to the Sword Combat Pack.

3. Prop Bone

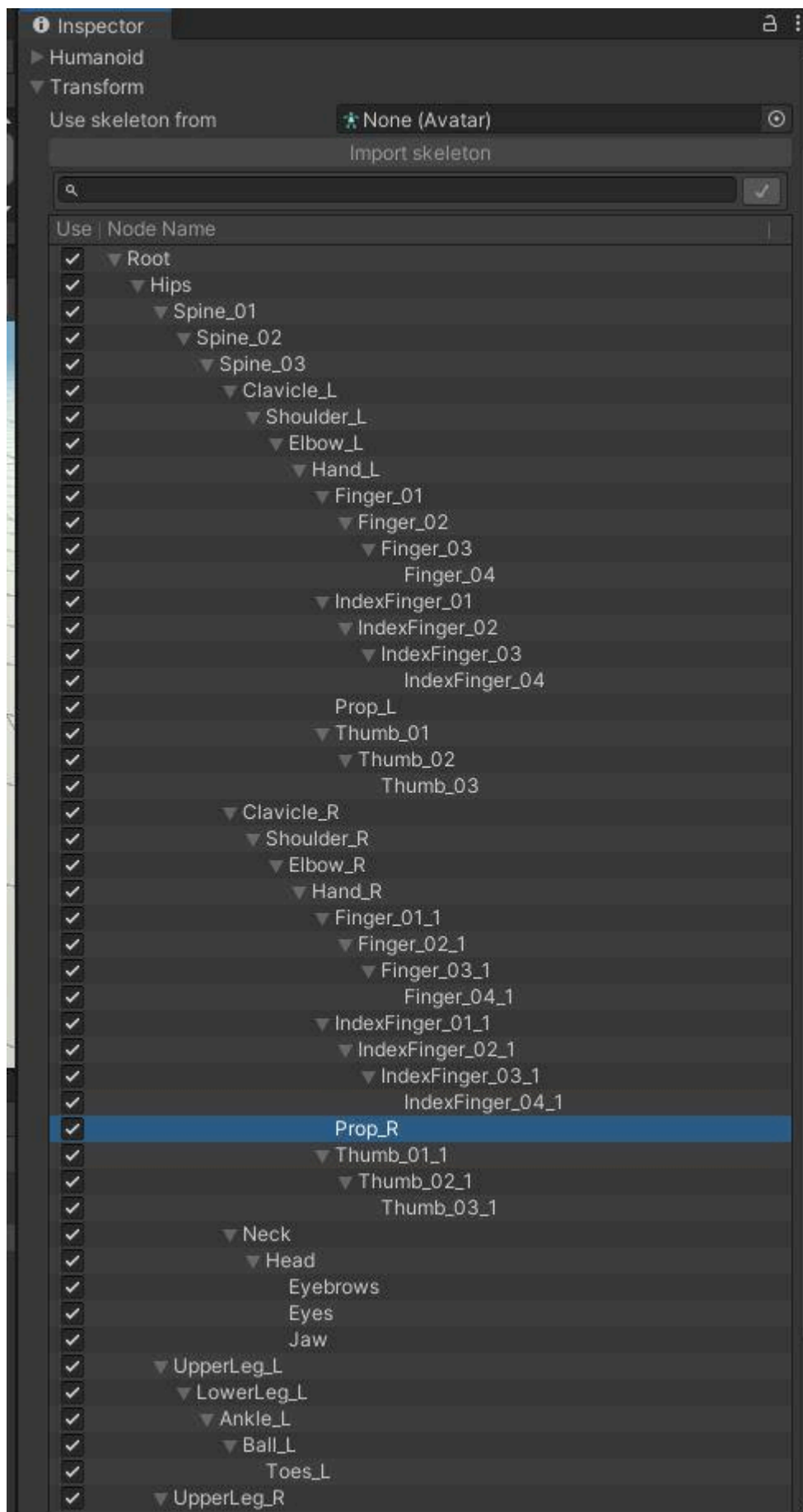
Synty Characters

In order to achieve more natural weapon motion, adjust the way a weapon is gripped mid-animation, and generally allow more freedom for how a weapon can behave, new **Prop_L** and **Prop_R** bones have been added to the Synty character under the **Hand_L** and **Hand_R** bones respectively.

Example of different hand grips, where the sword is oriented differently in relation to the hand:



As Unity's Humanoid Avatar system does not inherently account for this extra prop bone, the setup uses an AvatarMask asset (with the Prop bones ticked 'on' to be included in the mask) on the combat animation clips. This allows the animations to correctly play animation that has been keyframed on the Prop bones themselves.

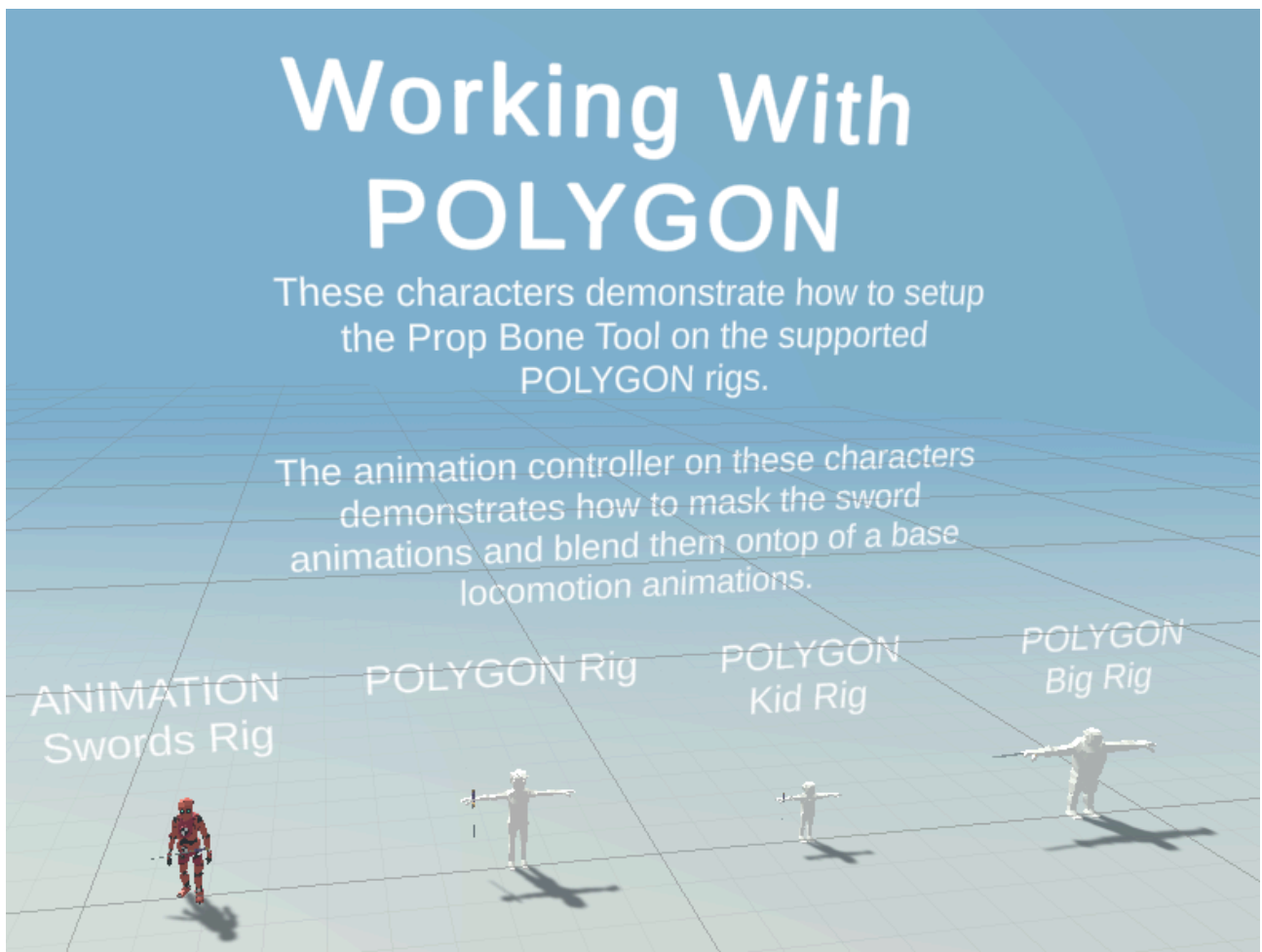


For this reason, all animation clips in the Sword Combat pack use the AvatarMask asset: **Mask_PolygonSyntyCharacter** located inside **Assets\Synty\AnimationSwordCombat\Meshes**

Prop Bone Binder Tool

For skeletal meshes that do not include a prop bone (e.g. older Synty characters, or non-Synty characters), the sword pack comes with a tool to add 'virtual' prop bones to any character so the sword animation can still be correctly played.

The virtual prop bone binder tool works with the standard POLYGON character Rig, POLYGON Kid Rig, POLYGON Big Rig. Demonstrations of these characters and how they are set up can be found in the ANIMATION Sword Combat gallery scene.



Setting up your character

Setting up your character only involves a couple of steps:

1. Select your character. Select either an instance in the scene or an open prefab.

The tool will not function when run on a prefab asset selected in the project view.

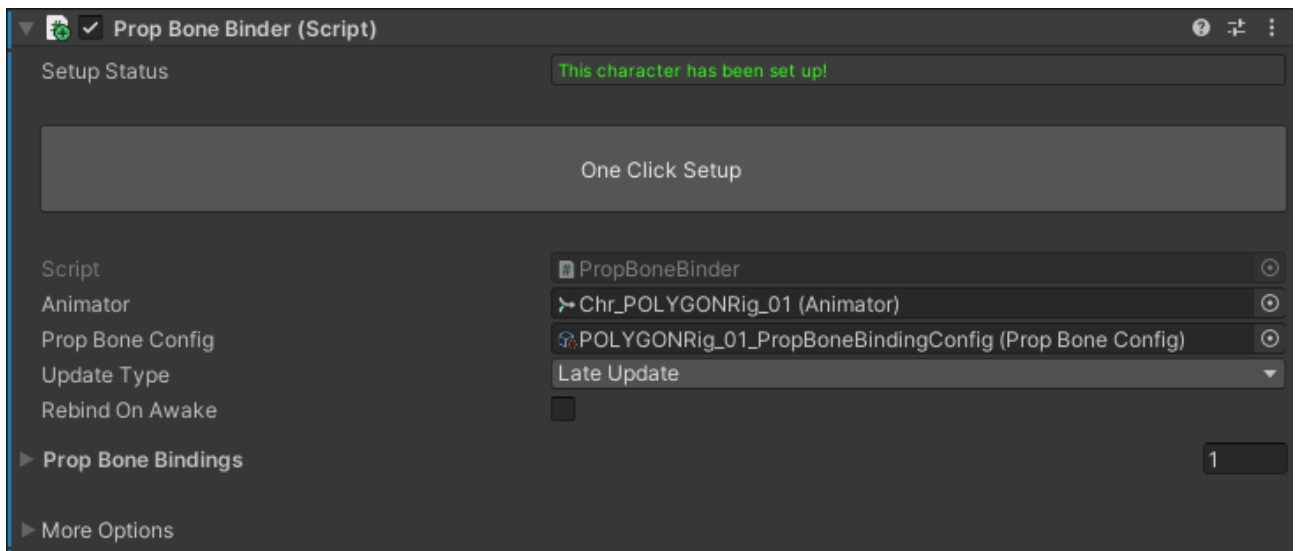
2. Use the menu item **Synty/Tools/Animation/Setup Prop Bones**
3. Attach your weapon to the newly created **Prop_R_Socket** game object located under **Hand_R** in the skeleton.



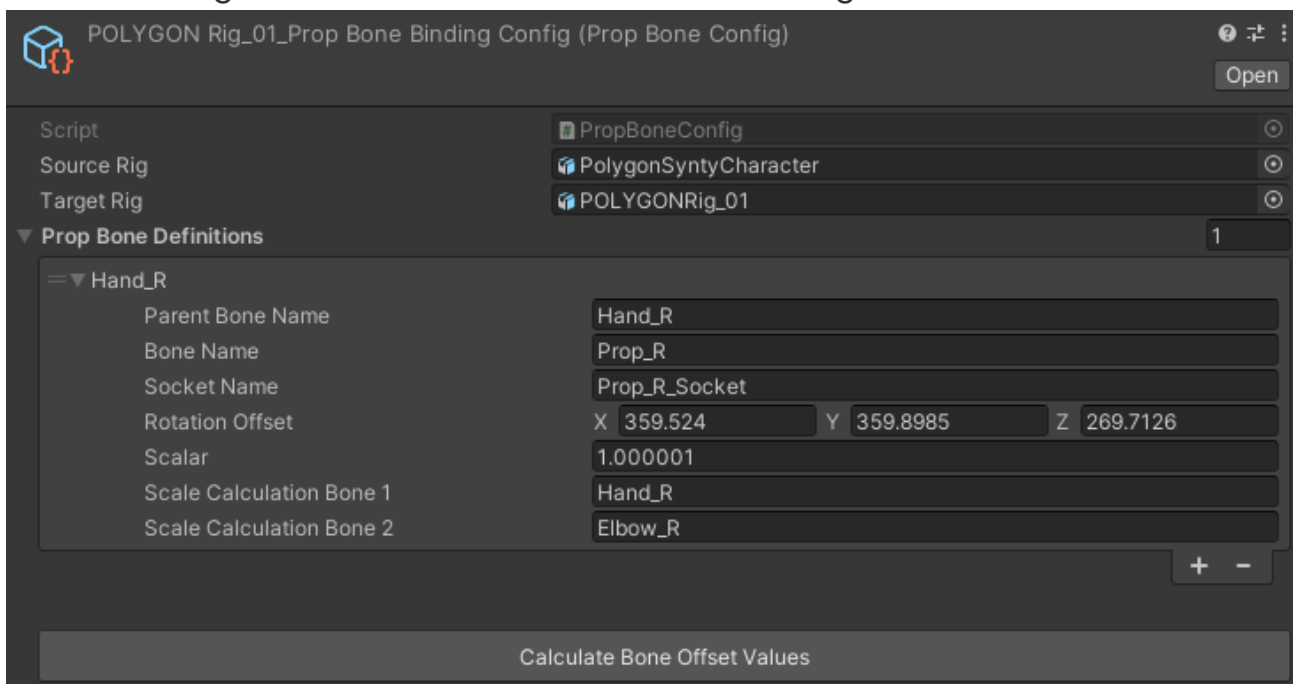
That's it!

Tip: To update multiple characters at once drag them all into the scene, select their root nodes and run the tool.

After following the steps above you should see the Prop Bone Binder component on the game object that your character's Animator component is on. The green status message denotes that the character is correctly configured for the animations to work.



You will find a **PropBoneBindingConfig** file in your project. This file details the location of the prop bone and the offset orientation and scale between your character's rig and the ANIMATION Sword Combat Rig.



The offset values are calculated automatically on set up but if needed you can edit these values manually when setting up your own custom rigs.

To calculate the values automatically make sure:

1. Set the **source rig** and **target rig** references
2. Make sure the **source rig** and **target rig** are in T Pose
3. Set names of all the bones in the **Prop Bone Definitions** list

4. Click the **Calculate Bone Offset Values** button

Source Rig: The rig the animations are keyed for. In this case it should be the ANIMATION Sword Combat Rig (PolygonSyntyCharacter)

Target Rig: The rig that you intend to play the animations on. This is your game's character model or prefab.

Parent Bone Name: The bone to create the virtual prop bone under.

Bone Name: The name of the virtual prop bone. The tool will generate this node.

Socket Name: The name of the object to attach your props to. The tool will generate this node.

Rotation Offset: Rotation offset used to compensate for differences in orientation of the parent bone between the **source rig** and the **target rig**

Scalar: Value used to compensate for differences in size between the **source rig** and the **target rig**. A value of 1 = **target rig** is the same scale as the reference rig, a value of 2 = **target rig** is twice the size as the reference rig.

Scale Calculation Bone 1 / Scale Calculation Bone 2: The bones used to determine the difference in size between the **source rig** and the **target rig**. These bones need to exist in both rigs for the automatic calculation of **Scalar** to work.

Prop Bone Binder Tool Troubleshooting

My weapon is not oriented correctly

- There could be a couple of reasons for this:
 - a. You need to find the correct orientation to attach your weapon to the **Prop_R_Socket** game object. The sword that comes with the swords pack is designed to be attached with 0,0,0 rotation. You can use this sword as a guide to ensure you've attached your weapon in the correct orientation.
 - b. The values in the Prop Bone Binding Config file are not set up correctly. POLYGON rigs should be set up correctly by default but these values can be manually set if need be.

The status does not say "This character has been set up":

- There are many reasons why a character may not be set up correctly. Try clicking the **One click setup** button and see if that resolves the issue, if not check the console for more details about what could have gone wrong.

I have a few different character rigs in my game. Does the tool work in this case?

- Yes! You will need a Prop Bone Binding Config file for each rig you are using. And ensure that each character references the correct config file. The tool should automatically set up the characters and create new config files when it needs to. simply select all your characters and use the menu options **Synty/Tools/Animation/Setup Prop Bones**

Removing the Prop Bone Binder Tool

You may find you need to remove the bones created by the Prop Bone Binding Tool. The tool provides you with the ability to do this. Located on the Prop Bone Binder component, under **More Options** is the **Reset** button. Click **Reset** and all objects created by the tool will be destroyed. Any props attached to the socket will be reparented to the **Hand_R** bone. After successfully removing the bones you can remove the Prop Bone Binder component from your character.

Prop Bone with Non-POLYGON Rigs

For the prop bone to animate the hierarchy paths needs to be a match for the animation property. The **Prop_R** path is:

Root/Hips/Spine_01/Spine_02/Spine_03/Clavicle_R/Shoulder_R/Elbow_R/Hand_R/Prop_R

If your rig is not named like this then it will not animate even with the virtual prop bone added.

Using Animations Without Prop Bones

It is possible to simply parent a weapon to the **Hand_R** bone, and to use this animation set on any Humanoid Unity character, however it is recommended to use a Synty character with the extra prop bone to take advantage of the fluid sword animations. Some animations in particular, such as stabbing attacks, feature the sword changing orientation within the hand, and will not look correct without a sword parented to the **Prop_R** bone.

Those animations most reliant on the Prop bone are:

- **A_Attack_LightCombo01C_Sword**
- **A_Attack_LightFencing01_Sword**
- **A_Attack_HeavyStab01_Sword**
- **A_Death_R_01_Sword**

- A_Draw_Sword
- A_Sheathe_Sword
- A_Idle_Flourish01_Sword

4. Quick Start

Gallery Scene

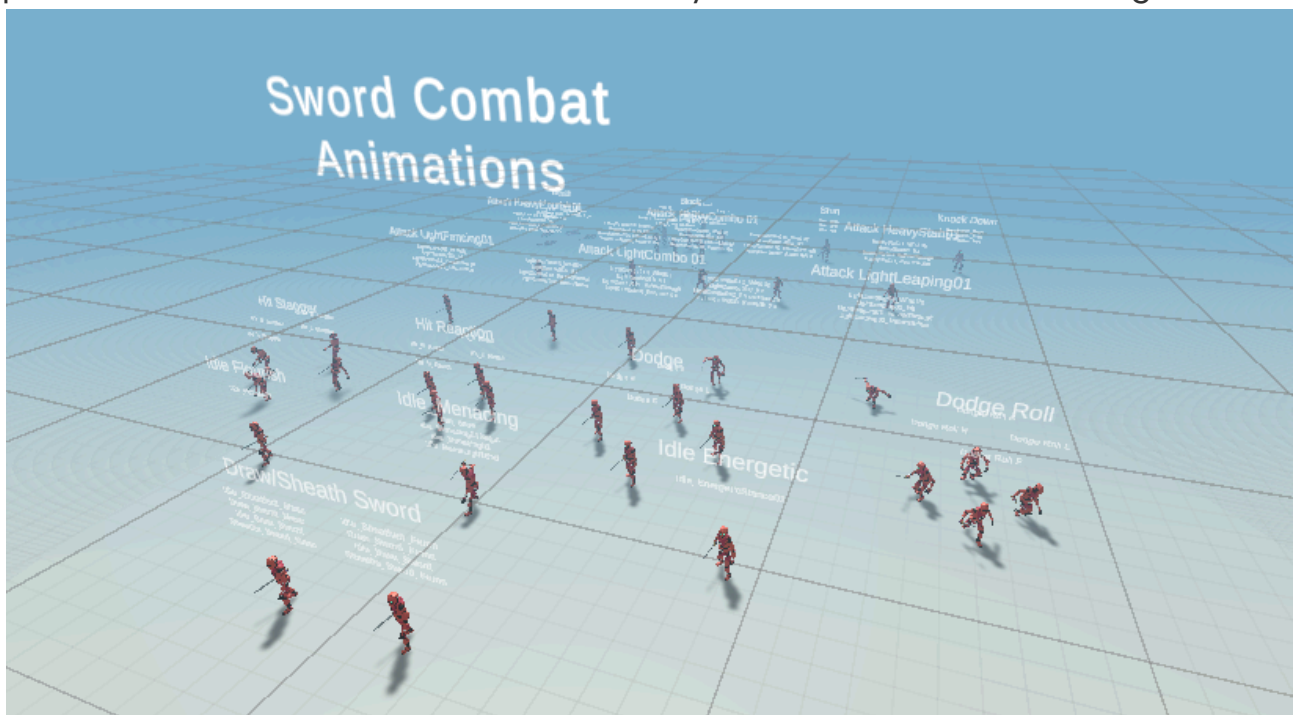
We have provided a gallery scene as an asset showcase of what is in this product. You will find this in:

Assets/Synty/AnimationSwordCombat/Samples/Scenes.

This scene demonstrates the various animations in this pack as a virtual gallery/library so users can break down how the animations work in isolation, and to understand further how they work in tandem:

- Labelled groups of animations to quickly track down specific animations
- Looped to see timing/length of animation

There are also several sword duels choreographed on Timelines using the pack's animations to demonstrate how they could flow and be used together.



Build your own

The following example will walk you through a typical use case of Sword Combat as a means to guide your experimentation further to find your own workflow. This assumes you have installed the pack correctly and are starting from a new scene.

Applying Animations to Characters

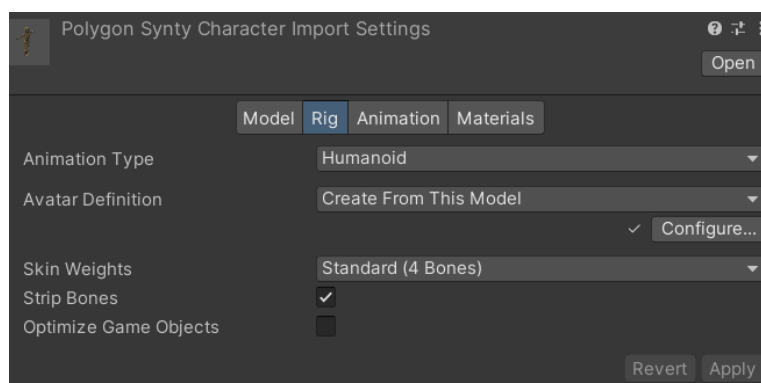
In Unity, a developer can import the animation package into a new scene and reference the **PolygonSyntyCharacterAvatar** to retarget the animation to their own character. This character can be a Synty character or a completely different biped, however the animations will work the best with Synty characters as the animations were created with their proportions in mind.

Import Characters:

- Import the Sword Combat package into the project for access to the animations and the **PolygonSyntyCharacterAvatar**
- Import your character that will be the target for Base Locomotion animations

Create a Humanoid Avatar on new character:

- On the new character, create a new Avatar and configure bone mappings



Creating a new Avatar

Animation Type:

- Navigate to the Inspector window in Unity of the Character.
- Click on the **Rig** tab to access the Avatar Configuration tab.

Avatar Definition:

- Set the Animation Type to **Humanoid**
- Set the Avatar Definition to **Create From This Model** to generate a humanoid avatar based on the character's rig.

Modifying existing Avatar

Access Avatar Configuration:

- Click on **Configure...** to access the Avatar Configuration tab.

Bone Mapping:

- Review and adjust bone mappings to ensure precise alignment with the character's skeletal structure.

Preview and Apply:

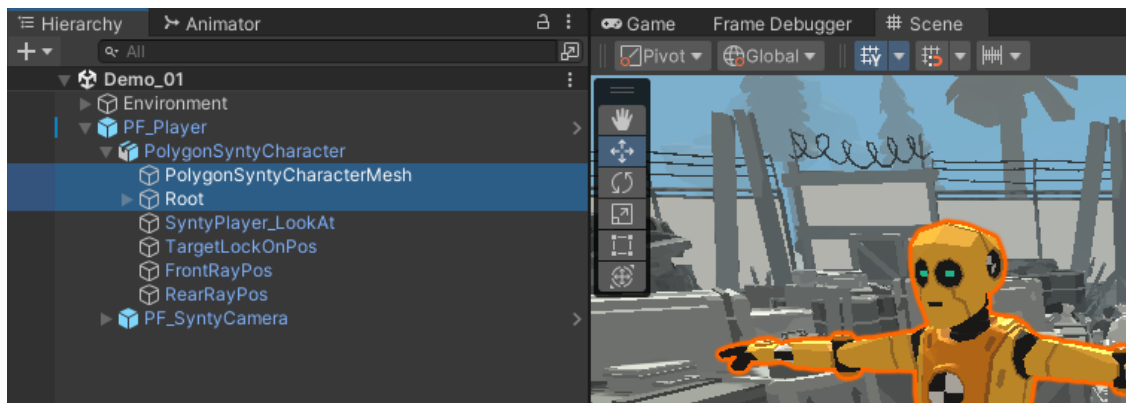
- Check for any errors that occur or if you are using the same bone in two definitions in the skeletal mapping
- Apply changes to update the avatar properties.

Now animations from Sword Combat can be applied to an Animator or a Timeline using the new character, and will retarget automatically due to the Humanoid setup.

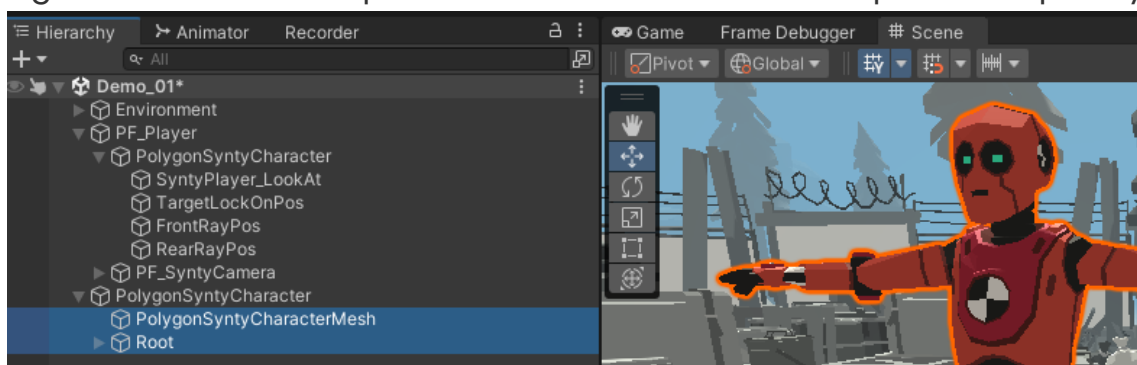
Integrating Synty Sword Combat with Synty Base Locomotion

Synty's Sword Combat animations are designed to integrate easily with existing Synty animation packages. If you already own Base Locomotion, you can get the **PF_Player** prefab set up to work with Sword Combat animations.

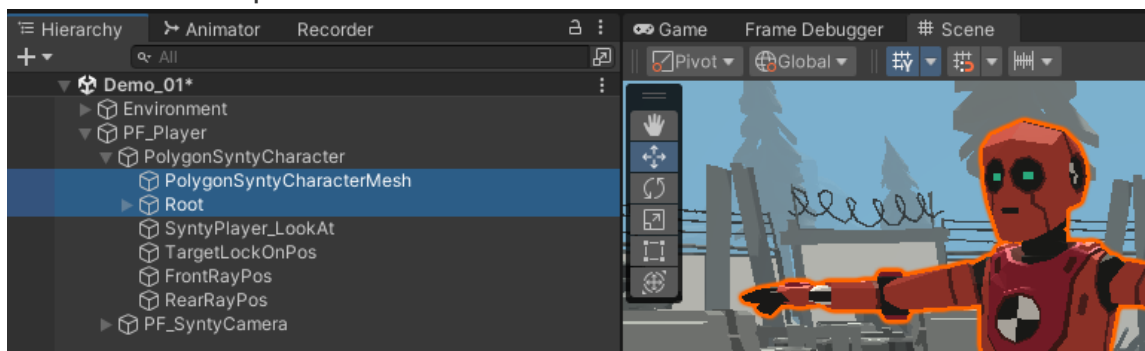
1. Either use the Section 4 guide on the **Prop Bone Binder Tool**, or replace the BaseLocomotion character with the Sword Combat character (which includes the prop bone) using these steps:
2. Open a scene in a project containing Base Locomotion and import the Sword Combat package (see Section 2 of this guide).
3. Select the **PF_Player** object in the scene hierarchy, right-click and choose 'Prefabs' > 'Unpack Completely'
4. Select and delete the **PolygonSyntyCharacterMesh** and **Root** nodes beneath the **PolygonSyntyCharacter** node.



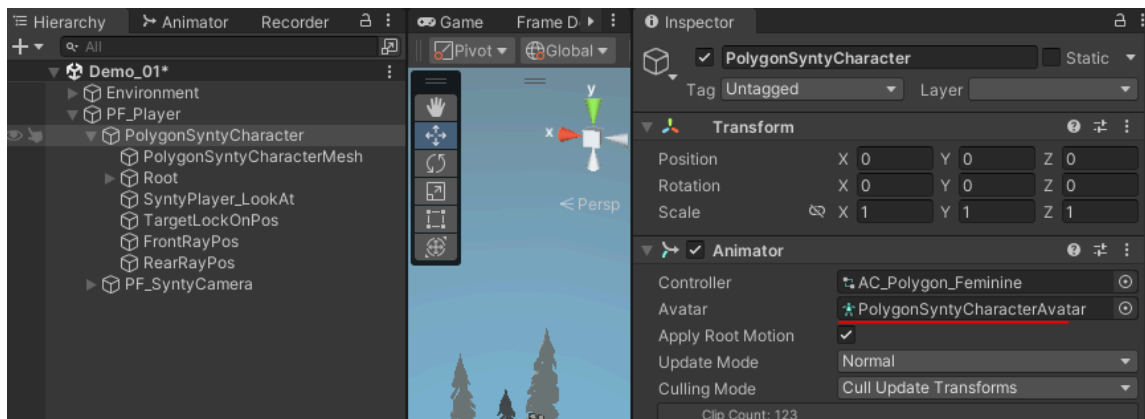
5. Drag a new **PolygonSyntyCharacter** from the **SwordCombat** package into the scene.
6. Right click on this new prefab and select 'Prefabs' > 'Unpack Completely'.



7. Now drag the two nodes **PolygonSyntyCharacterMesh** and **Root** from the new Sword Combat Prefab into the place of the two deleted nodes under the **PolygonSyntyCharacter** inside **PF_Player**. You can delete the rest of the Sword Combat prefab's nodes.



8. On the node **PolygonSyntyCharacter** (below **PF_Player**), update the Avatar on the Animator component to use the **PolygonSyntyCharacter** Avatar from the Sword Combat pack.

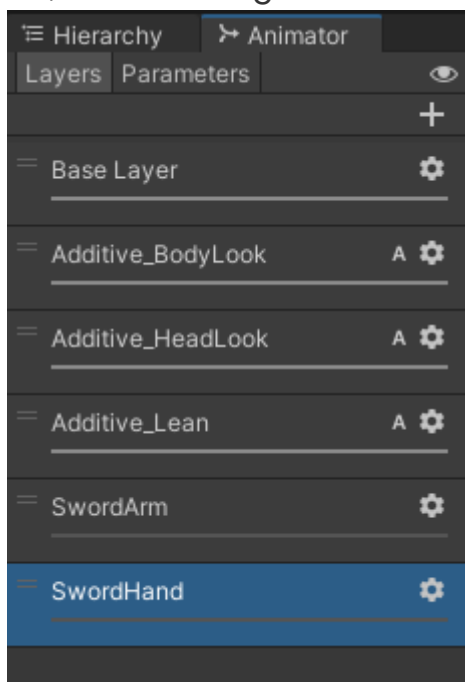


9. From here you can begin to add new functionality to the existing controller, e.g. new States in the Animator for attacks, dodges etc.

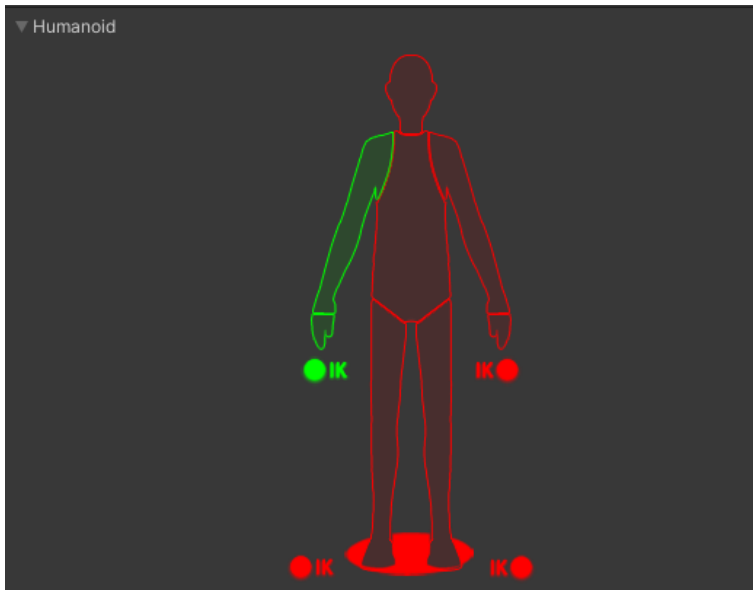
Adding a sword to existing Synty Base Locomotion Animations

A simple way to add a 'sword-in-hand' pose to Synty's Base Locomotion pack of animations and controller is to use Animation Layers in Unity.

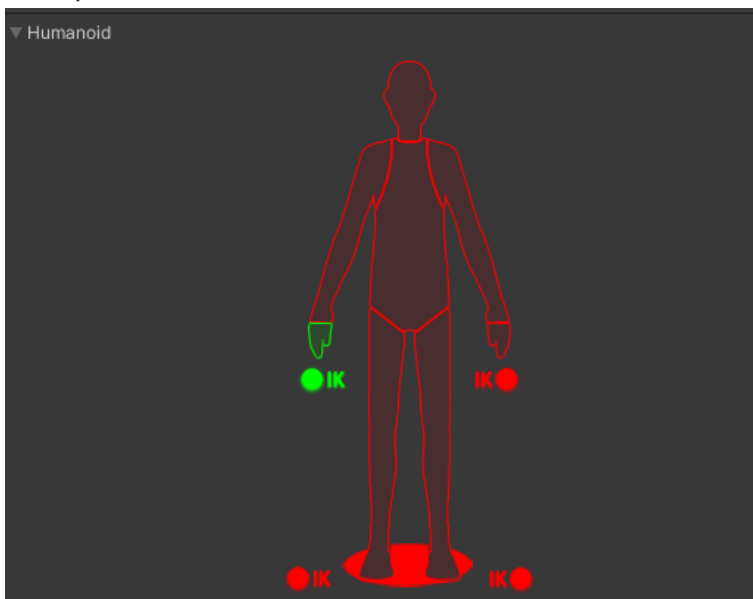
1. Select the BaseLocomotion pack's Animation Controller you intend to use (e.g. **AC_Polygon_Feminine**)
2. In the **Animator** tab, create two new Animation Layers called **SwordArm** and **SwordHand**
3. Set both layers to 'Override' in the 'Blending' option.
4. Set the weight on the **SwordArm** layer to a value somewhere around 0.7 or 0.6, and the weight on the **SwordHand** layer to 1.



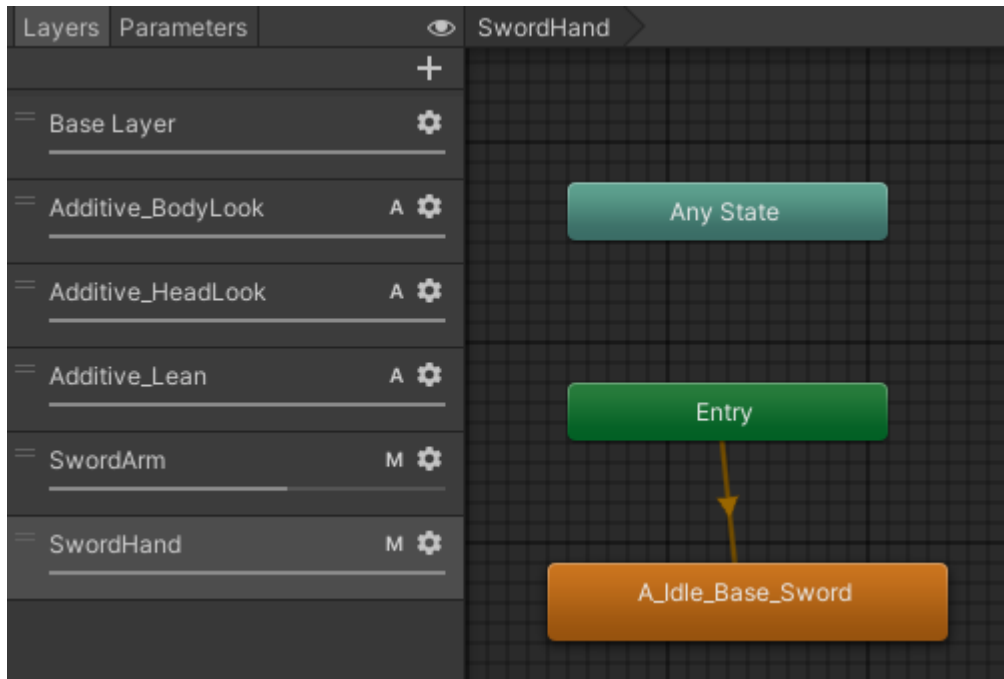
5. Create an AvatarMask asset called **Mask_Arm_R**. In the inspector, under the **Humanoid** section, deselect all except for the right-arm components.



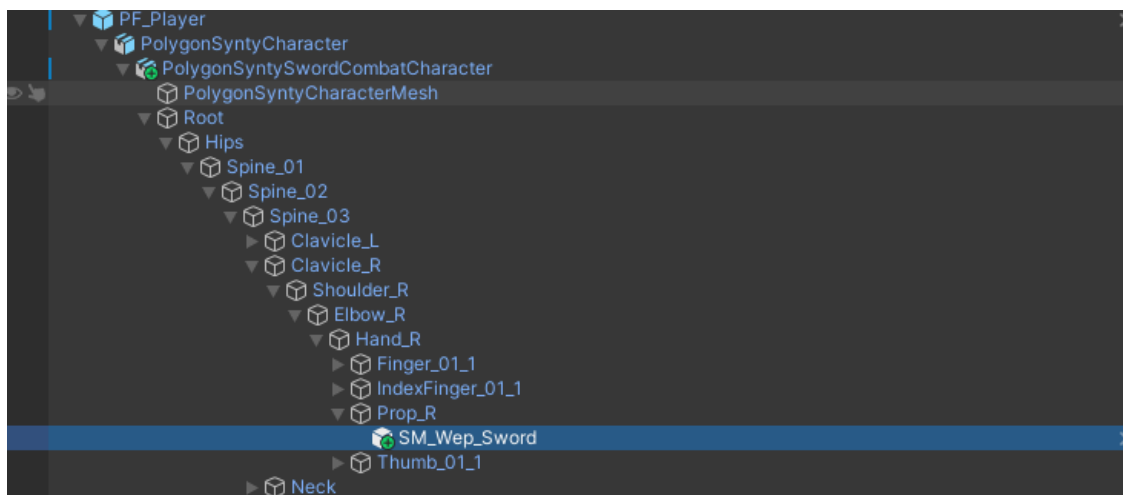
6. Create a second AvatarMask asset, name it **Mask_Hand_R** and repeat the same steps as for the other mask, but this time select only the right hand component.



7. Now back in the Animator, apply **Mask_Arm_R** to the **SwordArm** animation layer, and **Mask_Hand_R** to the **SwordHand** animation layer.
8. Navigate to **Assets/Synty/AnimationSwordCombat/Animations/Polygon/Idle/Base/** and for both layers, drag and drop the default sword combat idle pose **A_Idle_Base_Sword** into the Animator graph so it becomes the default animation for that layer.



9. Drag a sword prop into the scene and parent it under the **Prop_R** bone on the character.



10. The Base Locomotion character should now hold the sword correctly. The reason for having two layers is so that the hand will grip the sword with a 100% override to have the correct pose on the fingers and prop bone, while the arm will have a more subtle 60-70% override, still inheriting 30-40% of the Base Locomotion animation to look more natural.



11. For the Animator to play animations from the Sword Combat pack itself (attacks, blocks etc), the layer overrides would need to be disabled so the full body animation plays normally.

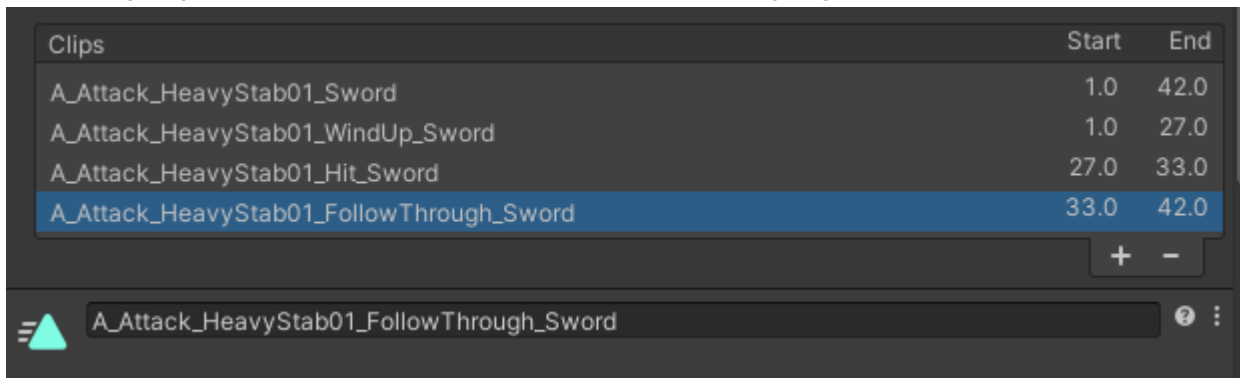
5. Combat Animations

Attack Animations

The 'Attack' animations in this combat pack are broken up into clips that designate specific parts of an attack. There is one clip that plays the animation in its entirety, as well as the same animation broken into three parts. These parts are:

- **WindUp**
 - This is the anticipation portion of an attack, it can telegraph what an enemy is about to do, or provide a window vulnerability when the character is in the process of attacking.
- **Hit**
 - This is intended as the brief window when the sword is active in dealing damage/colliding with an enemy.
- **FollowThrough**
 - The follow-through is another window of potential vulnerability when the character is recovering from their attack.

As these three parts are all clips on a single Animation, each part can be easily reconfigured non-destructively. Clip timing/frames can be adjusted, or even extra clips/parts added or removed based on a project's combat needs.



The screenshot shows a software interface with a table of animation clips. The table has three columns: 'Clips', 'Start', and 'End'. There are four rows of data. The fourth row is highlighted in blue. Below the table, there is a search bar containing the text 'A_Attack_HeavyStab01_FollowThrough_Sword' and a play button icon.

Clips	Start	End
A_Attack_HeavyStab01_Sword	1.0	42.0
A_Attack_HeavyStab01_WindUp_Sword	1.0	27.0
A_Attack_HeavyStab01_Hit_Sword	27.0	33.0
A_Attack_HeavyStab01_FollowThrough_Sword	33.0	42.0

All attacks have an additional **ReturnToIdle** animation that would usually play directly after the attack's **FollowThrough** section finishes, to return the character to its default idle combat pose.

Because some attacks (combos) can branch directly into another attack without returning to idle, for consistency the **ReturnToIdle** part is always a separate animation file rather than a clip, across all attacks.

Heavy and Light Attacks

The Sword Combat animation pack uses 'Light' and 'Heavy' as definitions that describe one-handed and two-handed attacks respectively. In general, Heavy attacks are slower, Light attacks are faster.

These different styles of attack could be harnessed in a myriad of creative ways to suit many types of project.

An example: In a project intending to create a risk vs reward combat scenario, Heavy attacks could be set up to deal greater damage. This could be a trade-off against the attacks taking longer to complete (leaving the player vulnerable to counterattacks), as well as using more stamina.

Attack Combos

The animation pack comes with two attack combos (Heavy and Light). Combos are a series of attacks that chain into one another (faster than if returning to the idle pose after each attack). Often this combo chain would be

set up to require a player input to be made within a particular timing window for each subsequent attack, and if successful, the next part of the combo playing.

Each combo in this Sword Combat pack has three separate attacks, named as A, B, and C.

Each of the 'A' and 'B' attacks in a combo can either:

- Continue into the next 'WindUp' phase of the subsequent attack.
- Return to the Idle pose.

Example: **A_Attack_LightCombo01A_Sword** is the first attack in the three-part light combo.

The end pose of **A_Attack_LightCombo01A_FollowThrough_Sword** is designed to transition seamlessly into both:

- **A_Attack_LightCombo01B_WindUp_Sword**
- **A_Attack_LightCombo01A_ReturnToIdle_Sword**

Leaping Attack

The attack **A_Attack_LightLeaping01_Sword** is designed to be versatile in how it can be implemented. While in its full length it starts from Idle, and features a short run and jump, it could equally blend directly into the attacking part from a 'falling' state or regular locomotion jump etc.

To add versatility in how it can be used, the attack comes in four different forms:

- Non-RootMotion
- Full RootMotion
- Horizontal RootMotion only
- Vertical RootMotion only

The non-RootMotion version allows all the height and distance of the jump to be added programmatically, or inherit these velocities from the controller.

Additionally, the root motion versions can be used in other ways. For example, if you wanted to include the leaping attack on a controller without using either its horizontal or vertical root motion at all, but still wanting to see the character rise to the full height of the animated jump, then the clip

A_Attack_LightLeaping01_RootMotionVertical_Sword settings could be modified to do this. In the inspector:

- Select the desired animation in the project panel
- In the inspector, select the 'Animation' tab.
- Scroll down to the **Motion** section, and set **RootMotion Node** to **None**
- For all **Root Transform** settings, set **Based Upon:** to **Original** and **Bake Into Pose** ticked on.

Blocking and Parrying

The animation pack has a variety of options for blocking during combat. There is a looping 'Block' stance, with separate transition animations from and back to the Idle pose.

From this **A_Block_Loop_Sword** animation, there are several animations/scenarios the character can seamlessly transition to:

- **A_Parry_L_Sword**
- **A_Parry_F_Sword**
- **A_Parry_R_Sword**
- **A_Parry_Break_Sword**

These animations provide some directional options for successfully blocking an incoming attack, while the **Parry_Break** could be used in many situations such as an unsuccessful or imperfect block.

The **Parry_F** animation has yet more options available, and can transition to:

- **A_Parry_F_ReturnToBlock_Sword**
- **A_Parry_F_CounterShove_Sword**
- **A_Parry_F_PommelStrike_Sword**

These provide a range of counterattack options, all stemming from the same end-pose of the **Parry_F** animation.

Sheathing/Unsheathing the sword

Most animations in this pack start and end in the **A_Idle_Base_Sword** pose, however the pack includes some transitions to and from a 'sheathed sword' pose that matches the idle pose used in Synty's Base Locomotion pack.

There are two variants: a Masculine and a Feminine version. For each variant,

there is an idle loop, a 'draw-sword' animation, and a 'sheathe-sword' animation.

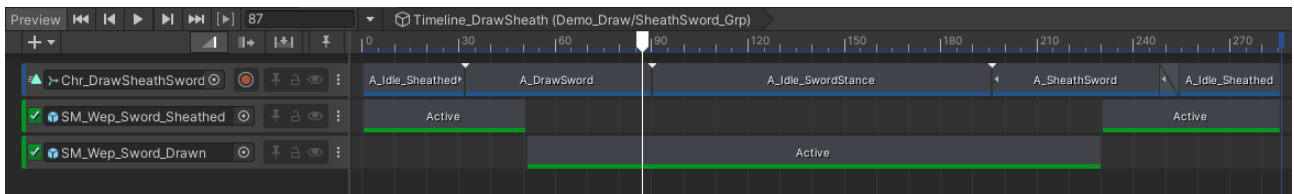
For a 'sheathed' weapon position, it is recommended to either reparent the weapon prop, or use a second weapon object (placed at the desired position on the hips and parented to the **Hips** bone) that has its visibility toggled during 'sheathe' or 'unsheathe' animations.

This is because the **Prop_R** bone is a child of the **Hand_R** bone and the arm hierarchy. Even if the **Prop_R** bone is animated to remain static and attached to the hips, if its parent hand bone or arm hierarchy makes any motion itself, then the keyframes on the Prop bone will effectively be counter-animating against its parent in order to itself not move, possibly causing motion jitter as it interpolates.



A setup exists in the Gallery scene demonstrating the transition from a sword parented to the **Prop_R** bone to a sword parented to **Hips**. The timeline **Timeline_DrawSheathe** on the **Demo_Draw/SheatheSword_Grp** group contains transitions during the animations **A_Draw_Sword** and **A_Sheathe_Sword**, swapping

between two swords to avoid motion jitter using an 'Active' track to hide and unhide each sword.



Equally, a visibility toggle achieving the same result could be triggered by an Animation Event to work on a controllable player, character etc.

6. Tips for working with Sword Combat

Animation Files

Optimizing Animation Performance

Unity has numerous options to help optimize performance when using animations. These options are within the animation tab for each fbx animation asset.

Import Settings:

- Use the **Rig** tab to configure Avatar Definition and Animation Type settings.
- Enable **Optimize Game Objects** to remove redundant Transform components.
- Consider enabling **Import Blend Shapes** if needed.

Animation Compression:

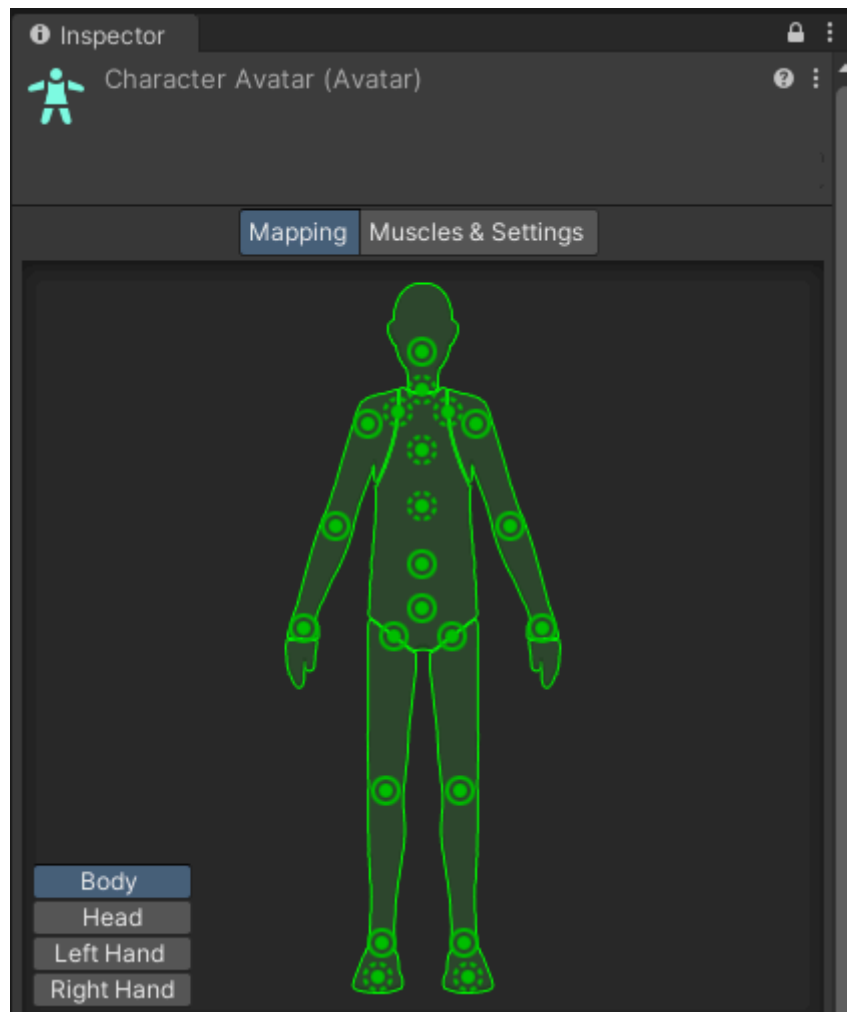
- Utilize Unity's animation compression options.
- Use **Optimal** for keyframe reduction without sacrificing visual quality.
- Experiment with different compression ratios to find the right balance between performance and quality.

Remove Scale Curves:

- This setting removes constant animation curves with values identical to the object initial scale value.
- If not needed, (i.e. if there is no animated scaling anywhere in the animations) consider applying this option.

- Scale curves can impact performance, and removing them can streamline the animation data.

Mecanim Humanoid Character Avatar



In Unity, the Mecanim Humanoid Character Avatar serves as a fundamental framework for bipedal character animation. Essentially, an avatar in this context is a digital representation of a character's skeletal structure and body proportions. To access the avatar of a character you can find the mesh that the avatar definition is created from, in the case of the Synty pack, the Avatar is nested in the **Synty\AnimationSwordCombat\Meshes** as the **PolygonSyntyCharacterAvatar**.

Here's a breakdown to clarify its purpose for users:

Humanoid Structure

Unity's mecanim Humanoid Character Avatar adheres to a bipedal structure, aligning with the standard anatomy of human characters.

Compatibility Across Characters:

Designed to be universally compatible, the avatar allows users to apply animations seamlessly to a variety of humanoid characters, streamlining the animation process.

Other character models on different bipedal rigs can be added to the project, with their own Avatar set up to allow the animations from this pack and others to be applied to them, bypassing the issue of compatibility.

Configurable and Adaptable:

Users can configure their own avatars by adjusting the bone mappings and aligning them with Unity's Humanoid Avatar configuration.

Adjusting Avatar Properties

In Unity, within the rig tab of an asset you are able to define the skeleton type to one of the following, **Humanoid**, **Generic** or **Legacy**.

Humanoid is used when working with humanoid characters. It provides a standardized bone structure that makes it easier to work with humanoid animations, retargeting, and blending. The Humanoid rig is particularly useful when using Unity's Mecanim animation system.

Generic is a more flexible option that doesn't adhere to the humanoid bone structure. It allows for more custom setups but may not be as compatible with certain features like retargeting humanoid animations.

Legacy is used for the older animation system in Unity. It's not recommended for new projects, as Unity has shifted its focus to Mecanim and the Humanoid rig. The content within this pack is set up to be used with the Humanoid skeleton type, to leverage the aforementioned retargeting and blending with the Mecanim animation system.

7. Naming conventions

A_	Animation
AC_	Animation Controller
M_	Material
PM_	Physics Material
SK_	Skeletal Mesh
SM_	Static Mesh
T_	Texture

8. Terms of use

The full terms of the End User License Agreement (EULA) apply and can be found at <https://syntystore.com/pages/end-user-licence-agreement>.

This is a summary of the license for the Synty Sword Combat software provided by Synty Studios Limited. This summary is for convenience only and is not legally binding.

Key Points

1. **License Grant:** When you purchase an Asset, you are granted a license to use the Asset subject to the terms of the EULA. All intellectual property rights in the Asset remain with Synty Studios Limited.
2. **Use of Asset:** You are entitled to incorporate the Asset into Products under your direct control, and into promotional materials for those Products. You can adapt the Asset for these purposes.
3. **Restrictions:** There are important restrictions on your use of the Asset. For example, you cannot use the Asset for Non-Fungible Tokens (NFTs), in Blockchain projects, Metaverse-related content, or with Generative AI Programs. You also cannot share or redistribute the Asset outside your team.
4. **Team Size and Seats:** Your license includes a limited number of seats for your team. If your team grows, you must purchase additional licenses.
5. **Unity Asset Store Purchases:** If you have purchased the Asset through the Unity Asset Store, the terms of the Unity Asset Store EULA also apply. You must consult the Unity Asset Store EULA for the full terms applicable to your purchase from the Unity Asset Store.

6. **Support and Source Files:** Support is provided at our discretion. You must not share source files of any Assets outside your team.
7. **Termination:** Your license can be terminated if you breach the EULA and fail to remedy the breach after notice from us.
8. **Warranties and Liability:** The Assets are provided "as is" without warranties of any kind, and our liability is limited.

For all the terms and conditions of the license, please read the full EULA available at <https://syntystore.com/pages/end-user-licence-agreement>. By using this Asset, you agree to be bound by the terms of the EULA.

9. Glossary

Term	Definition & Context
Animation Controller	<i>An Animation Controller in Unity is a system that manages the state machine for character animations. It defines how animations transition from one state to another based on conditions.</i>
Animation Layer	<i>Animation Layers in Unity are used for managing complex state machines for different body parts. For example, having a head look layer for turning the head with the camera direction, or having a lower-body layer for walking and jumping and an upper-body layer for throwing objects or shooting.</i>
Animation State	<i>Animation States represent individual animation clips or motions within an Animation Controller. These states define the specific animations a character can be in.</i>
Blend Tree	<i>A Blend Tree is a Unity mechanism that allows smooth transitions between animations by blending multiple animations based on input parameters, such as Speed or Direction.</i>
Character Avatar	<i>In Unity's Mecanim system, the Character Avatar refers to the digital representation of a character's skeletal structure, including bone hierarchy and rigging. It serves as the foundation for applying animations and controlling the character's movements within the game.</i>
Mecanim	<i>Mecanim is Unity's animation system, encompassing the Animation Controller, Animator component, and state machine. It provides a visual interface for designing complex character animations and facilitates the integration of character avatars, animation clips, and transitions.</i>
State Machine	<i>In the context of Unity's Animation Controller, a State Machine is a computational model that defines the various states a character or object can inhabit. It manages transitions between states based on specified conditions, influencing the character's behavior and animations.</i>

© 2024 Synty Studios Limited

