

**NOTIOM MOCAP LIVE HANDBOOK**

NoitomMocapLive\_Unity\_v1.1.0.0

Noitom Technology Co.Ltd：<http://noitom.com.cn/>

Dec, 10th, 2021

# TABLE OF CONTENTS

[TABLE OF CONTENTS 2](#_Toc89860702)

[1. Introduction 3](#_Toc89860703)

[2. Import Package 4](#_Toc89860704)

[3. Structure Overview 5](#_Toc89860705)

[3.1 Example Scenes 5](#_Toc89860706)

[3.2 Core Scripts Overview 5](#_Toc89860707)

[3.3 Public Variables of Neuronanimatorinstance.cs 7](#_Toc89860708)

[4. Data streaming setup in Axis 8](#_Toc89860709)

[4.1 Axis Neuron/Axis Neuron Pro Setup 8](#_Toc89860710)

[4.2 Axis Studio Setup 8](#_Toc89860711)

[5. Neuron Skeleton Tools 10](#_Toc89860712)

[6. How to configure a character model 12](#_Toc89860713)

[7. How to Apply Real-time Data from Axis 14](#_Toc89860714)

[8. HMD INTEGRATION 17](#_Toc89860715)

[9. Setup in AHM/Alice 18](#_Toc89860716)

[9.1 AHM/Alice Data Streaming Setup 18](#_Toc89860717)

[9.2 AHM/Alice Device Setup 18](#_Toc89860718)

[10. How to Apply AHM/Alice Device Data 20](#_Toc89860719)

[11. How to Apply FBX file Data from Axis 21](#_Toc89860720)

[11.1 Humanoid Animation type. 21](#_Toc89860721)

[11.2 Generic Animation Type 24](#_Toc89860722)

[12. APPENDIX A: SKELETON MAPPING 25](#_Toc89860723)

[12.1 PN Studio skeleton structure mapping 25](#_Toc89860724)

[12.2 PN/PN Pro skeleton structure mapping 26](#_Toc89860725)

[13. APPENDIX B: SKELETON BONES 27](#_Toc89860726)

[14. APPENDIX C: BINARY DATA SEQUENCE 29](#_Toc89860727)

# 1. Introduction

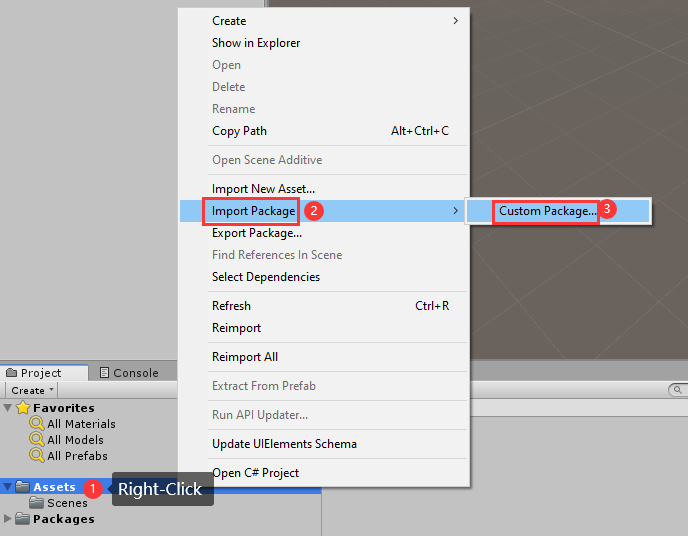
This document should help you get familiar with real-time motion capture data reading and how to use fbx file exported from Axis Neuron &Axis Studio inside the game engine Unity 3D. Axis Neuron &Axis Studio not only allows the export of motion capture data but also be able to stream motion capture data in real-time to third party applications, making the data available to drive characters and any GameObject in animation.

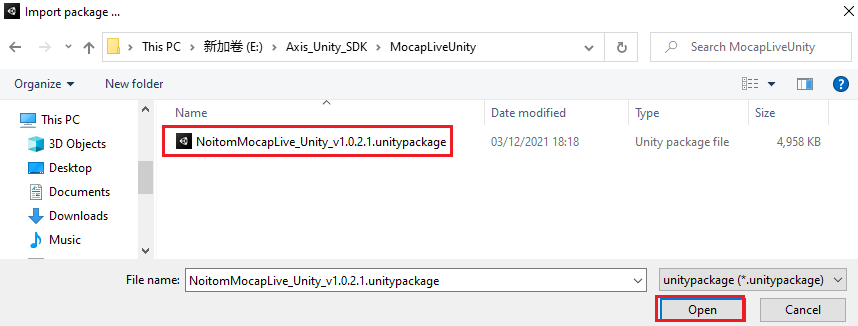
The data stream to the game engine is identical regardless if you use recorded or live motion data. This allows you to record certain actions and use them to test before putting on the full system and doing a live test. You can have multiple actors at the same time inside Axis and have them all streamed into Unity. The data stream is based on the BVH structure including header information and body dimension data is available from Axis via a command interface. Because motion-data is streamed over network the computer running Axis Neuron &Axis Studio doesn’t necessarily have to be the same computer running Unity.

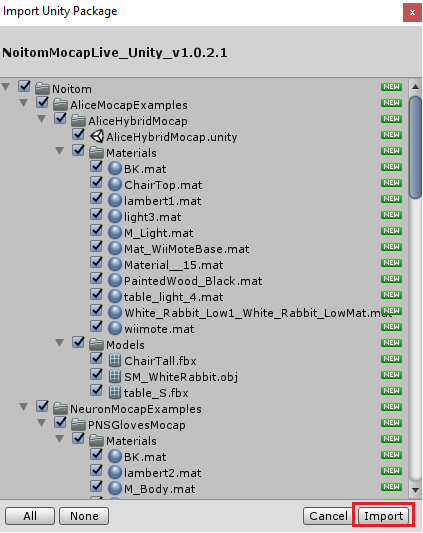
If you find bugs or need help please contact us at [Noitom\_service@noitom.com](mailto:Noitom_service@noitom.com%20)

# 2. Import Package

Choose Assets > Import Package > Custom Packages to import **NoitomMocapLiveUnity.unitypackage**.







# 3. Structure Overview

## 3.1 Example Scenes

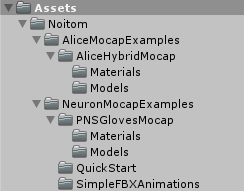
Here are some pre-made examples explained of the SDK package:

* + - 1. **QuickStart**: The simplest scene to show how to use this SDK.

Use NeuronTransformInstance.cs script to drive avatar fbx.

* + - 1. **PNSGlovesMocap**: A simple example of how to drive hands only part of avatar fbx.
      2. **SimpleFBXAnimations**：A simple example of how to use exported fbx animations file from Axis Neuron/Axis Studio to drive avatar fbx.
      3. **AliceHybridMocap:** A simple example of how to use hybrid props from Alice in Unity.

Use NeuronTracker.cs script to motion Props fbx



## 3.2 Core Scripts Overview

1. **Assets/Noitom/Scripts/Mocap/MocapApiManager.cs**

Manages connections with Axis, get motion data from MocapApi. Users does not need to pay attention to this script

1. **Assets/Noitom/Scripts/Mocap/NeuronActor.cs**

Data class to store only the most recent motion data frame, also provides methods to parse the received motion data which is received as float values from the network.

1. **Assets/Noitom/Scripts/Mocap/NeuronSourceManager**

Neuronsourcemanager plays the role of connecting data sources. Each connection needs to be added to the Neurosourcemanager instance, and hang props and characters under it as child nodes.

1. **Assets/Noitom/Scripts/Mocap/NeuronSource.cs**

NeuronSource manages instances of NeuronActor with two dictionaries called activeActors and suspendedActors. NeuronSource monitors the latest timestamp update inside

NeuronActor in the OnUpdate method and use a threshold to judge if any actor is lost (number of actors in Axis Neuron has changed or the connection was lost completely). When this happens NeuronSource will add or remove actors between the two dictionaries and notify NeuronActor.

1. **Assets/Noitom/Scripts/Mocap/NeuronInstance.cs**

Base class for all kinds of instances for receiving motion data.

Inherits from UnityEngine.MonoBehaviour. NeuronInstance provides callbacks for state changes and the receiving of mocap info from a NeuronActor instance which was bound to this instance by connect or other methods. This class is not intended to be used directly but can be inherited to provide custom methods to apply motion data, handle states change and mocap info.

1. **Assets/Noitom/Scripts/Mocap/NeuronAnimatorInstance.cs**

Inherited from NeuronInstance. Provides custom methods to apply Neuron motion data to the transform components of the bones bound in the Unity animator component. Needs a humanoid skeleton setup to work properly.

1. **Assets/Noitom/Scripts/Mocap/NeuronAnimatorPhysicalReference.cs**

Data class for initialization and cleanup of a reference skeleton used for motions based upon Unity’s rigidbody component. Used by NeuronAnimatorInstance if physics toggle is enabled.

1. **Assets/Noitom/Scripts/Mocap/NeuronTransformsInstance.cs**

Inherited from NeuronInstance. Provides custom methods to apply Neuron motion data directly to transform components. Use this for non-humanoid skeletons or skeletons with more bones then the default setup used in Unity.

1. **Assets/Noitom/Scripts/Mocap/NeuronTransformsPhysicalReference.cs**

Data class for initialization and cleanup of a reference skeleton used for motions based upon Unity’s rigidbody component. Used by NeuronTransformsInstance if physics toggle is enabled.

1. **Assets/Noitom/Scripts/Mocap/ NeuronRigidbody.cs**

Use this script to receive hybrid mocap props data from Axis Studio

1. **Assets/Noitom/Scripts/Mocap/ NeuronRigidbodyListHelper.cs**

Use this script to show how many hybrid mocap props in Axis Studio and show their ids.

1. **Assets/Noitom/Scripts/Mocap/NeuronTracker.cs**

Use this script to receive rigidbody props data from Alice.

1. **Assets/Noitom/Scripts/Utilities/BoneLine.cs**

Utility class using a line renderer to draw bone lines.

1. **Assets/Noitom/Scripts/Utilities/BoneLines.cs**

Utility class for Neuron editor to add or remove BoneLines.

1. **Assets/Noitom/Scripts/Utilities/BoneRigidbodies.cs**

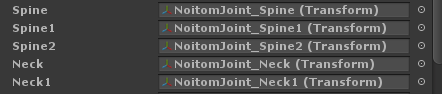
Utility class for Neuron editor to add or remove Rigidbodies.

1. **Assets/ Noitom /Scripts/Utilities/FPSCounter.cs**

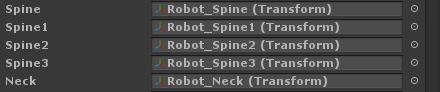
Utility class to calculate the FPS (Frames-Per-Second).

## 3.3 Public Variables of Neuronanimatorinstance.cs

1. **Actor ID** is character‘s id number. Default value is 0 which is the first character in Character panel of Axis
2. **Skeleton Type**
3. Select **Perception Neuron Studio**, it means the script will use the **PN Studio** skeleton structure which includes 3 joints of spine, and 2 joints of neck.

****

1. Select **Perception Neuron**, it means the script will use the **PN/PN Pro** skeleton structure which includes 4 joints of spine and 1 joint of neck.

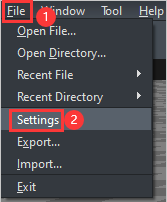
****

1. **Bound Animator** the Animator component this instance should use for the received motion data. You can use this if you don’t want to keep the script on the same GameObject as the animator component.
2. **Motion Update Method** tells the instance if it should use rigidbody functions provided by Unity to move and rotate each bone. The default method is to apply the received float values directly to the transform components of each bone.
3. **Disable bone movement** tells the instance if it should apply avatar’s bone’s transform movement

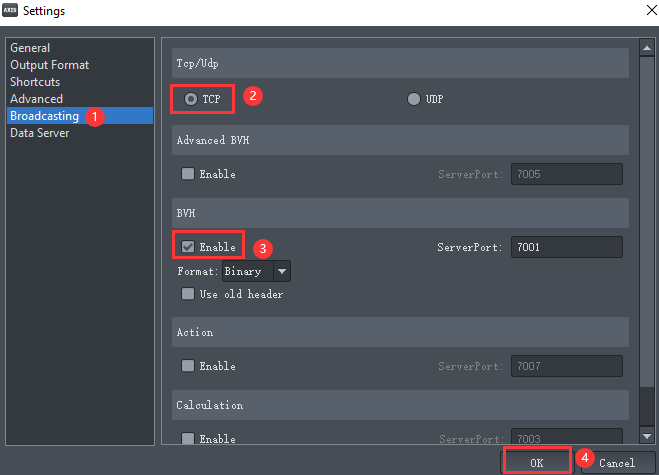
# 4. Data streaming setup in Axis

## 4.1 Axis Neuron/Axis Neuron Pro Setup

1. Run Axis Neuron/Axis Neuron Pro



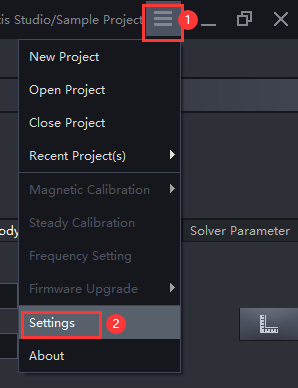
1. Click File >Settings >Broadcasting, select TCP/UDP protocol and enable BVH data streaming by checking the Enable checkbox. In this case we use TCP protocol.



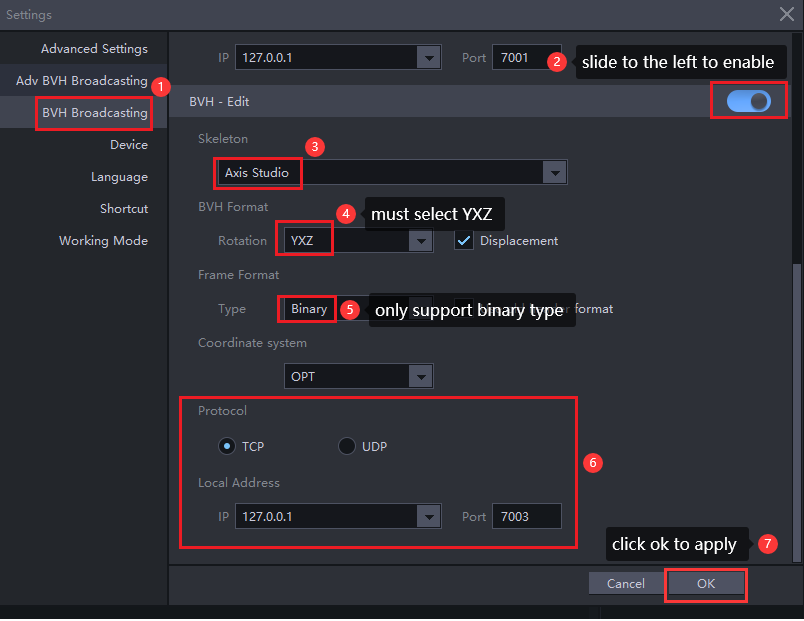
1. Playback a recorded motion capture data or connect a device

## 4.2 Axis Studio Setup

1. Run Axis Studio and open a project
2. Click Menu >Settings



1. Navigate to BVH Broadcasting, enable BVH-Edit (or BVH-Capture) ，in this case we select BVH -Edit. Set network and other appropriate options, then click OK.

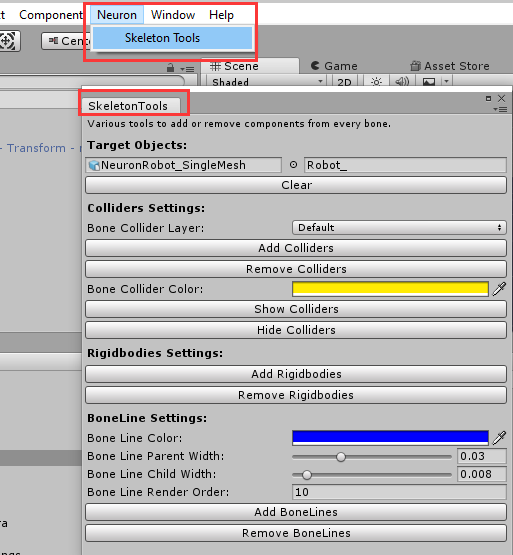


Note：Enable BVH-Capture if you stream motion capture data from Capturing viewport.

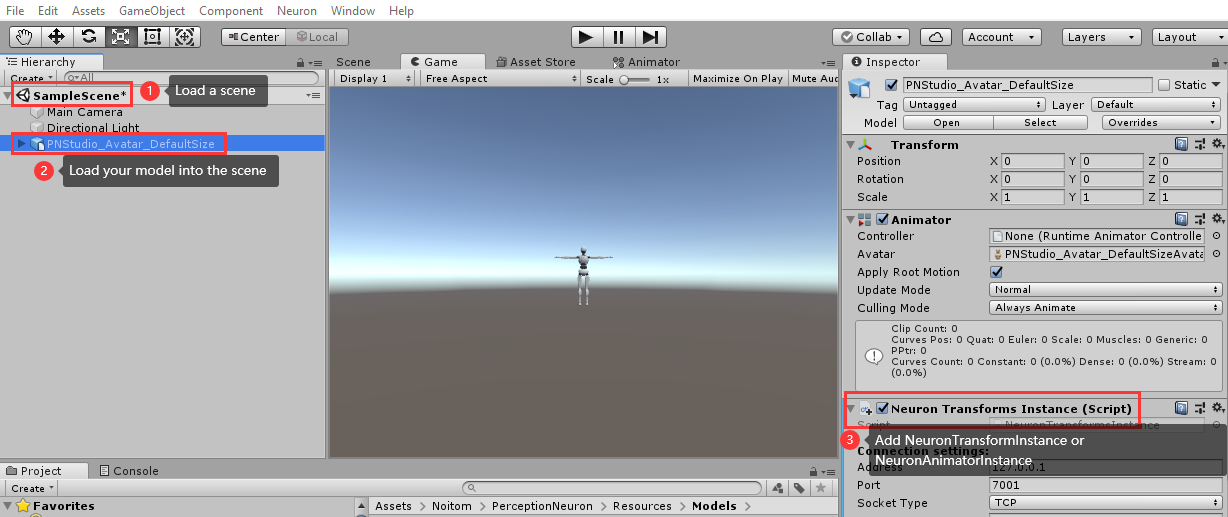
1. Playback a recorded motion capture data or connect a device

# 5. Neuron Skeleton Tools

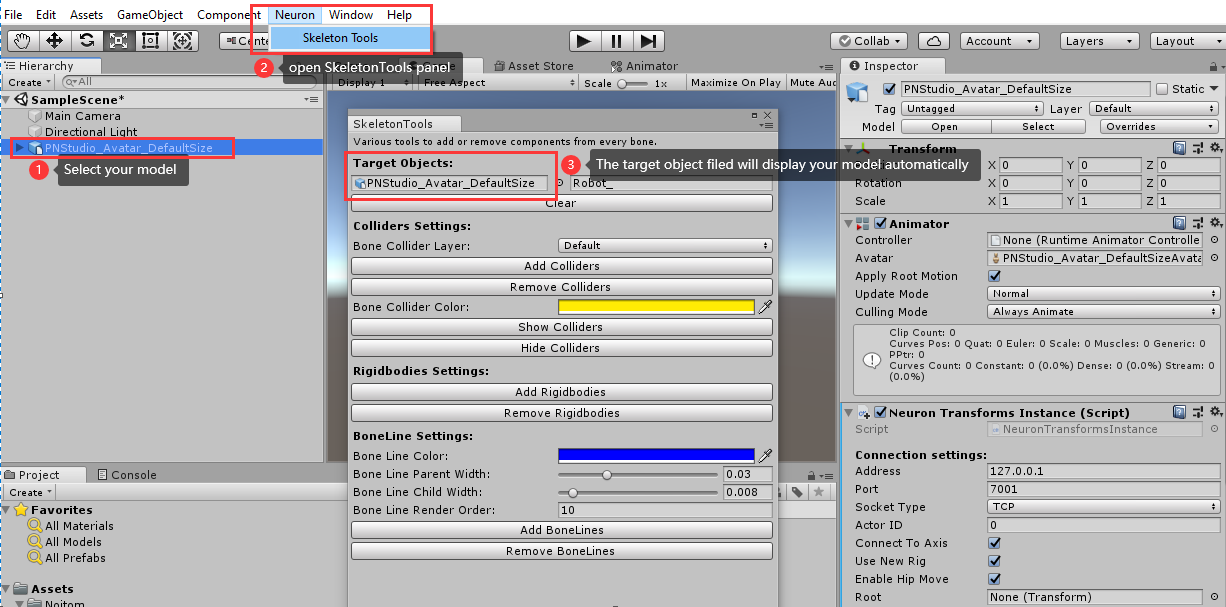
We programmed a nifty tool to help you with configuring and making changes to your avatar. Its called Skeleton Tools and can be found in the menu Neuron -> Skeleton Tools.



1. Select your GameObject that either has a NeuronAnimatorInstance or NeuronTransformInstance script attached to it. This will provide the tools script with the right interface to work with.

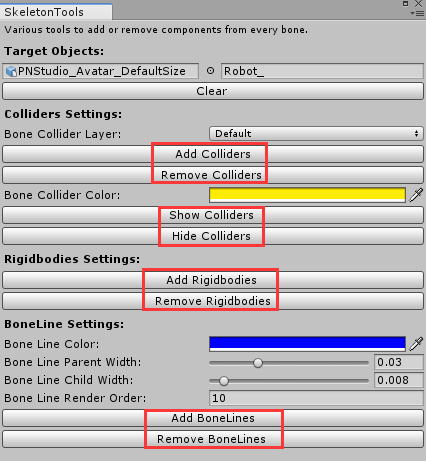


1. Keep your model selected and click Neuron >SkeletonTools to open the SkeletonTools panel.



1. Now you can add/remove components for bones such as:
2. Adding/removing of colliders.
3. Showing/hiding of colliders.
4. Adding/removing of rigidbody components.
5. Adding/removing of bone lines. (Bone lines are a great tool for debugging and can also be added to instantiated reference skeleton, when using physics update, during runtime).

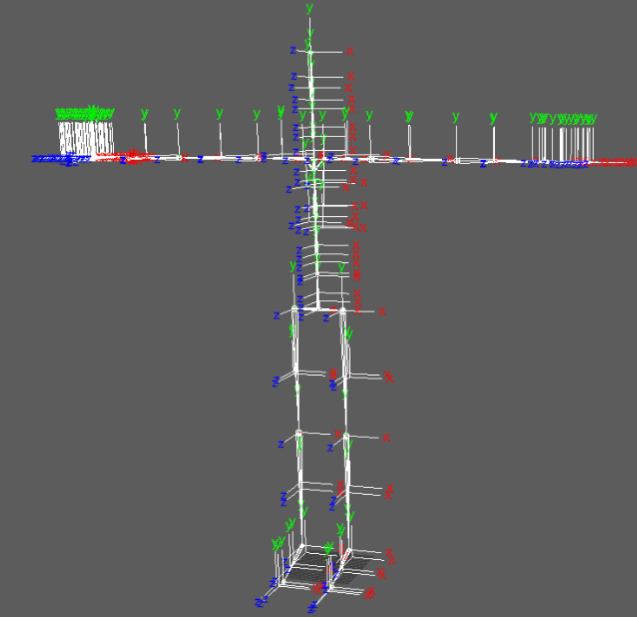
The render order setting defines which line to render onto of each other if you have multiple bone lines.



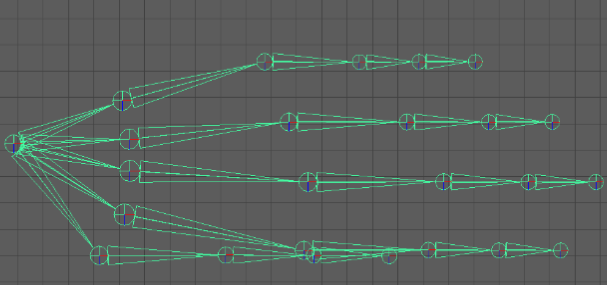
# 6. How to configure a character model

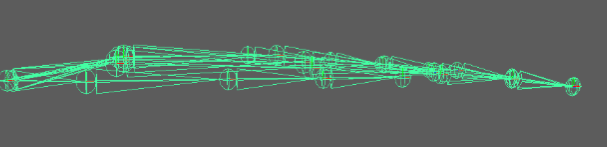
The following is a guide for how to configure a new character model to receive real-time motion data by Perception Neuron. Before we start you need to be aware of a few things:

1. Your model needs to be rigged on a humanoid skeleton
2. The bones of your rig can not have any existing rotations in them
3. Your model needs to be rigged in a T-Pose
4. The rotation value of all bones in T pose is zero
5. Each bone’s local rotation axes should be the same orientation as shown in the following figure:

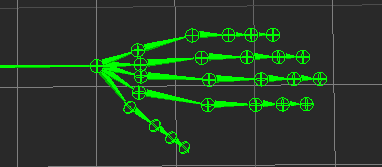


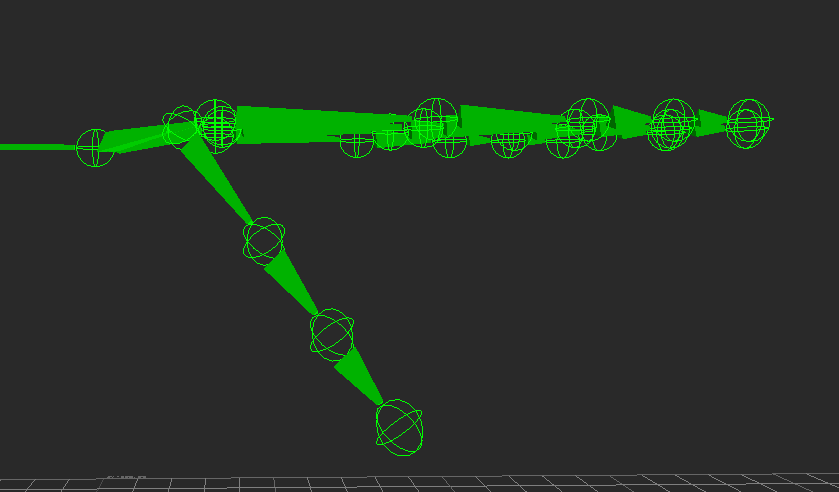
1. Fingers posture refer to the following two kinds, one is thumb inside the palm and the other one is thumb open.
2. Thumb inside the palm shown as in the following figures.



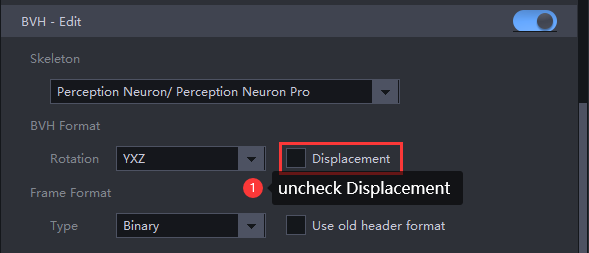


1. Thumb open shown as in the following figures.





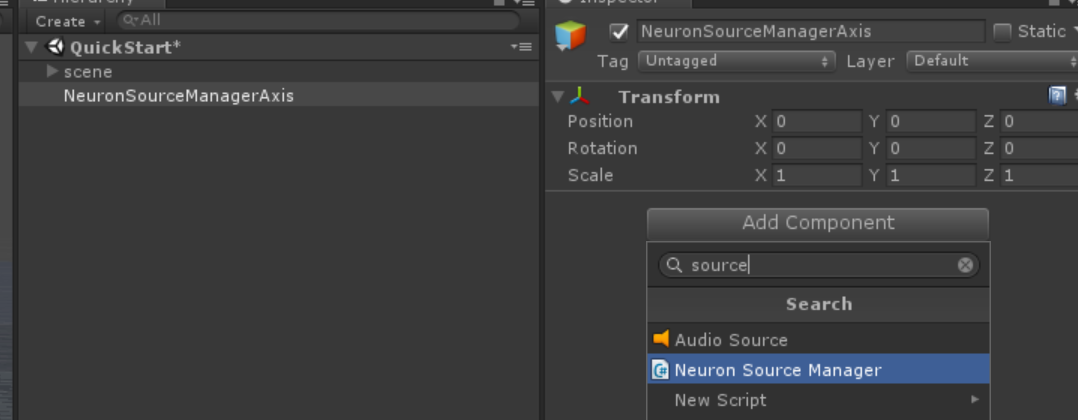
1. We support two sets of Skeleton Structure PN/PN Pro and PN Studio. Please refer to [APPENDIX B: SKELETON BONES](#_APPENDIX_B:_SKELETON)
2. Set BVH output data without displacement in Axis If you’re model is not rigged with our skeleton setup or your model may be twisted. For the BVH data without displacement, except the root node (Hip) having displacement and rotation, other bones only have rotation data.

.

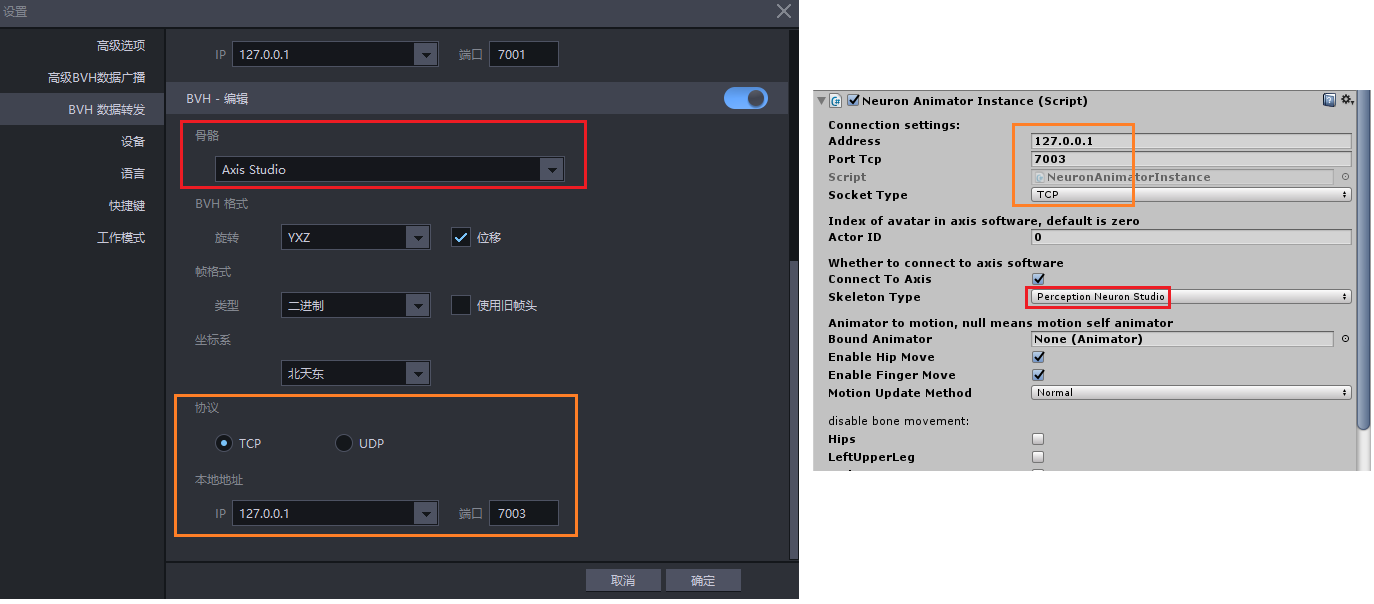
Note: If you don’t consider the points mentioned above the applied motion from the integration will look wrong. Please make sure to correct those issues. Worst case suggestion is to rig your model on the included skeleton template found at: Noitom/PerceptionNeuron/Resources/Models

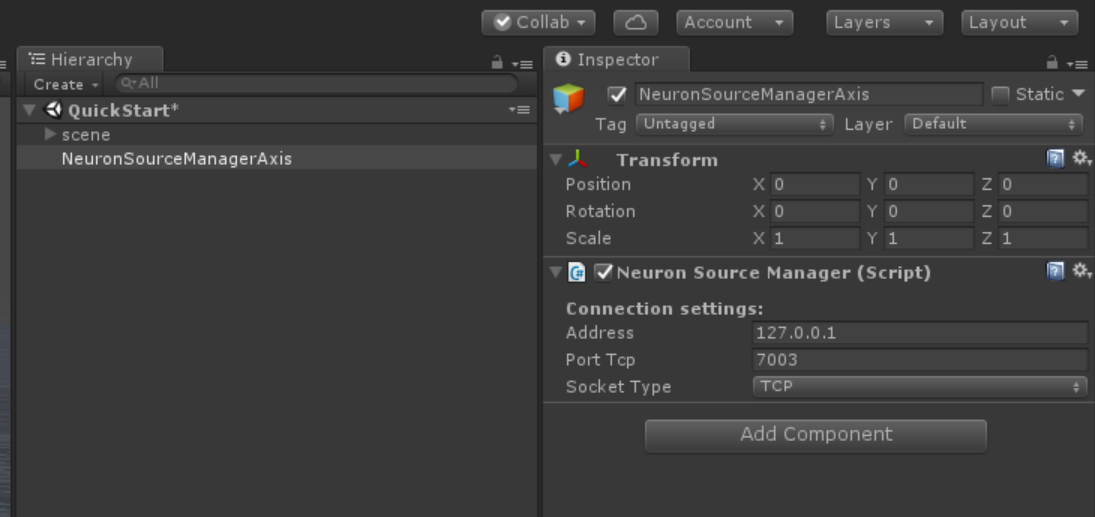
# 7. How to Apply Real-time Data from Axis

1. Import PerceptionNeuron.unitypackage
2. Add an empty GameObject. In the scene, add the neurosourcemanager component through addcomponent

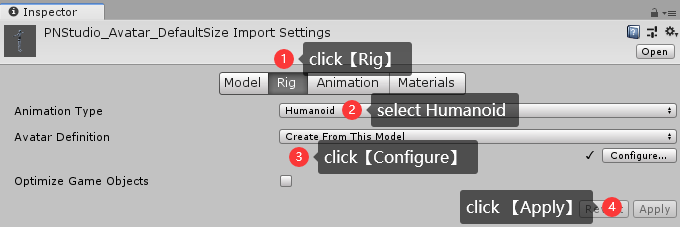


1. The connection settings are set according to the data flow parameters set in the axis software. For the data flow forwarding settings of the axis software, refer to how to set the data flow forwarding in the axis software. The data source of this example will use the axis Studio software to playback the data. It is recommended that the axis Studio software circularly broadcast and play the dynamic capture files to ensure that the data flow is not interrupted. The relevant settings of the data flow are shown in the figure:

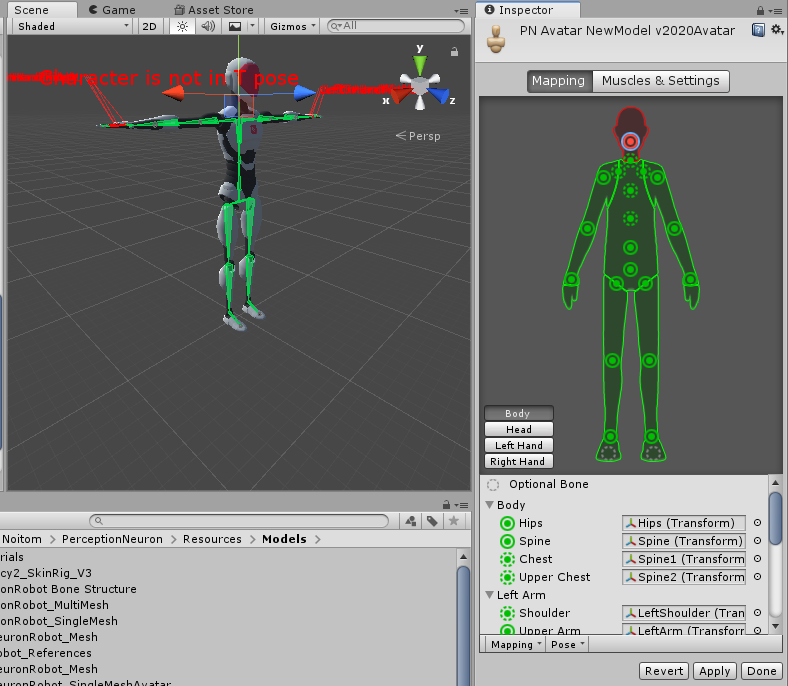




1. You need to use a humanoid human skeleton model. For the model, please refer to how to configure / select the dynamic capture model and import the model into unity. The axis Studio software sample model will be used in this example
2. In the Project tab in Unity click on your model and then click on the Rig tab in the inspector windows. Set your Animation Type to “Humanoid”, then click on 【Apply】.



1. Next click on 【Configure】to check the bone mapping. Usually Unity will find the right bone references for almost all the humanoid skeleton so its good check them.

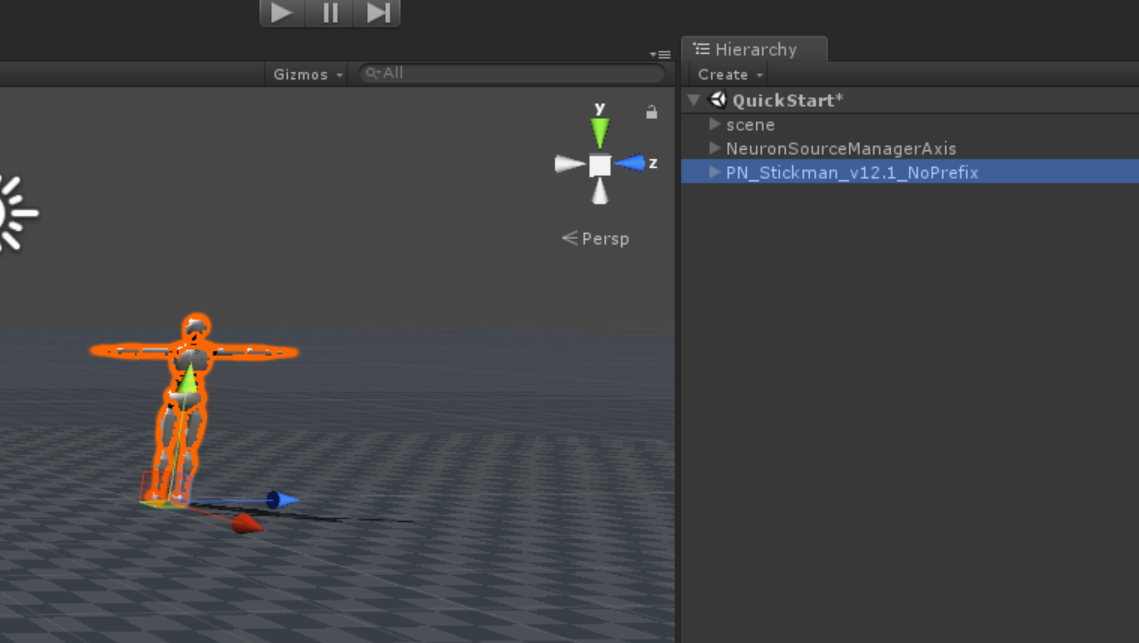


1. Click on Mapping and then click on Load to load our template file that fixes the mapping for you. IN the new window navigate to Noitom/PerceptionNeuron/Resources and load the file AxisStudioMapping.ht.

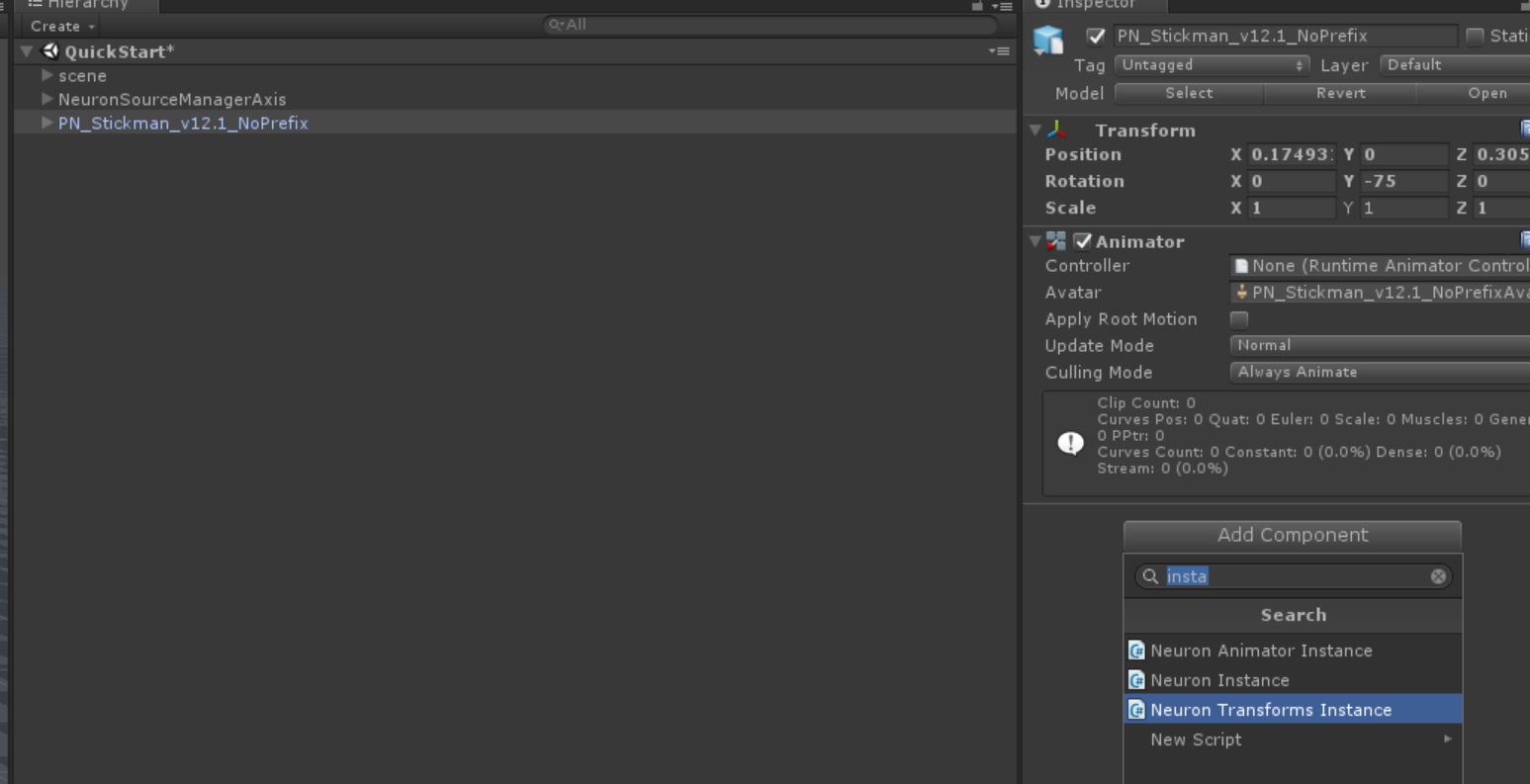
Please refer to [APPENDIX A: SKELETON MAPPING](#_APPENDIX_A:_SKELETON)

Make sure you have all the right bones mapped to the correct body segments inside each of the tabs for Body and Click on 【Apply】and then on 【Done】

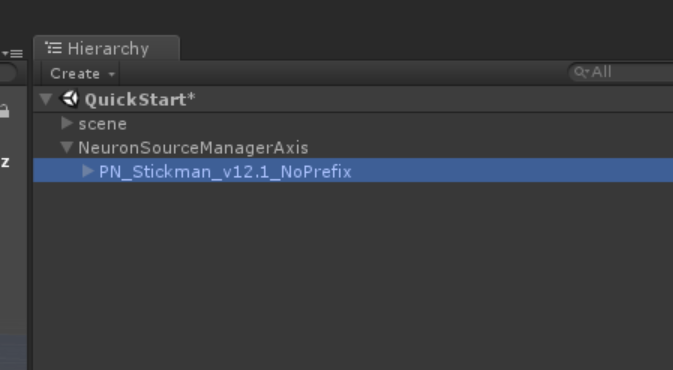
1. Load the scene you want to use your model in. Click and drag your model into the Hierarchy tab to load it into the scene.



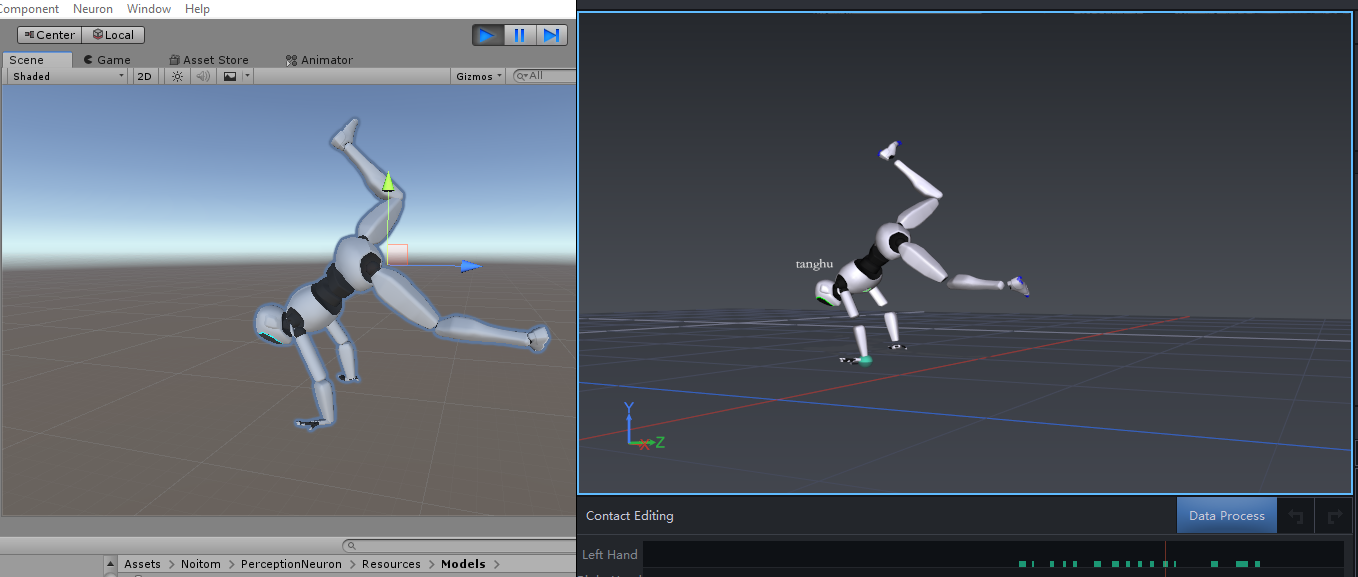
1. Select your model in the Hierarchy tab and click on 【Add Component】in the inspector window. IN the search field type “Neuron” and select the script “NeuronTransformInstance.cs”.



1. Adjust the settings of the NeuronTransformInstance Script according to your own setup. enable the toggle “Connect To Axis” to make the script connect to Axis, click 【bind】 to do skeleton retargeting automatically before hit Play. Make sure you have Axis Neuron/ Axis Studio running and playing a recorded animation in loop or a live actor connected to it. Also make sure you turned on BVH data streaming in the Neuron Axis settings.



1. Click on 【Play】 in Unity to see the motion being applied to your new model.



# 8. HMD INTEGRATION

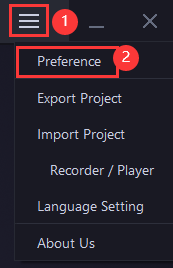
Getting any HMD synced up with the Axis integration can be a bit tricky. Here is how we got it done and what we found out to be working the best. There are three core rules worth mentioning:

1. The rotation tracker of the HMD should always have priority. Don’t overwrite its rotation values with something else and don’t use the head rotation values from the Axis.
2. Don’t use the positional tracking of the HMD.
3. Never parent the HMD cameras or GameObject to the skeleton setup.

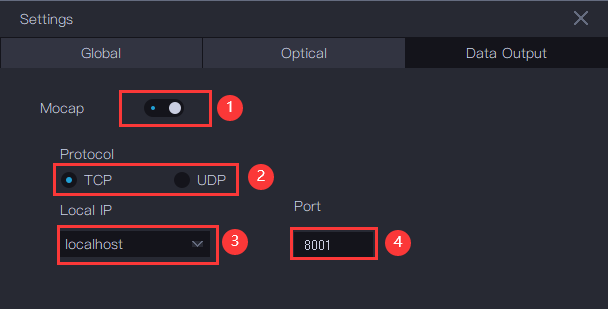
# 9. Setup in AHM/Alice

## 9.1 AHM/Alice Data Streaming Setup

1. Run AHM/Alice



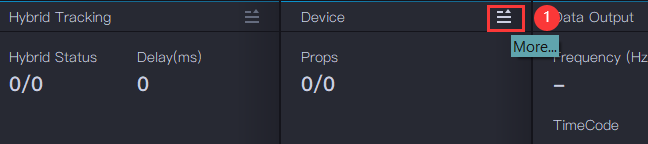
1. Click  >Preference >Data Output, enable Mocap, select TCP/UDP protocol and set the IP address and the port. In this case we use TCP protocol.



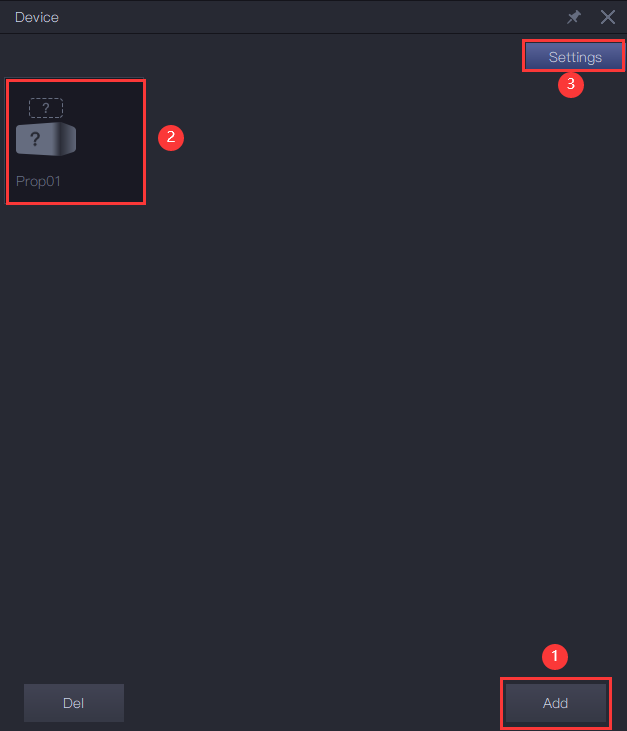
1. Click 【OK】 to apply.

## 9.2 AHM/Alice Device Setup

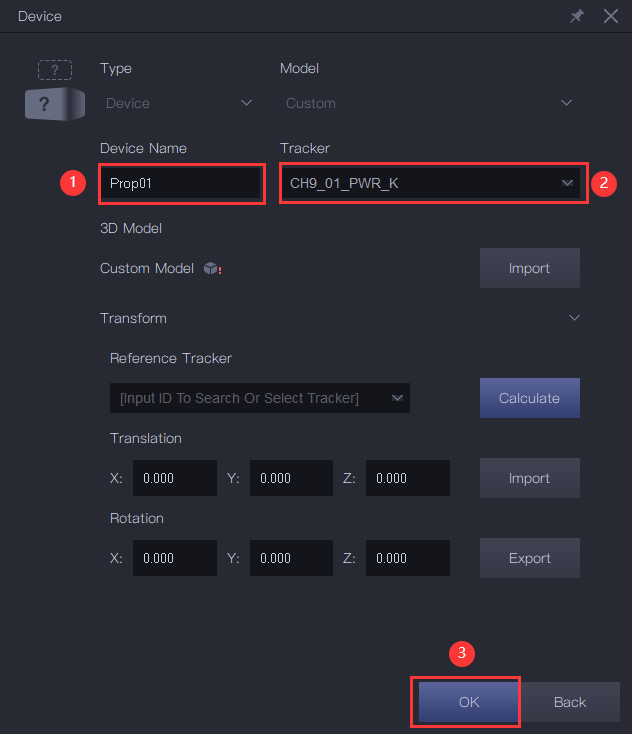
1. Run AHM/Alice
2. Click Device >



1. Click 'Add' to add new device. Select this device and click 'Settings'.



1. Input the name of Device (the name will be used in Unity to identify the specific device), and select the AHM/Alice tracker (the hybrid tracker) to tracking this device. Then click 'OK' to apply.



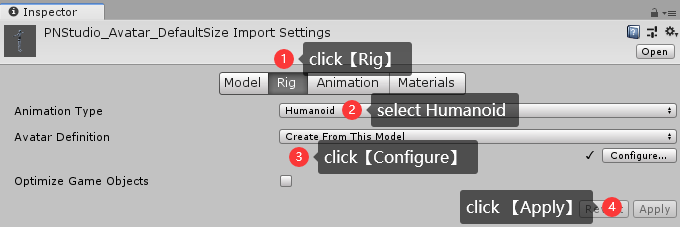
# 10. How to Apply AHM/Alice Device Data

1. Create a GameObject and 【Add Component】, select "Neuron Tracker".
   1. Address: the IP address set in 9.1
   2. Port: the port set in 9.1
   3. Socket Type: TCP/UDP set in 9.1
2. Set the parameters as the settings in AHM/Alice
   1. Device Name: the name of the device set in 9.2

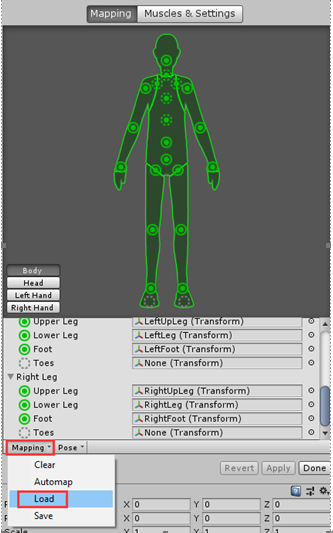
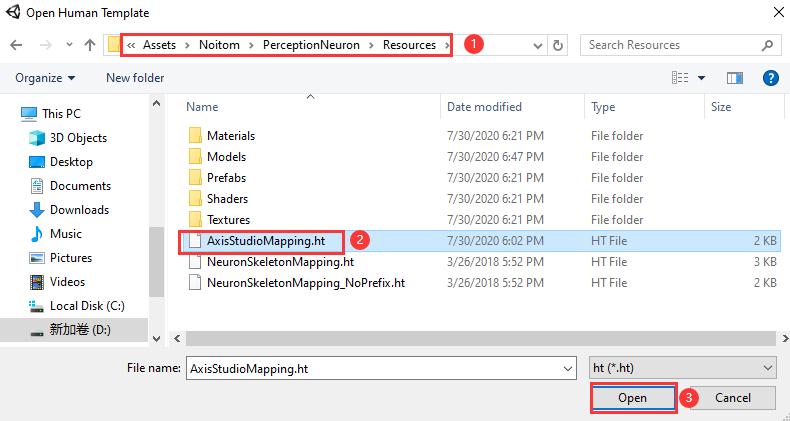
# 11. How to Apply FBX file Data from Axis

## 11.1 Humanoid Animation type.

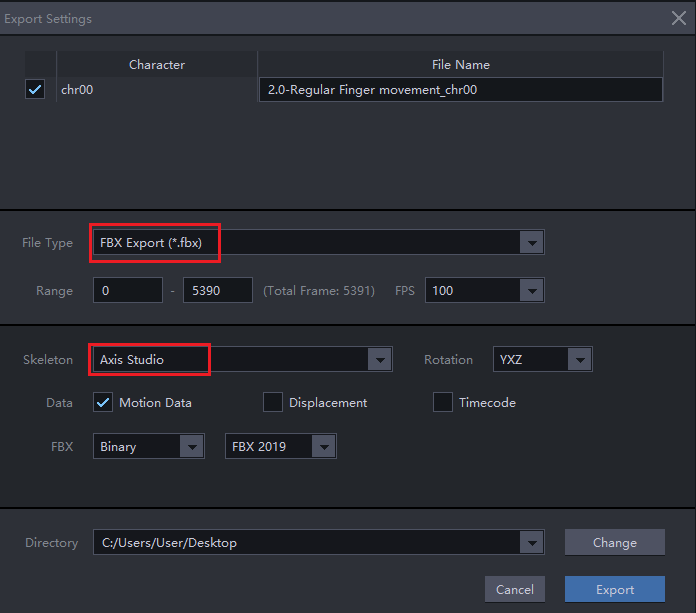
1. Import your model and in the Project tab in Unity click on your model and then click on the Rig tab in the inspector windows. Set your Animation Type to “Humanoid”, then click on 【Apply】.



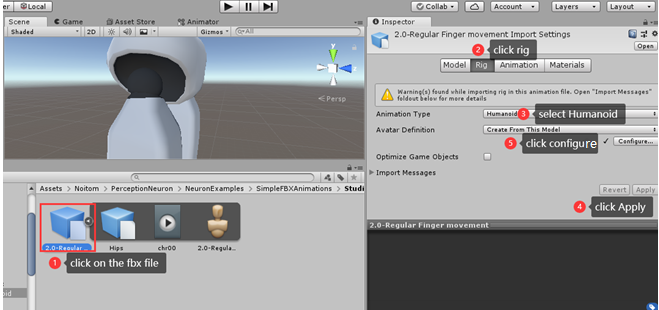
1. Next click on 【Configure】to check the bone mapping. Usually Unity will find the right bone references for almost all the humanoid skeleton so its good check them. You can load pre-make .ht mapping file or manually do the mapping.

****

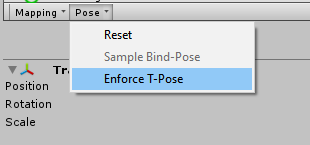
1. Click【Apply】
2. If the bone Mapping is correct, but the character is not in the correct pose, you will see the message “Character not in T-Pose”. You can try to fix that with **Enforce T-Pose** or rotate the remaining bones into T-pose.
3. Export an fbx file from Axis Neuron/Axis Studio. In this case the export setting in Axis Studio is shown as the figure.

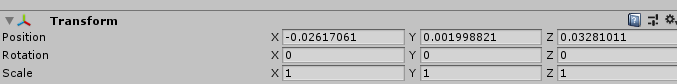


1. Import this .fbx format animation file into Unity and click on it. On the Rig tab in the inspector windows. Set your Animation Type to “Humanoid”, then click on 【Apply】.

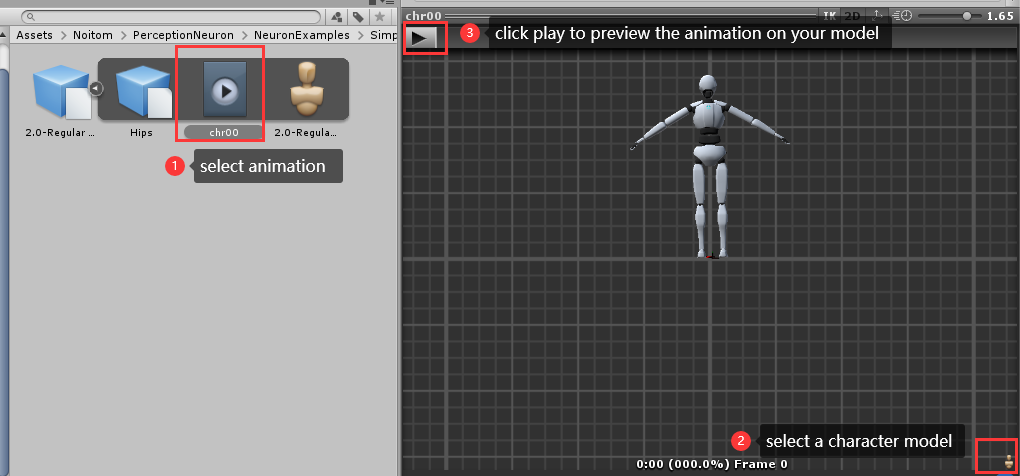


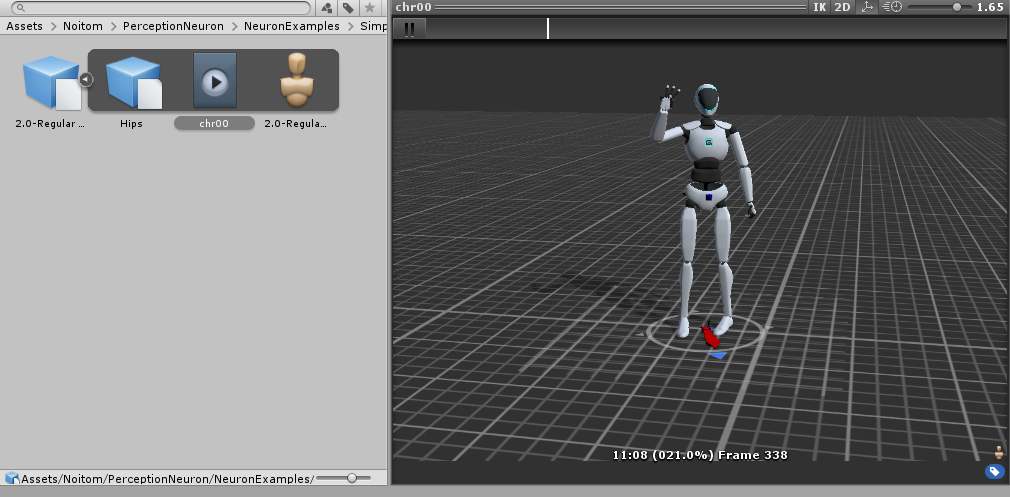
1. Make sure the bone Mapping and the initial T-pose are exactly same as the settings of the imported model we set in Step4, You may need to do Enforce T-Pose and rotate some bone’s value in the Transform panel.





1. Click on 【Apply】and then on 【Done】
2. You can preview the animation on your model



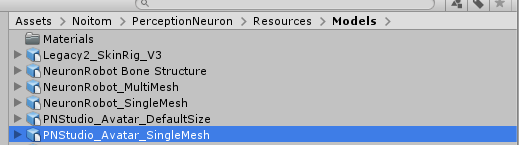


## 11.2 Generic Animation Type

You can also apply Generic Animation type if the bone names and hierarchical structure of

the model are exactly same as our animation .fbx file.

Refer to Noitom/PerceptionNeuron/Resources/Models/PNStudio\_Avatar\_SingleMesh.fbx

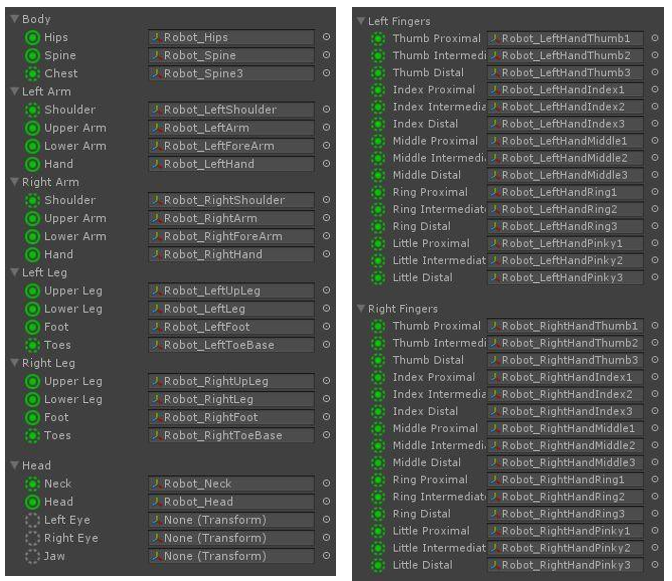


# 12. APPENDIX A: SKELETON MAPPING

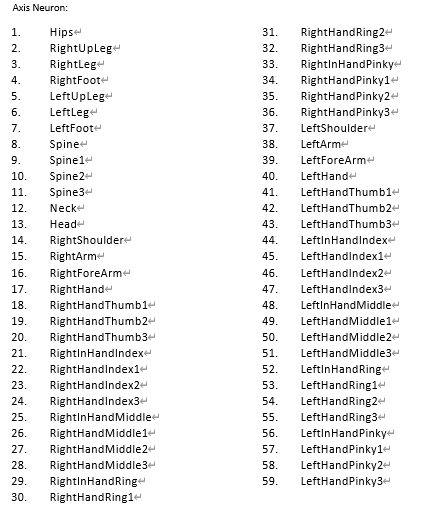
## 12.1 PN Studio skeleton structure mapping

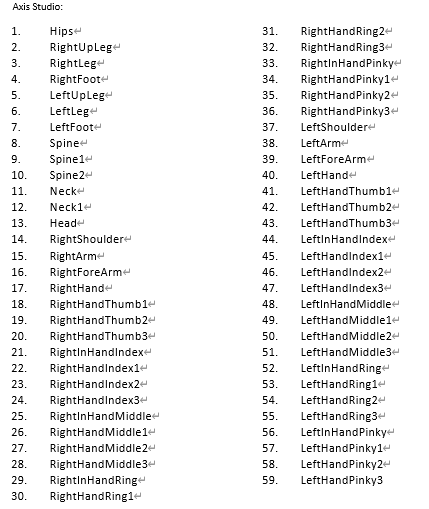
****

## 12.2 PN/PN Pro skeleton structure mapping



# 13. APPENDIX B: SKELETON BONES



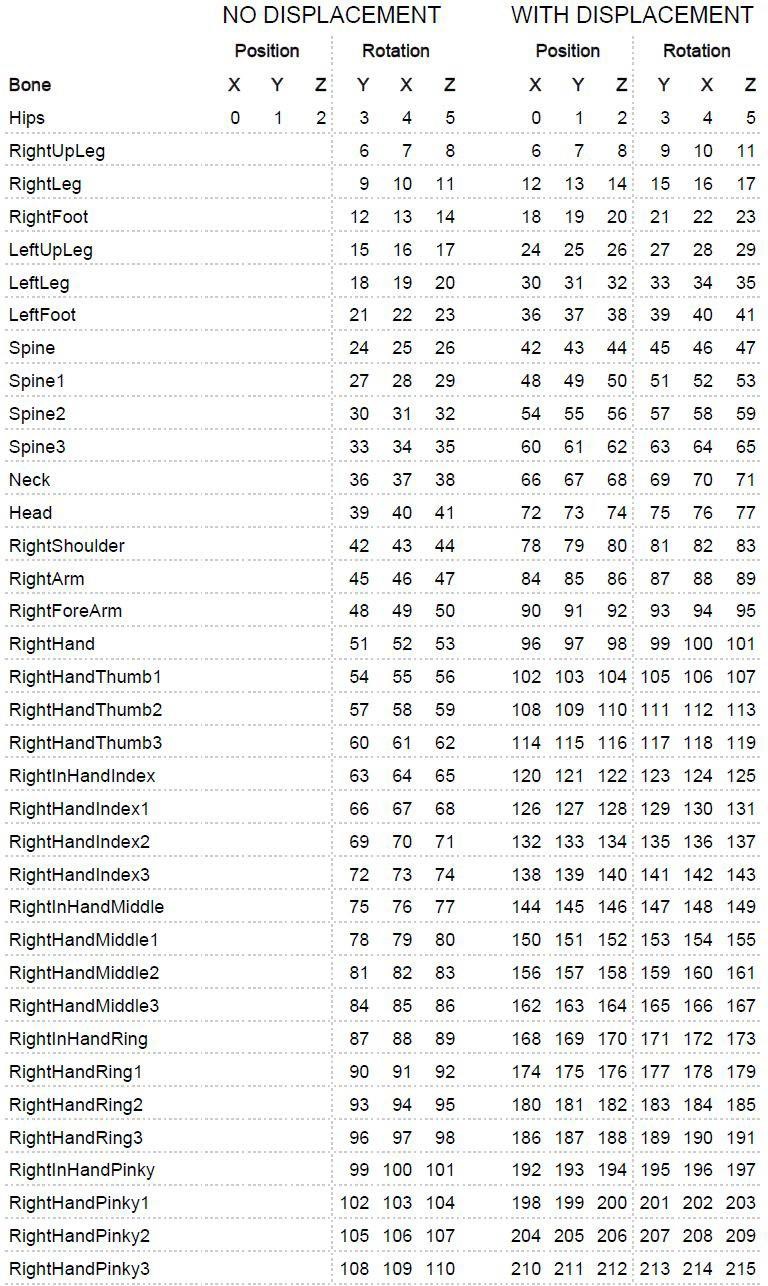


This is the complete structure of our skeleton model. If we use the humanoid skeleton in Unity, we skip some of them resulting in 51 bones remaining. However, we need to include the data of the bones we skip in the following bones for the correct values. We skip all the InHand bones and two of the spine bones.

# 14. APPENDIX C: BINARY DATA SEQUENCE

On the next two pages you’ll find a complete graph of the whole sequence of the binary data received from Axis Neuron. It is a one-dimensional float array with different ordering and length depending on whether you’re using displacement data or not.

**Axis Neuron**



**AxisStudio**

