PERCEPTION NEURON UNITY HANDBOOK

Perception Neuron Unity Integration 0.2.19

Noitom Technology Co.Ltd： http://noitom.com.cn/
Nov,2th,2020

TABLE OF CONTENTS

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 in animation.

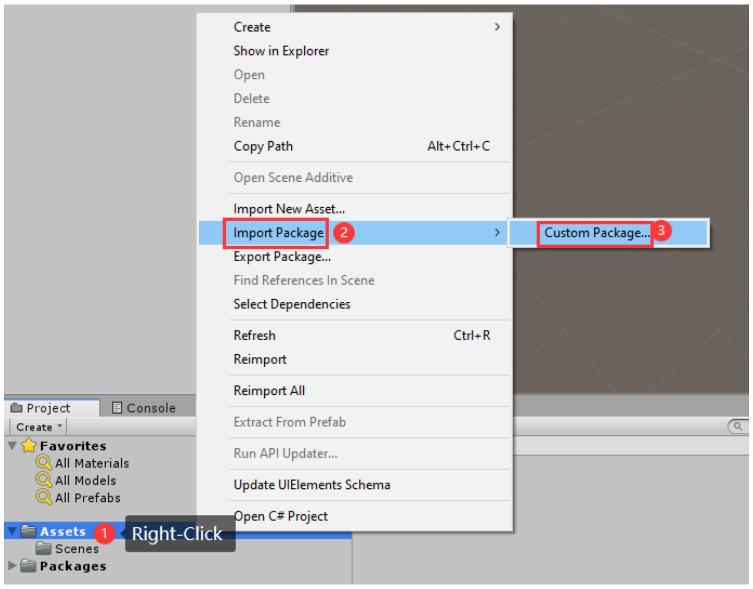
The data stream to the game engine is identical regardless if you use recorded or live motion data. This allows you to record certain actions and use them to test before putting on the full system and doing a live test.You can have multiple actors at the same time inside Axis and have them all streamed 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 doesn’t necessarily have to be the same computer running Unity.

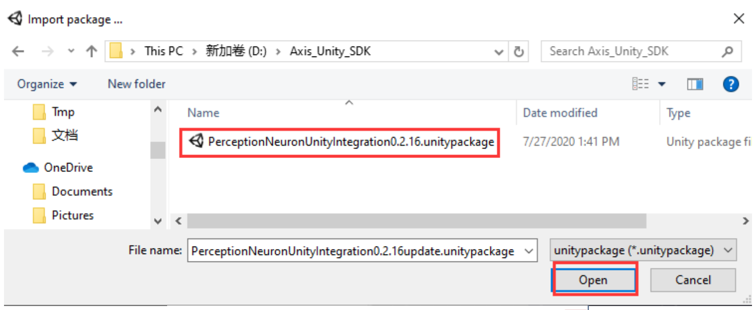
If you find bugs or need help please contact us at Noitom_service@noitom.com

If you’re looking to extend your real-time Perception Neuron experience, then have a look at the NeuronDataReader SDK.

2.Import SDK Package

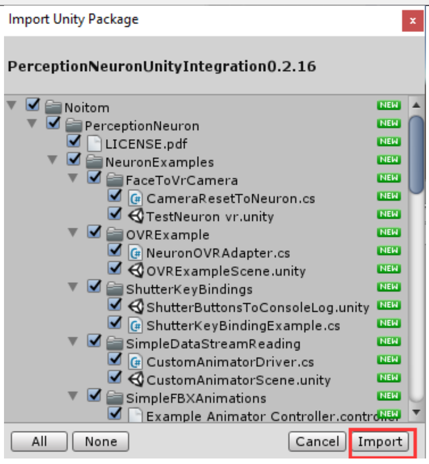
Choose Assets > Import Package > Custom Packages to import PerceptionNeuron Unity SDK.





File name: PerceptionNeuronUnityIntegration0.2.16update.unitypackage

unitypackage (*.unitypackage)



3. Structure Overview

3.1 Example Scenes

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

1. **FaceToVrCamera**: A simple example to observeneuron-body in VR mode.

   The CameraResetToNeuron class help user to align the VR-HMD axis with the neuron axis.

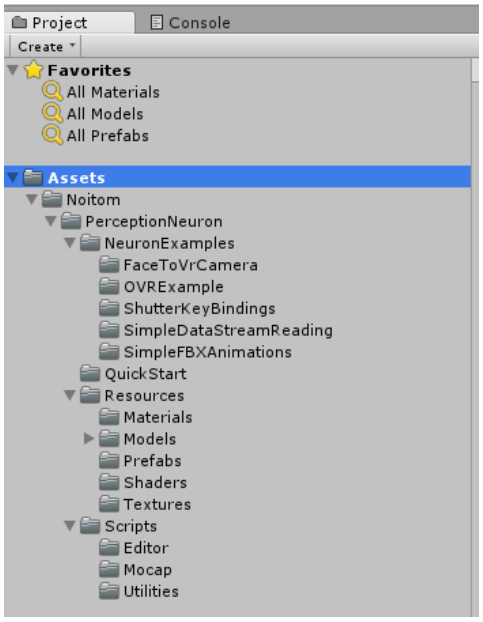
   This feature will auto effect after scene started, or user can press ‘R’ key to re- align.

   You can refer to CameraResetToNeuron.cs for details. when you use this script, make sure you have resetthe camera’s position and rotation to zero.

2. **OVRExample**： A simple example of how to use Perception Neuron with a VR HMD.

3. **ShutterKeyBinding**： A simple example of Shutter hotkey

4. **SimpleFBXAnimation**： some simple example of how to use exported animations from Axis Neuron/Axis Studio directly inside Unity.



```
Favorites
    All Materials
    All Models
    All Prefabs

Assets
  Noitom
    PerceptionNeuron
      NeuronExamples
        FaceToVrCamera
        OVRExample
        ShutterKeyBindings
        SimpleDataStreamReading
        SimpleFBXAnimations
      QuickStart
      Resources
        Materials
        Models
        Prefabs
        Shaders
        Textures
      Scripts
        Editor
        Mocap
        Utilities
```

3.2 Core Scripts Overview

1. **Assets/Neuron/Scripts/Mocap/NeuronDataReaderManaged.cs**

   C# managed version for receiving BVH data from Axis Neuron.

2. **Assets/Neuron/Scripts/Mocap/NeuronConnection.cs**

   NeuronConnection manages connections with Axis Neuron using NeuronDataReaderManaged.cs. You can connect to multiple instances of Axis Neuron and each connection will be mapped to a NeuronSource instance.

3. **Assets/Neuron/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.

4. **Assets/Neuron/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. NeuronActor also saves mocap info and provides methods to register callbacks when it was resumed or suspended by NeuronSource.

5. **Assets/Neuron/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.

6. **Assets/Neuron/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.

7. **Assets/Neuron/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.

8. **Assets/Neuron/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.

9. **Assets/Neuron/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.

10. **Assets/Neuron/Scripts/Mocap/NeuronInstancesManager.cs**

    For each NeuronActor, NeuronInstancesManager keep exactly one NeuronAnimatorInstance. Used in NeuronDebugViewer.

11. **Assets/Neuron/Scripts/Utilities/BoneLine.cs**

    Utility class using a line renderer to draw bone lines.

12. **Assets/Neuron/Scripts/Utilities/BoneLines.cs**

    Utility class for Neuron editor to add or remove BoneLines.

13. **Assets/Neuron/Scripts/Utilities/BoneRigidbodies.cs**

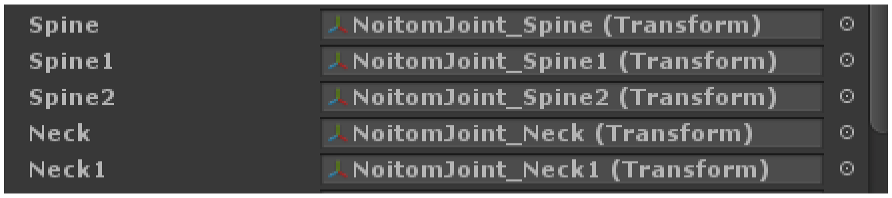
    Utility class for Neuron editor to add or remove Rigidbodies.

14. **Assets/Neuron/Scripts/Utilities/FPSCounter.cs**

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

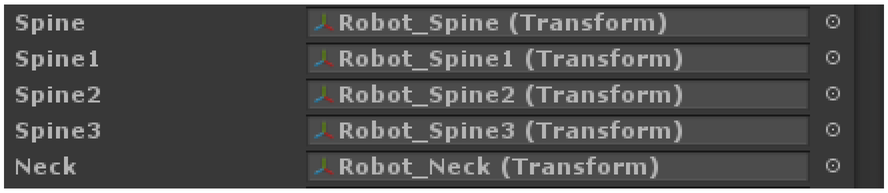
3.3 Public Variables of Neuronanimatorinstance.cs

1. **Address** is the IP address of the machine running Axis Neuron. If it's the same computer, the address should be 127.0.0.1. You can use this stream motion data from any other computer in the same network. In this case change the address accordingly.
2. **Port** is the port number associated with the BVH data stream output setting in Axis.
3. **Socket Type** is the socket type to be used for that data stream. This should be identical to your setting in Axis Neuron.
4. **Actor ID** is the id number for the actor you want to use. If you have more than one actor connected in Axis Neuron this id number will increase. Default is 0 which is the first actor.
5. **Connect To Axis** means the script will connect to Axis Neuron and apply the motion data. You can use this toggle to starts/stop the data stream.
6. **Use New Rig** if this flag is checked, it means the script will use the **PN Studio** seleton structure which includes 3 joints of spine, and 2 joints of neck.



Spine	NoitomJoint_Spine (Transform)
Spine1	NoitomJoint_Spine1 (Transform)
Spine2	NoitomJoint_Spine2 (Transform)
Neck	NoitomJoint_Neck (Transform)
Neck1	NoitomJoint_Neck1 (Transform)

if this flag is not checked, it means the script will use the **PN/PN Pro** seleton structure which includes 4 joints of spine and 1 joint of neck.



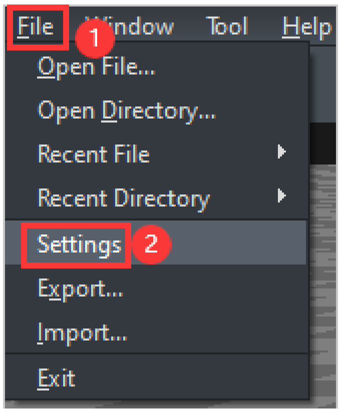
Spine	Robot_Spine (Transform)
Spine1	Robot_Spine1 (Transform)
Spine2	Robot_Spine2 (Transform)
Spine3	Robot_Spine3 (Transform)
Neck	Robot_Neck (Transform)

7. **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.
8. **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.

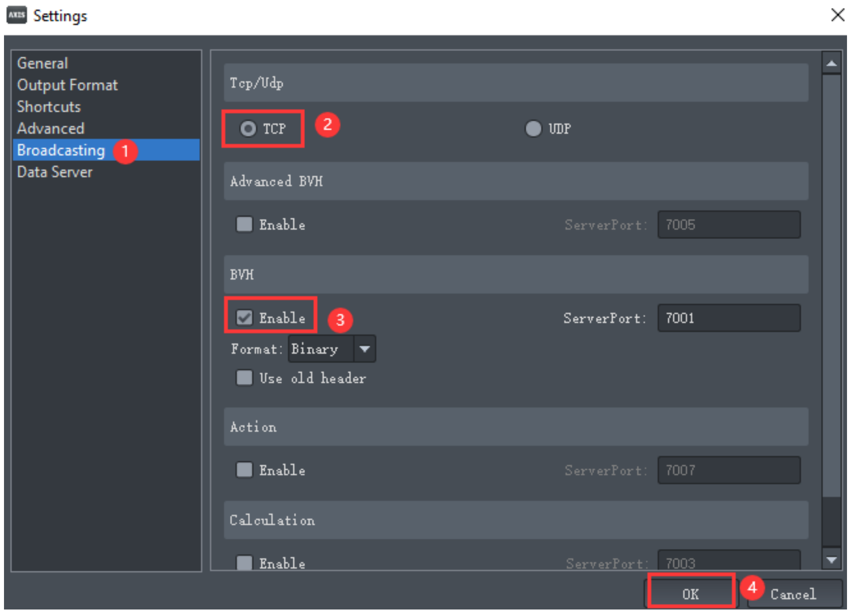
4.Data Streaming setup in Axis

4.1 Axis Neuron Setup

1. Run Axis Neuron/Axis Neuron Pro



2. 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.



Tcp/Udp: [✓ TCP] UDP

Advanced BVH: Enable ServerPort: 7005

BVH: [✓ Enable] ServerPort: 7001
Format: Binary
Use old header

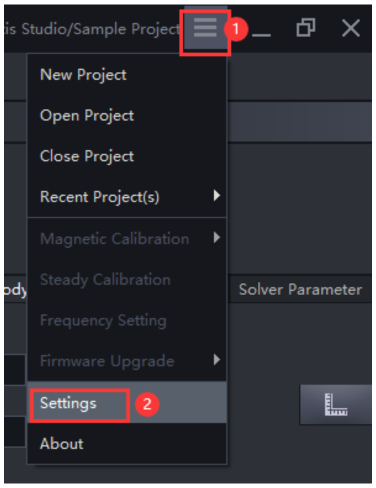
Action: Enable ServerPort: 7007

Calculation: Enable ServerPort: 7003

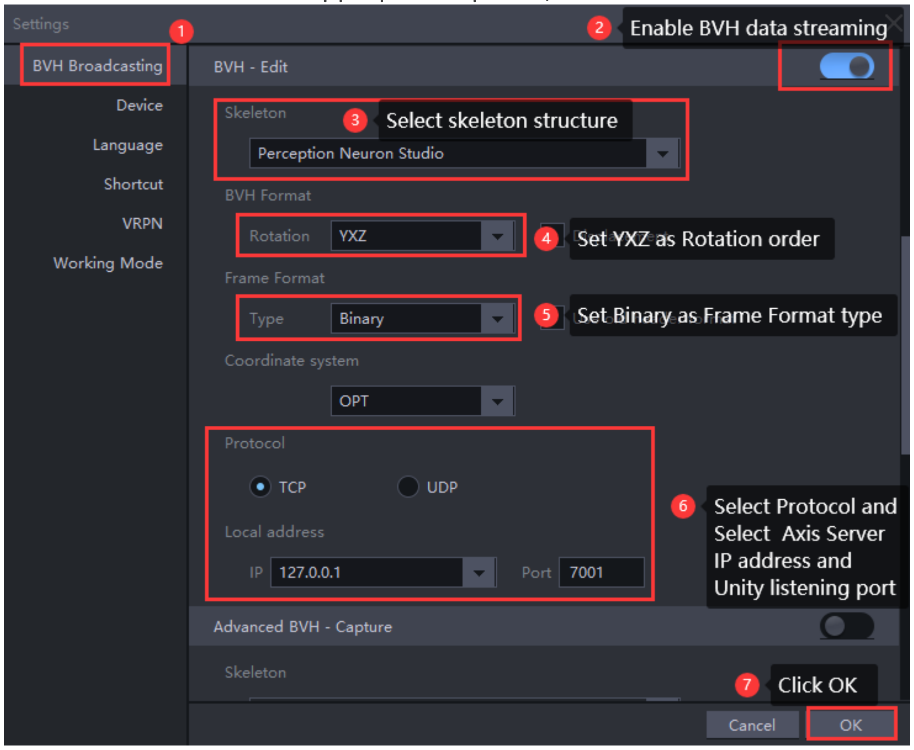
3. Playback a recorderd motion capture data or connect a device

4.2 Axis Studio Setup

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



3. 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.



1 BVH Broadcasting

2 Enable BVH data streaming

BVH - Edit: on

3 Select skeleton structure

Skeleton: Perception Neuron Studio

BVH Format

Rotation: YXZ

4 Set YXZ as Rotation order

Frame Format

Type: Binary

5 Set Binary as Frame Format type

Coordinate system: OPT

Protocol: TCP [✓] UDP

Local address

IP: 127.0.0.1 Port: 7001

6 Select Protocol and Select Axis Server IP address and Unity listening port

Advanced BVH - Capture: off

Skeleton

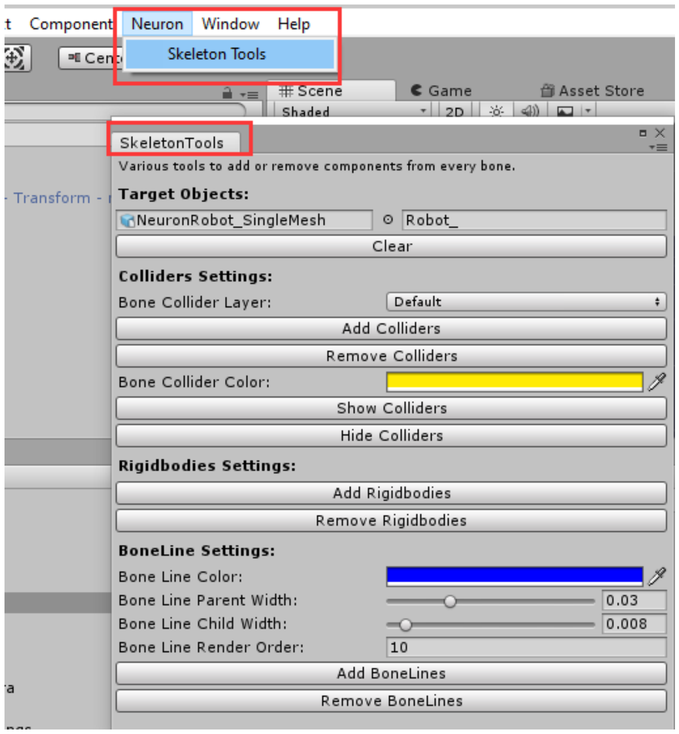
7 Click OK

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

4. Playback a recorderd 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.



SkeletonTools

Various tools to add or remove components from every bone.

Target Objects:

NeuronRobot_SingleMesh | Robot_

Clear

Colliders Settings:

Bone Collider Layer: Default

Add Colliders

Remove Colliders

Bone Collider Color:

Show Colliders

Hide Colliders

Rigidbodies Settings:

Add Rigidbodies

Remove Rigidbodies

BoneLine Settings:

Bone Line Color:

Bone Line Parent Width: 0.03

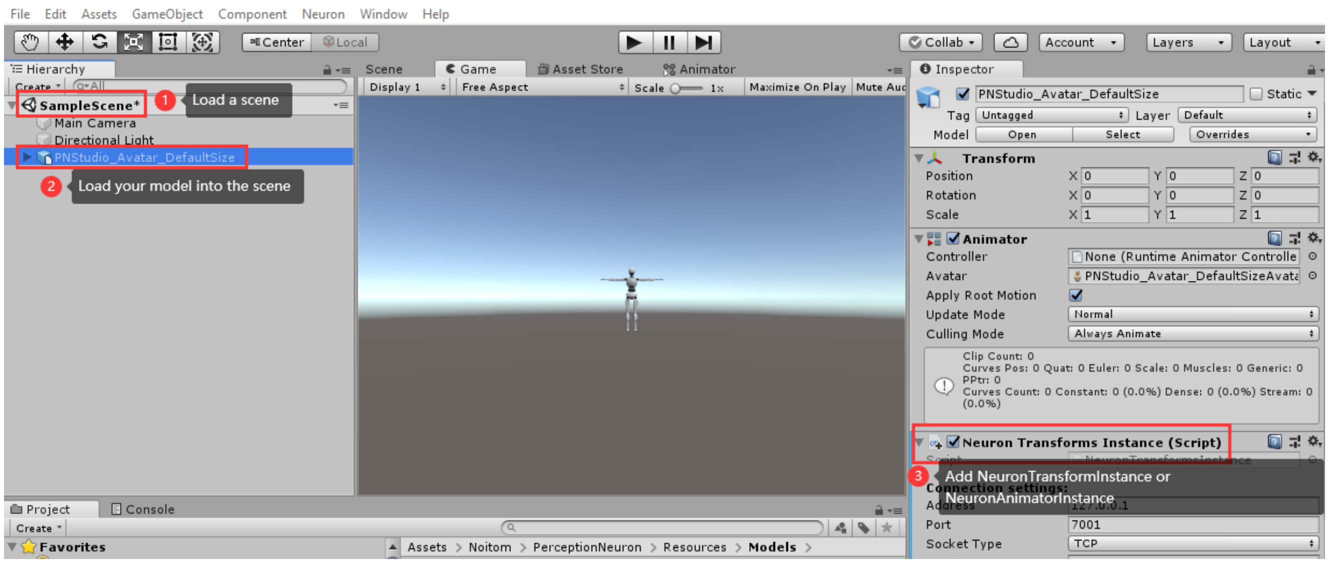
Bone Line Child Width: 0.008

Bone Line Render Order: 10

Add BoneLines

Remove BoneLines

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.



SampleScene*
Main Camera
Directional Light
PNStudio_Avatar_DefaultSize

1 Load a scene

2 Load your model into the scene

PNStudio_Avatar_DefaultSize Static

Tag Untagged Layer Default

Model Open Select Overrides

Transform

Position	X	0	Y	0	Z	0
Rotation	X	0	Y	0	Z	0
Scale	X	1	Y	1	Z	1

Animator

Controller: None (Runtime Animator Controlle

Avatar: PNStudio_Avatar_DefaultSizeAvata

Apply Root Motion: ✓

Update Mode: Normal

Culling Mode: Always Animate

Clip Count: 0
Curves Pos: 0 Quat: 0 Euler: 0 Scale: 0 Muscles: 0 Generic: 0 PPtr: 0
Curves Count: 0 Constant: 0 (0.0%) Dense: 0 (0.0%) Stream: 0 (0.0%)

Neuron Transforms Instance (Script)

Script: NeuronTransformsInstance

3 Add NeuronTransformInstance or NeuronAnimatorInstance

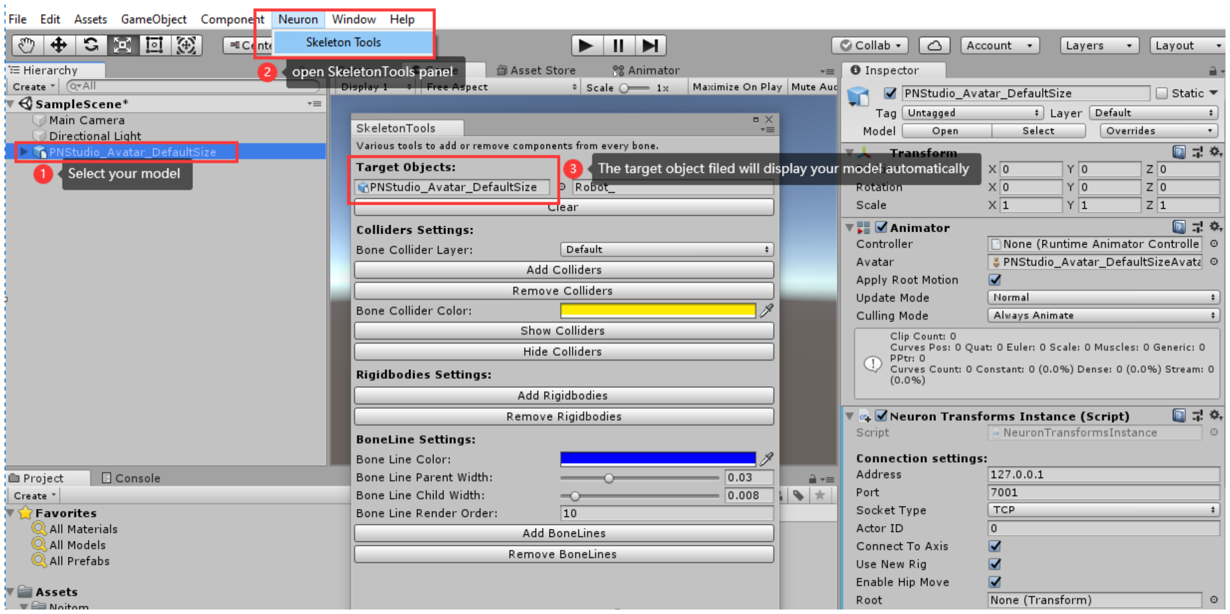
Connection settings:

Address: 127.0.0.1

Port: 7001

Socket Type: TCP

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


File Edit Assets GameObject Component Neuron Window Help
Skeleton Tools
1 Select your model
2 open SkeletonTools panel
3 The target object filed will display your model automatically
Hierarchy
SampleScene*
Main Camera
Directional Light
PNStudio_Avatar_DefaultSize
Project
Console
Favorites
All Materials
All Models
All Prefabs
Assets
Inspector

SkeletonTools
Various tools to add or remove components from every bone.

Target Objects: PNStudio_Avatar_DefaultSize | Robot_

Clear

Colliders Settings:

Bone Collider Layer:	Default
Bone Collider Color:	🟨

Add Colliders / Remove Colliders / Show Colliders / Hide Colliders

Rigidbodies Settings:

Add Rigidbodies / Remove Rigidbodies

BoneLine Settings:

Bone Line Color:	🟦
Bone Line Parent Width:	0.03
Bone Line Child Width:	0.008
Bone Line Render Order:	10

Add BoneLines / Remove BoneLines

Inspector: PNStudio_Avatar_DefaultSize, Tag Untagged, Layer Default

Transform

	X	Y	Z
Rotation	0	0	0
Scale	1	1	1

Animator

Controller	None (Runtime Animator Controlle
Avatar	PNStudio_Avatar_DefaultSizeAvata
Apply Root Motion	✓
Update Mode	Normal
Culling Mode	Always Animate

Clip Count: 0
Curves Pos: 0 Quat: 0 Euler: 0 Scale: 0 Muscles: 0 Generic: 0 PPtr: 0
Curves Count: 0 Constant: 0 (0.0%) Dense: 0 (0.0%) Stream: 0 (0.0%)

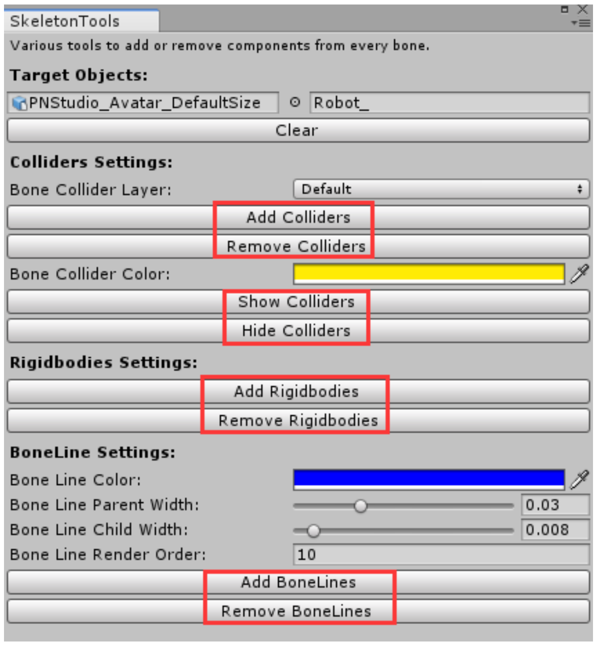
Neuron Transforms Instance (Script)

Script: NeuronTransformsInstance

Connection settings:

Address	127.0.0.1
Port	7001
Socket Type	TCP
Actor ID	0
Connect To Axis	✓
Use New Rig	✓
Enable Hip Move	✓
Root	None (Transform)

3. Now you can add/remove components for bones such as:
   1) Adding/removing of colliders.
   2) Showing/hiding of colliders.
   3) Adding/removing of rigidbody components.
   4) 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 ontop of eachother if you have multiple bone lines.



SkeletonTools
Various tools to add or remove components from every bone.

Target Objects:
PNStudio_Avatar_DefaultSize | Robot_

Clear

Colliders Settings:

Bone Collider Layer:	Default
Bone Collider Color:	🟨

Add Colliders
Remove Colliders
Show Colliders
Hide Colliders

Rigidbodies Settings:

Add Rigidbodies
Remove Rigidbodies

BoneLine Settings:

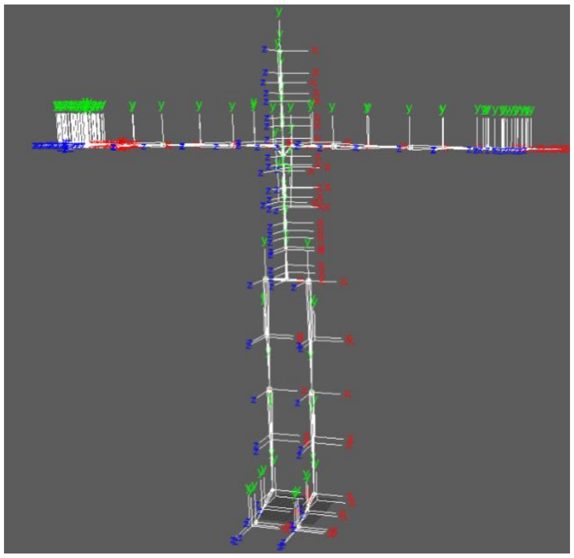
Bone Line Color:	🟦
Bone Line Parent Width:	0.03
Bone Line Child Width:	0.008
Bone Line Render Order:	10

Add BoneLines
Remove BoneLines

6. How to Configure a New Skeleton

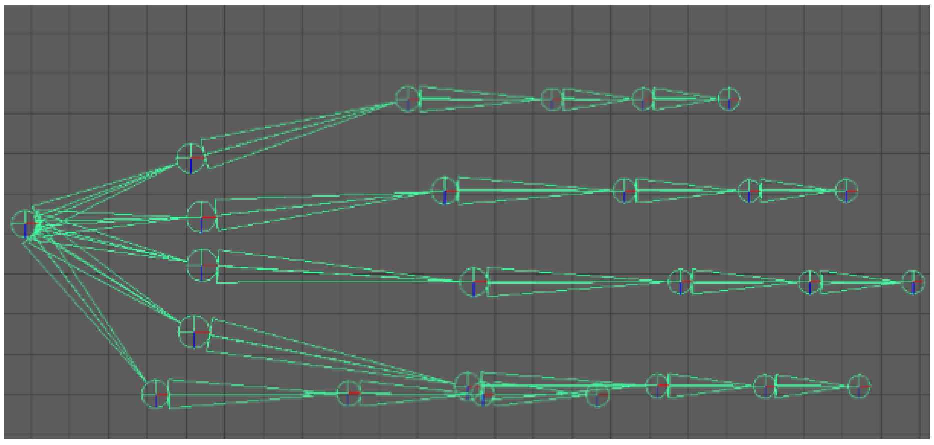
The following is a guide for how to configure a new character model to receive real-time motion data by Perception Neuron. Before we start you need to be aware of a few things:

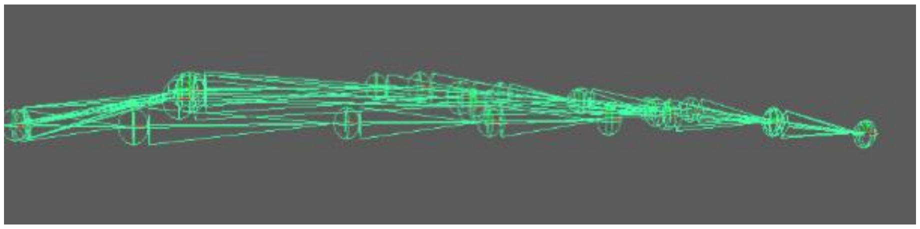
1. Your model needs to be rigged on a humanoid skeleton
2. The bones of your rig can not have any existing rotations in them
3. Your model needs to be rigged in a T-Pose
4. The rotation value of all bones in T pose is zero
5. Each bone's local rotation axes should be the same orientation as shown in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



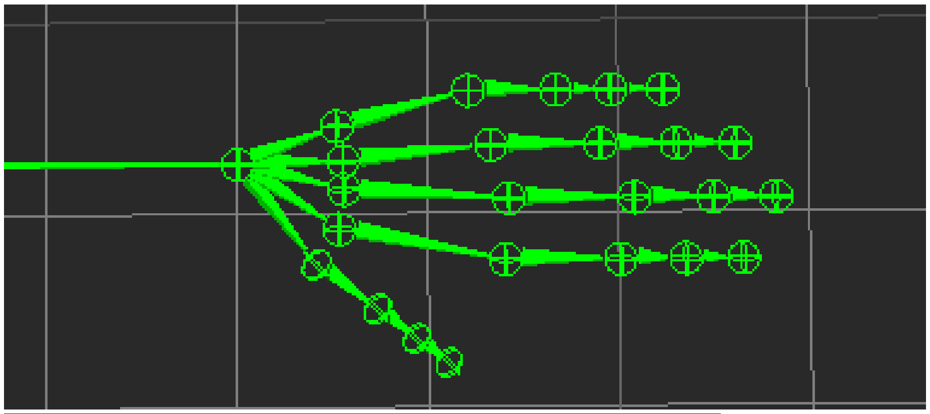
6. Fingers posture refer to the following two kinds, one is thumb inside the palm and the other one is thumb open.

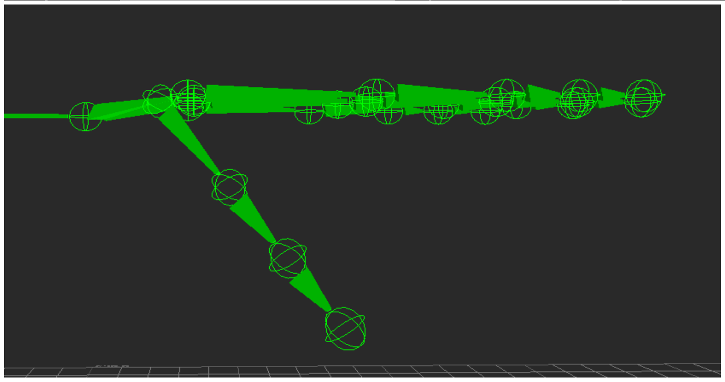
   1) Thumb inside the palm shown as in the following figures.



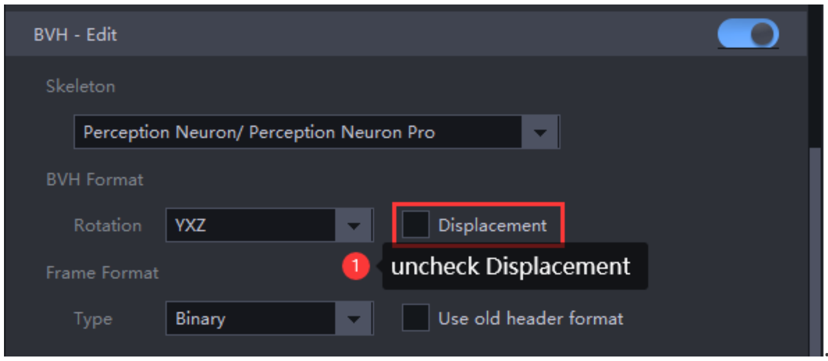


2) Thumb open shown as in the following figures.





7. We support two sets of Skeleton Structure PN/PN Pro and PN Studio. Please refer to [APPENDIX B: SKELETON BONES]()
8. 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.



BVH - Edit

Skeleton: Perception Neuron/ Perception Neuron Pro

BVH Format

Rotation: YXZ ☐ Displacement

1 uncheck Displacement

Frame Format

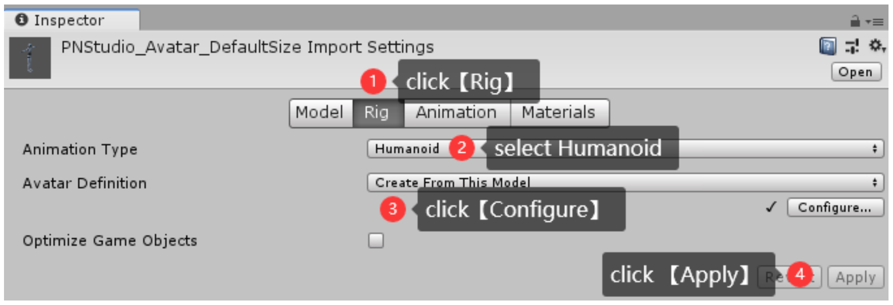
Type: Binary ☐ Use old header format

.

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. Import your Humanoid skeleton model into Unity.
3. 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】.



Inspector

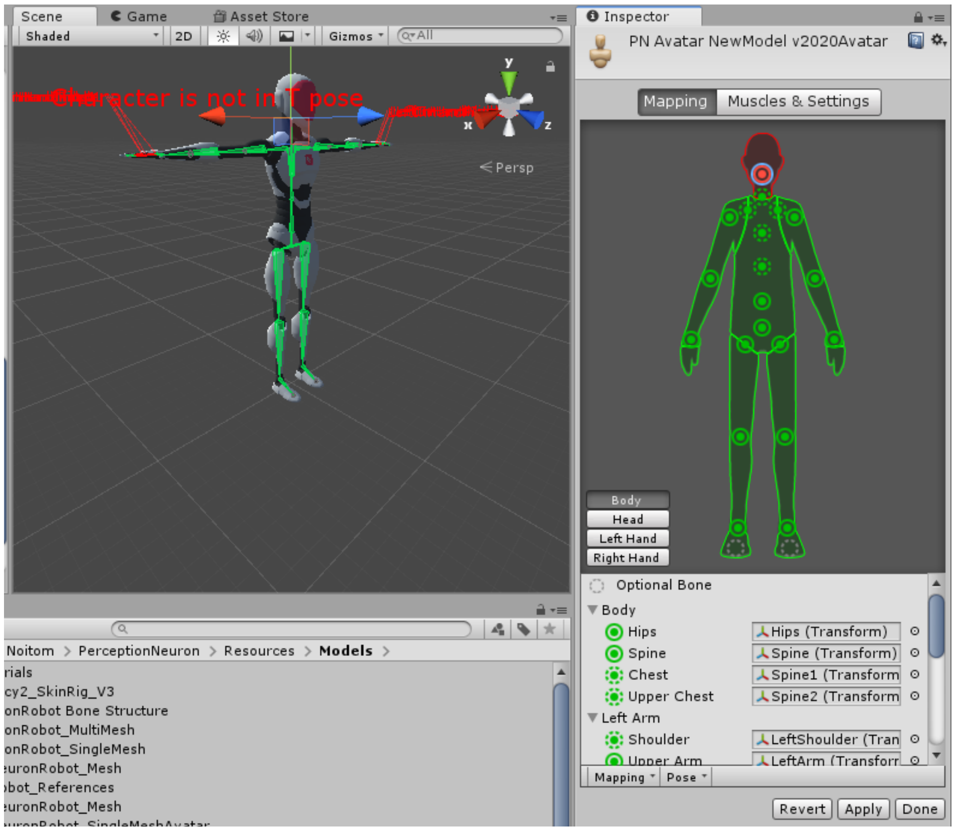
PNStudio_Avatar_DefaultSize Import Settings

Animation Type: Humanoid

Avatar Definition: Create From This Model

Optimize Game Objects: ☐

4. 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.



Inspector

PN Avatar NewModel v2020Avatar

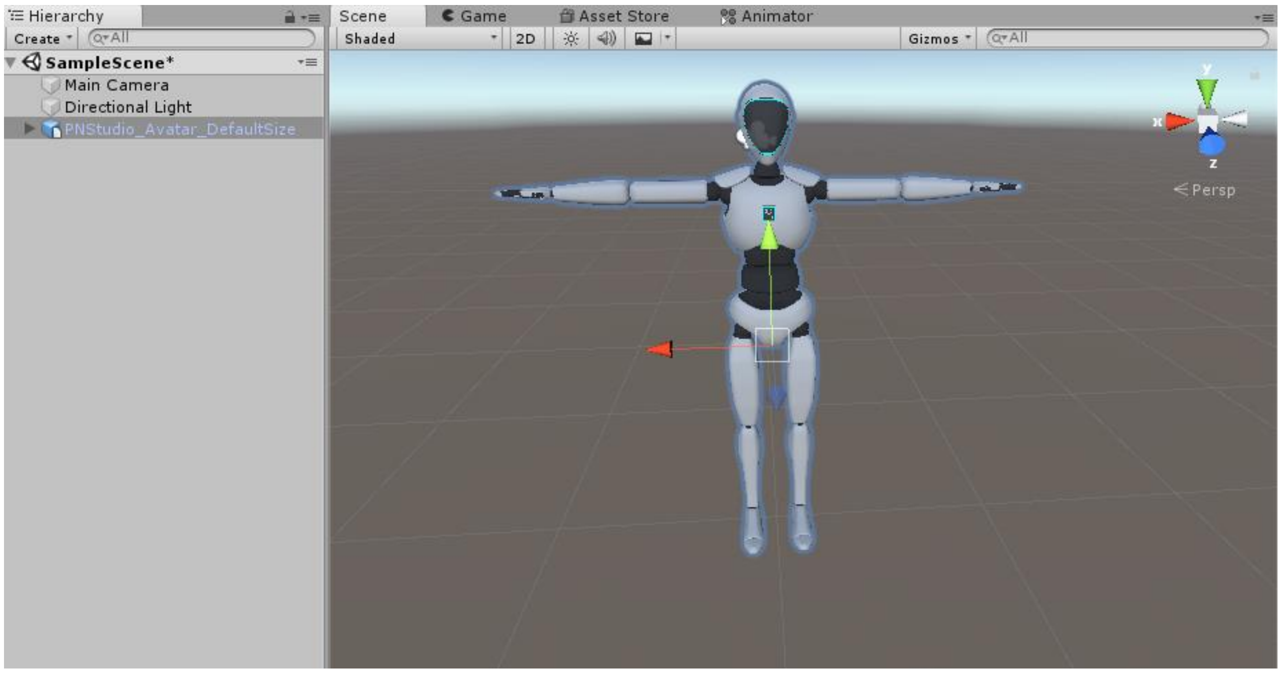
Hips	Hips (Transform)
Spine	Spine (Transform)
Chest	Spine1 (Transform
Upper Chest	Spine2 (Transform
Shoulder	LeftShoulder (Tran
Upper Arm	LeftArm (Transforr

Noitom > PerceptionNeuron > Resources > **Models** >

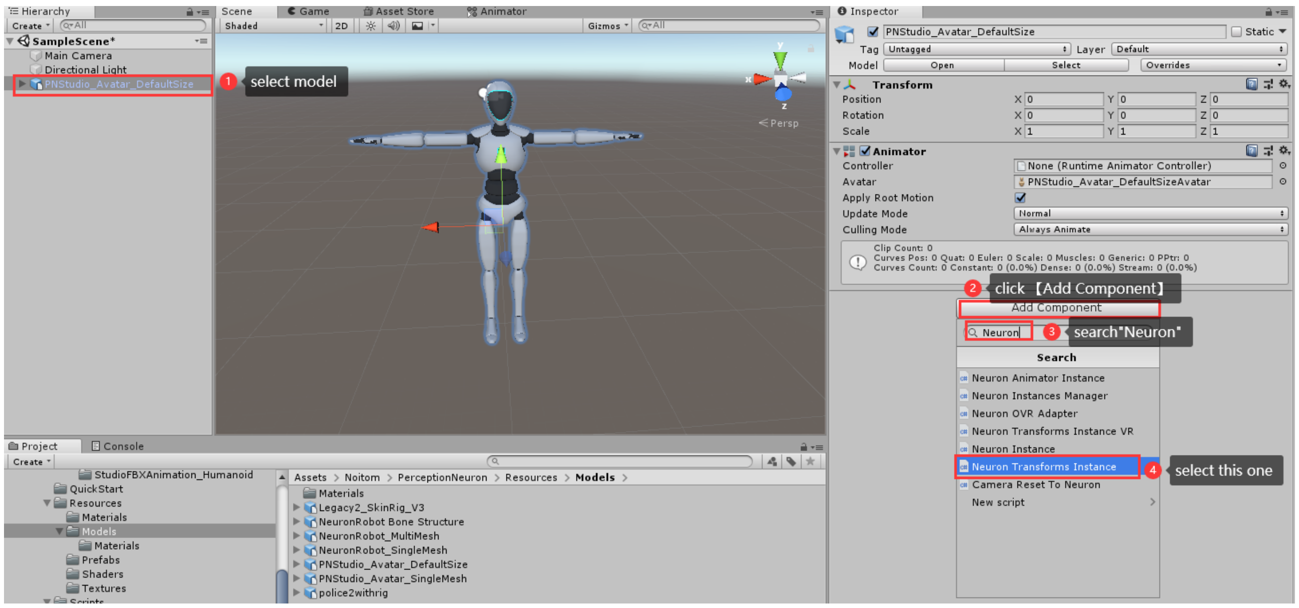
rials
cy2_SkinRig_V3
onRobot Bone Structure
onRobot_MultiMesh
onRobot_SingleMesh
euronRobot_Mesh
obot_References
euronRobot_Mesh

5. 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
   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】

6. 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.



7. 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”.



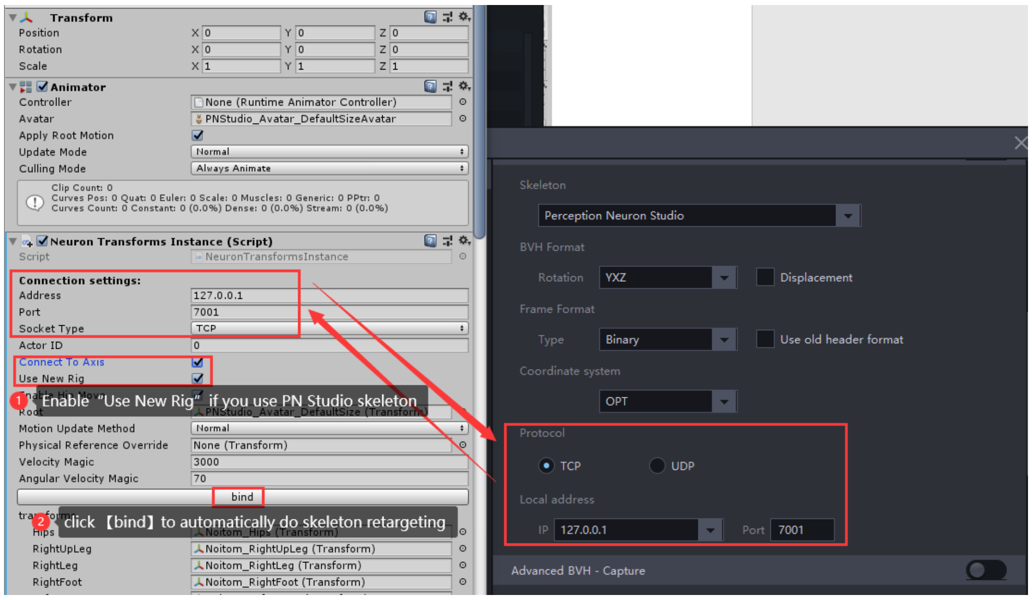
PNStudio_Avatar_DefaultSize
Tag Untagged Layer Default
Model Open Select Overrides

Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

Animator
Controller None (Runtime Animator Controller)
Avatar PNStudio_Avatar_DefaultSizeAvatar
Apply Root Motion ✓
Update Mode Normal
Culling Mode Always Animate

Clip Count: 0
Curves Pos: 0 Quat: 0 Euler: 0 Scale: 0 Muscles: 0 Generic: 0 PPtr: 0
Curves Count: 0 Constant: 0 (0.0%) Dense: 0 (0.0%) Stream: 0 (0.0%)

8. 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.


Transform

Position	X 0	Y 0	Z 0
Rotation	X 0	Y 0	Z 0
Scale	X 1	Y 1	Z 1

Animator

Controller	None (Runtime Animator Controller)
Avatar	PNStudio_Avatar_DefaultSizeAvatar
Apply Root Motion	✓
Update Mode	Normal
Culling Mode	Always Animate

Clip Count: 0
Curves Pos: 0 Quat: 0 Euler: 0 Scale: 0 Muscles: 0 Generic: 0 PPtr: 0
Curves Count: 0 Constant: 0 (0.0%) Dense: 0 (0.0%) Stream: 0 (0.0%)

Neuron Transforms Instance (Script)

Script	NeuronTransformsInstance
Connection settings:	
Address	127.0.0.1
Port	7001
Socket Type	TCP
Actor ID	0
Connect To Axis	✓
Use New Rig	✓
Enable Hip Move	✓
Root	PNStudio_Avatar_DefaultSize (Transform)
Motion Update Method	Normal
Physical Reference Override	None (Transform)
Velocity Magic	3000
Angular Velocity Magic	70

bind

transforms

Hips	Noitom_Hips (Transform)
RightUpLeg	Noitom_RightUpLeg (Transform)
RightLeg	Noitom_RightLeg (Transform)
RightFoot	Noitom_RightFoot (Transform)

1 Enable "Use New Rig" if you use PN Studio skeleton

2 click 【bind】to automatically do skeleton retargeting

Skeleton: Perception Neuron Studio

BVH Format

Rotation: YXZ — Displacement: ☐

Frame Format

Type: Binary — Use old header format: ☐

Coordinate system: OPT

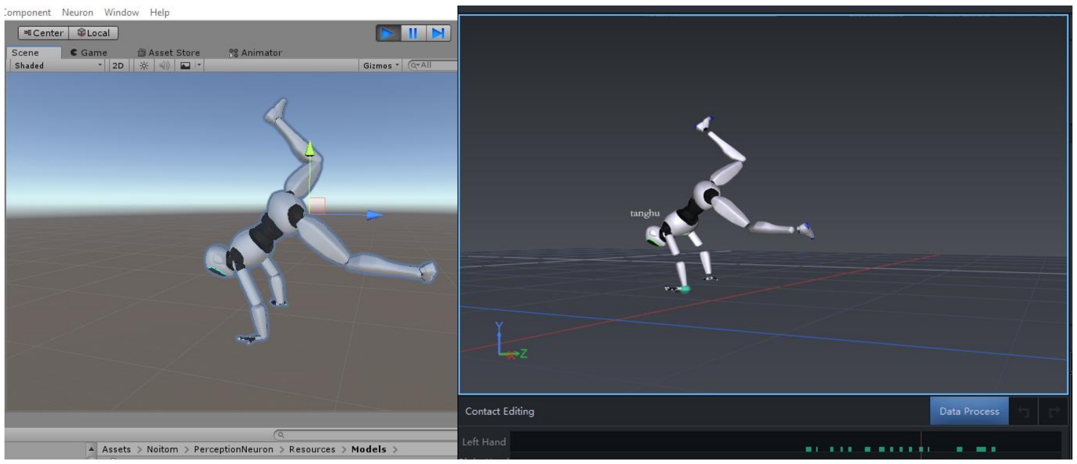
Protocol: [● TCP] ○ UDP

Local address

IP: 127.0.0.1 — Port: 7001

Advanced BVH - Capture

9. 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 Perception Neuron 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 Perception Neuron system.
2. Don't use the positional tracking of the HMD.
3. Never parent the HMD cameras or GameObjects to the skeleton setup.

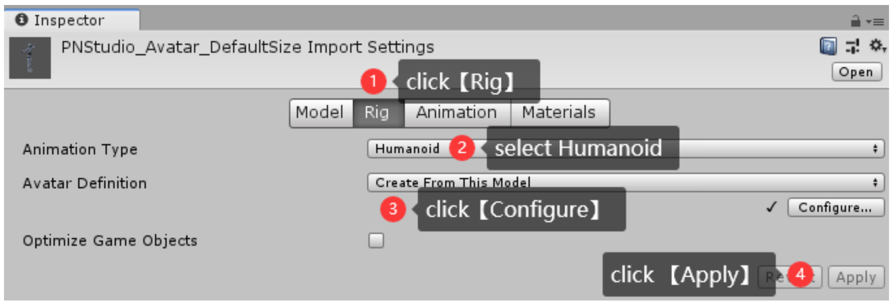
Inside the examples folder there is an example called: VRExample. It contains a simple script and a scene showing you how to combine our system with any virtual reality HMD. The example script works in the following ways:

1. On every new frame we position the VR Camera Rig to the same position as the head target object.
2. This target object is an empty GameObject inside the Head skeleton hierarchy. We use the target object to provide an easy way to define an offset and to set the correct position on the head.
3. Since we never change the rotation of the VR Rig you need to reset its tracker once you have the HMD on your head. This way the rig will be aligned correctly with your virtual body. Make sure you're facing forward in a T-Pose manner when resetting the VR tracking. Press the R key on your keyboard to do so.

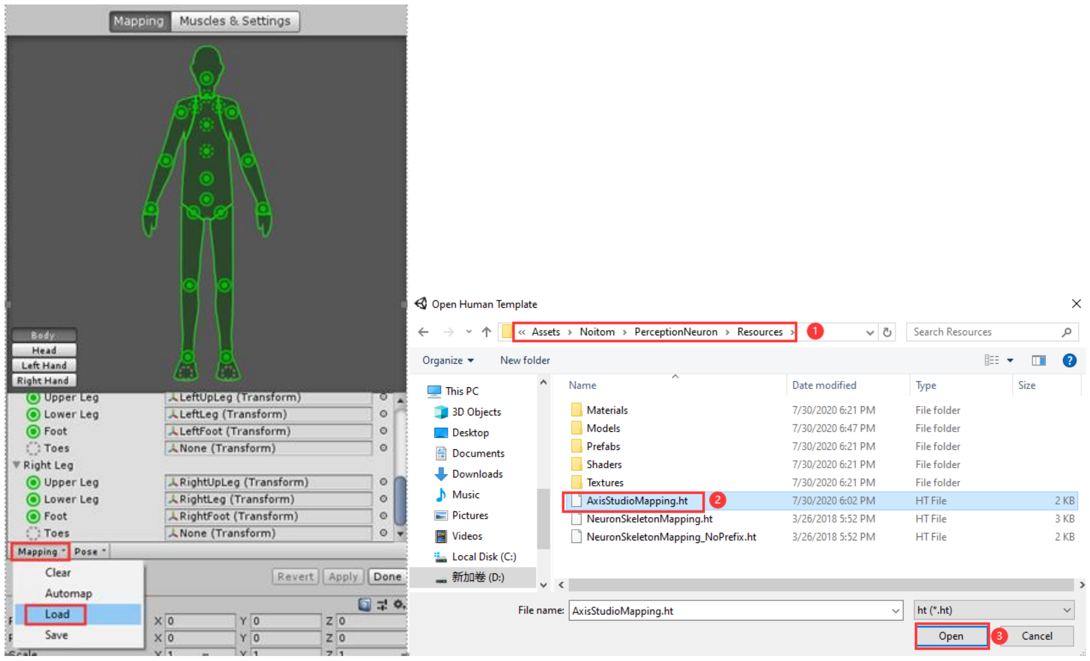
9.How to Apply FBX file Data from Axis

9.1 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】.



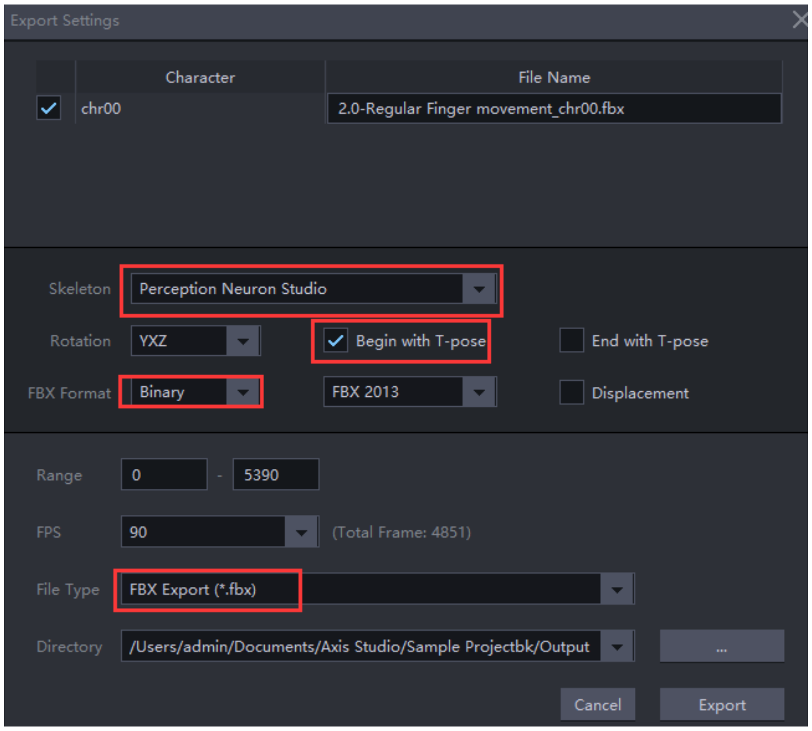
2. Next click on 【Configure】 to check the bone mapping. Usually Unity will find the right bone references for almost all the humanoid skeleton so its good check them. You can load pre-make .ht mapping file or manually do the mapping.



3. Click【Apply】

4. 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.

5. Export a fbx file from Axis Neuron/Axis Studio. In this case the export setting in Axis Studio is shown as the figure.



Export Settings

	Character	File Name
✓	chr00	2.0-Regular Finger movement_chr00.fbx

Skeleton: Perception Neuron Studio

Rotation: YXZ ✓ Begin with T-pose ☐ End with T-pose

FBX Format: Binary FBX 2013 ☐ Displacement

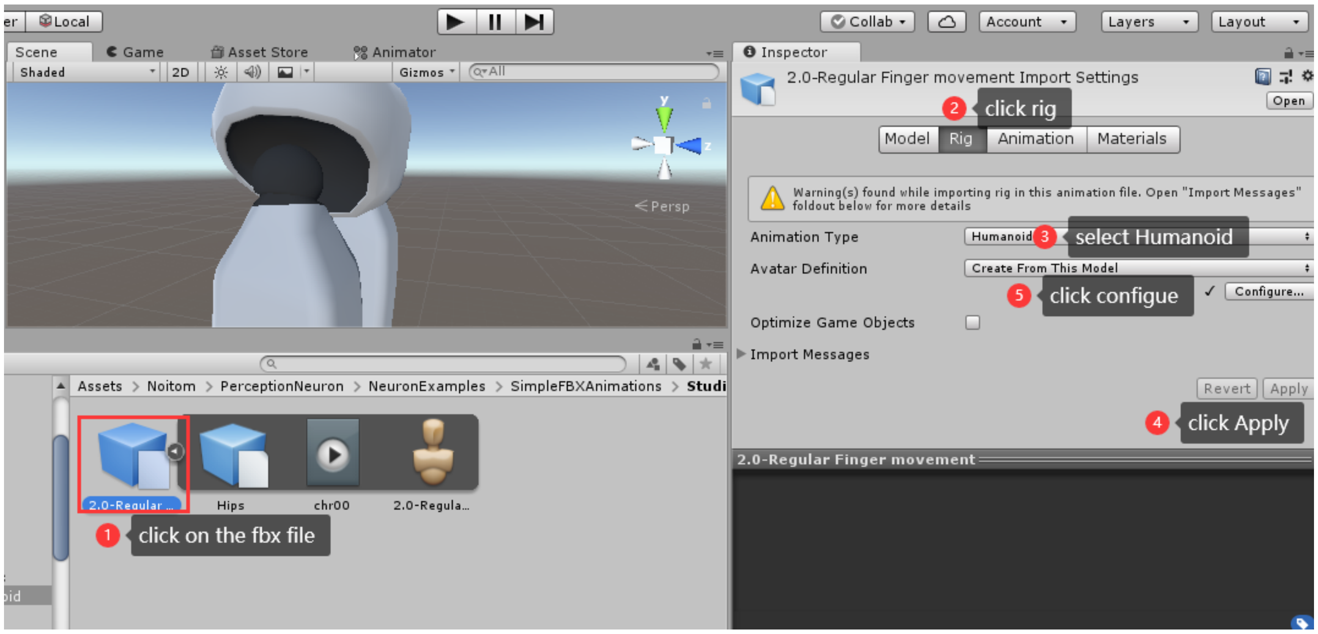
Range: 0 - 5390

FPS: 90 (Total Frame: 4851)

File Type: FBX Export (*.fbx)

Directory: /Users/admin/Documents/Axis Studio/Sample Projectbk/Output

6. 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】.



Inspector

2.0-Regular Finger movement Import Settings

Warning(s) found while importing rig in this animation file. Open "Import Messages" foldout below for more details

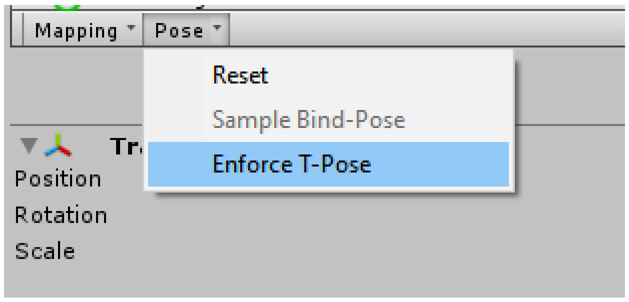
Animation Type: Humanoid

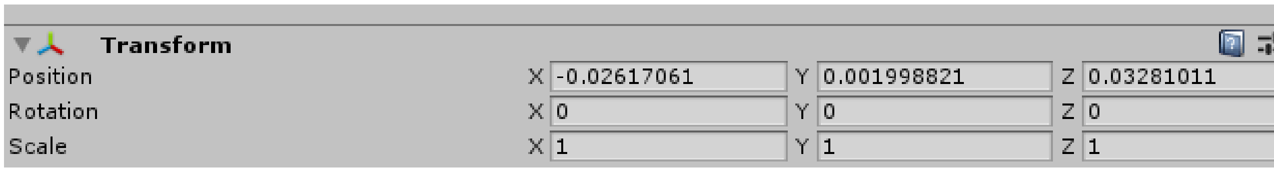
Avatar Definition: Create From This Model

Optimize Game Objects ☐

Assets > Noitom > PerceptionNeuron > NeuronExamples > SimpleFBXAnimations > Studi

7. 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.

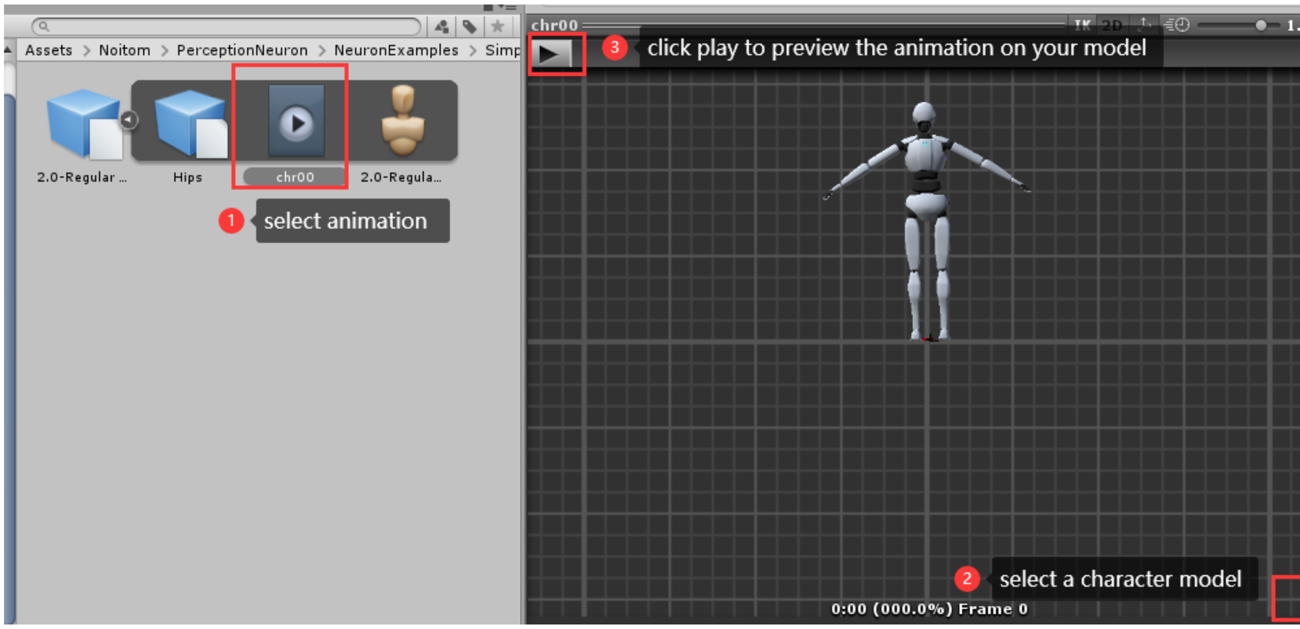


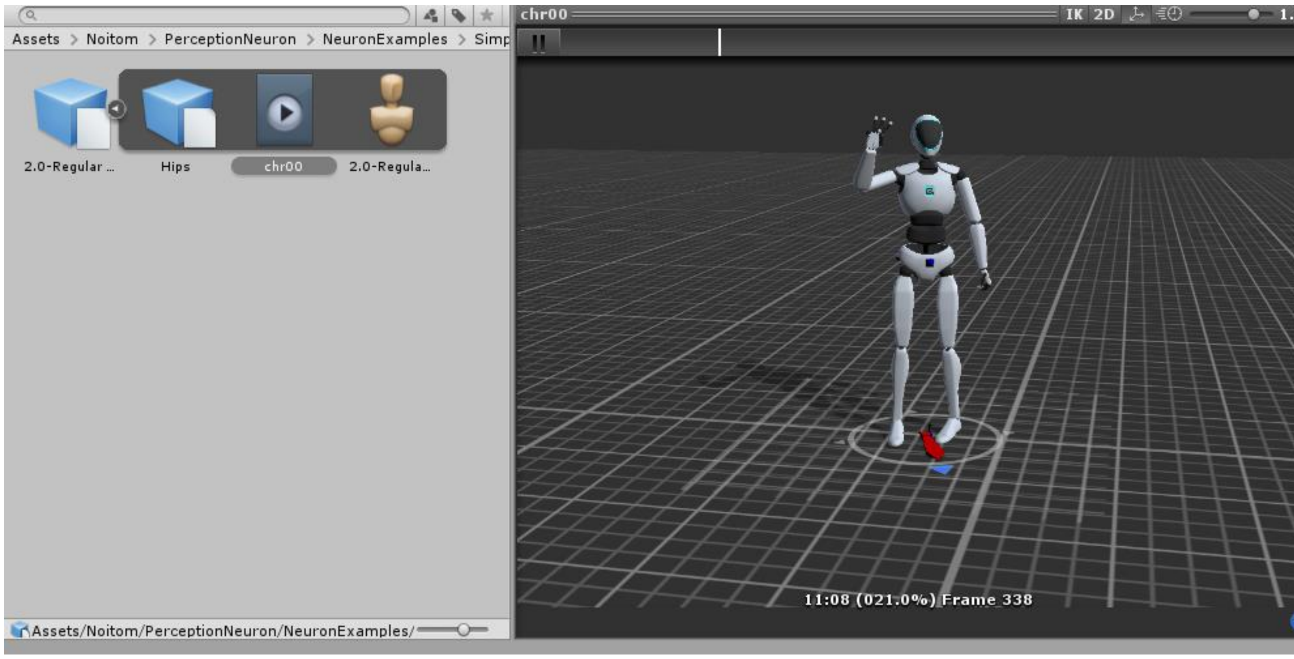

Transform

	X	Y	Z
Position	-0.02617061	0.001998821	0.03281011
Rotation	0	0	0
Scale	1	1	1

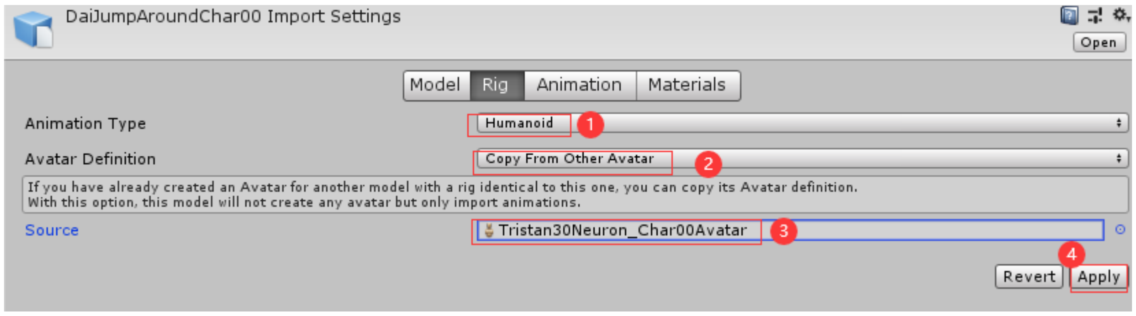
8. Click on 【Apply】and then on 【Done】

9. You can preview the animation on your model


Assets > Noitom > PerceptionNeuron > NeuronExamples > Simp
2.0-Regular ...
Hips
chr00
2.0-Regula...
1 select animation
chr00
3 click play to preview the animation on your model
2 select a character model
0:00 (000.0%) Frame 0


Assets > Noitom > PerceptionNeuron > NeuronExamples > Simp
2.0-Regular ...
Hips
chr00
2.0-Regula...
chr00
IK 2D
11:08 (021.0%) Frame 338
Assets/Noitom/PerceptionNeuron/NeuronExamples/

Note：If we import a fbx animation file exported from AxisNeuron, we can set Rig >Avatar Definition as " Copy From Other Avatar" and select "Tristan30Neuron_Char00Avatar" as source.


DaiJumpAroundChar00 Import Settings

Animation Type: Humanoid

Avatar Definition: Copy From Other Avatar

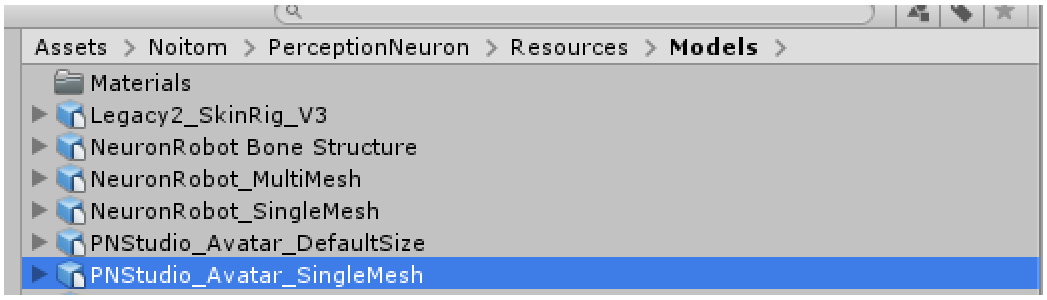
If you have already created an Avatar for another model with a rig identical to this one, you can copy its Avatar definition.
With this option, this model will not create any avatar but only import animations.

Source: Tristan30Neuron_Char00Avatar

9.2 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.

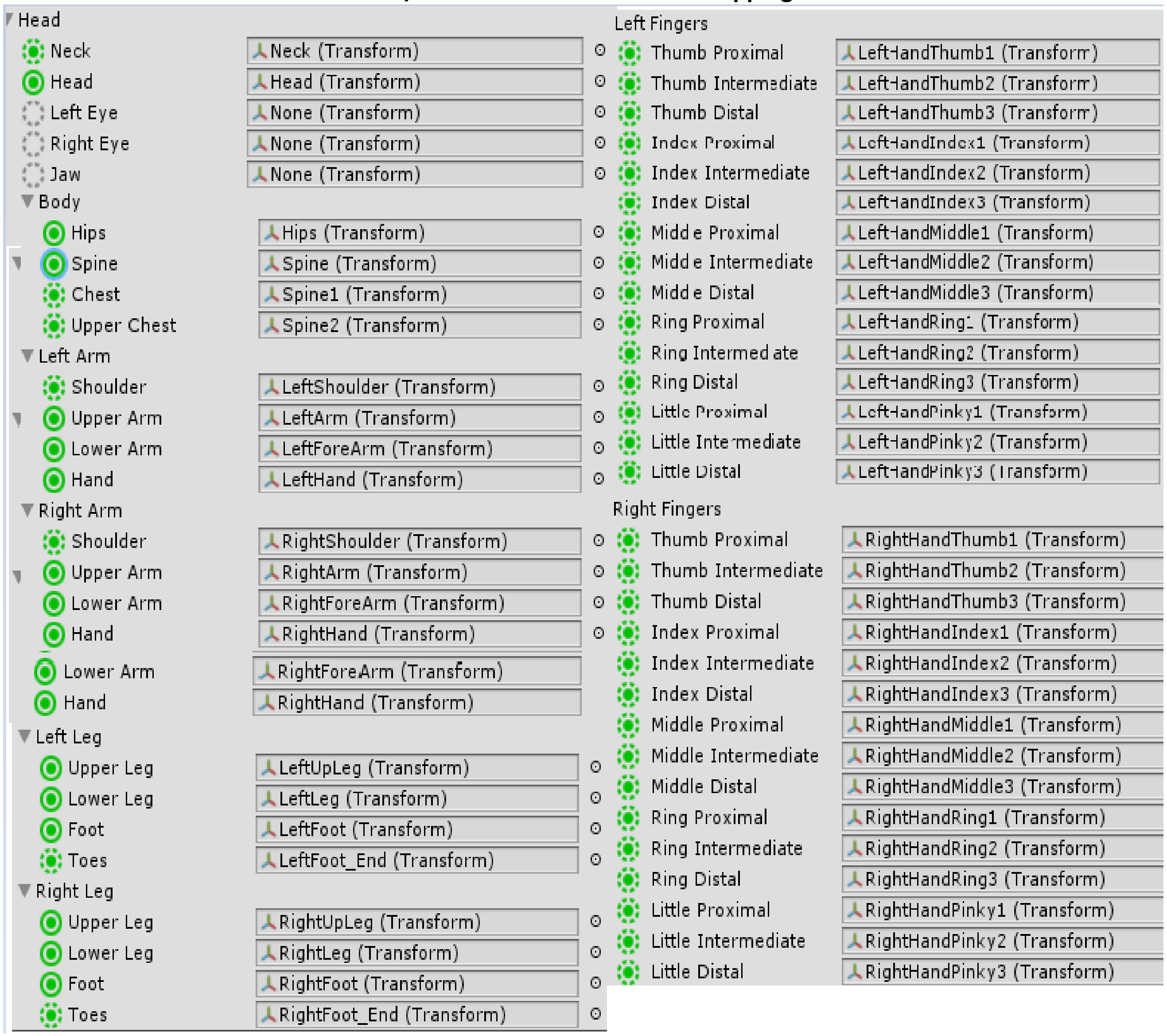
Refer to Noitom/PerceptionNeuron/Resources/Models/PNStudio_Avatar_SingleMesh.fbx


Assets > Noitom > PerceptionNeuron > Resources > Models >

- Materials
- Legacy2_SkinRig_V3
- NeuronRobot Bone Structure
- NeuronRobot_MultiMesh
- NeuronRobot_SingleMesh
- PNStudio_Avatar_DefaultSize
- PNStudio_Avatar_SingleMesh

APPENDIX A: SKELETON MAPPING

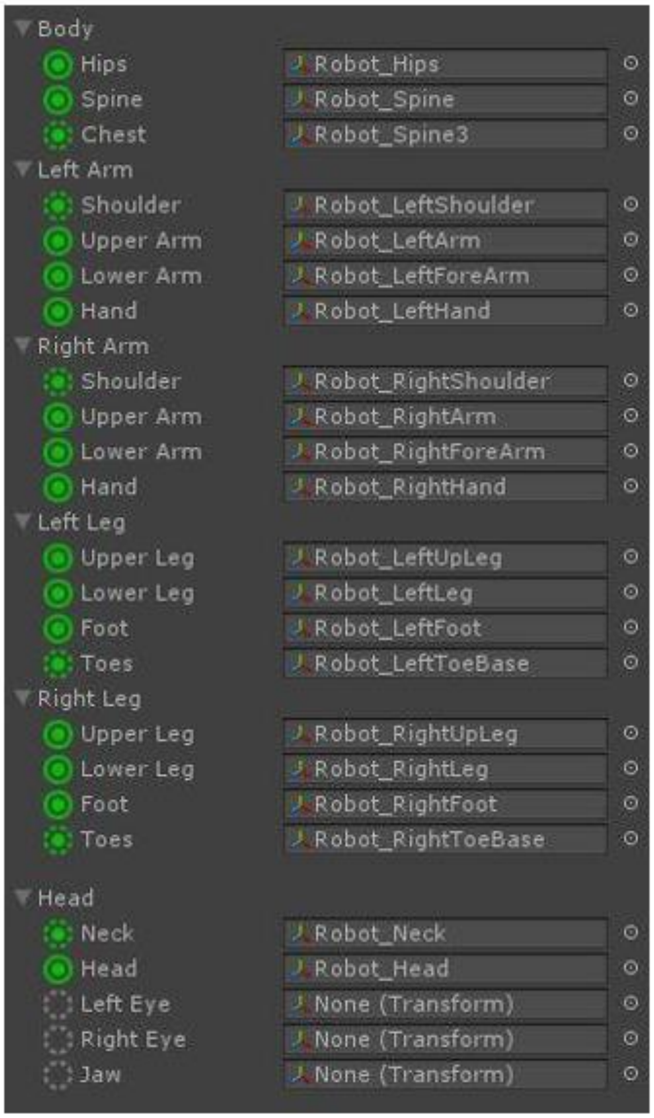
PN/PN Pro skeleton structure mapping



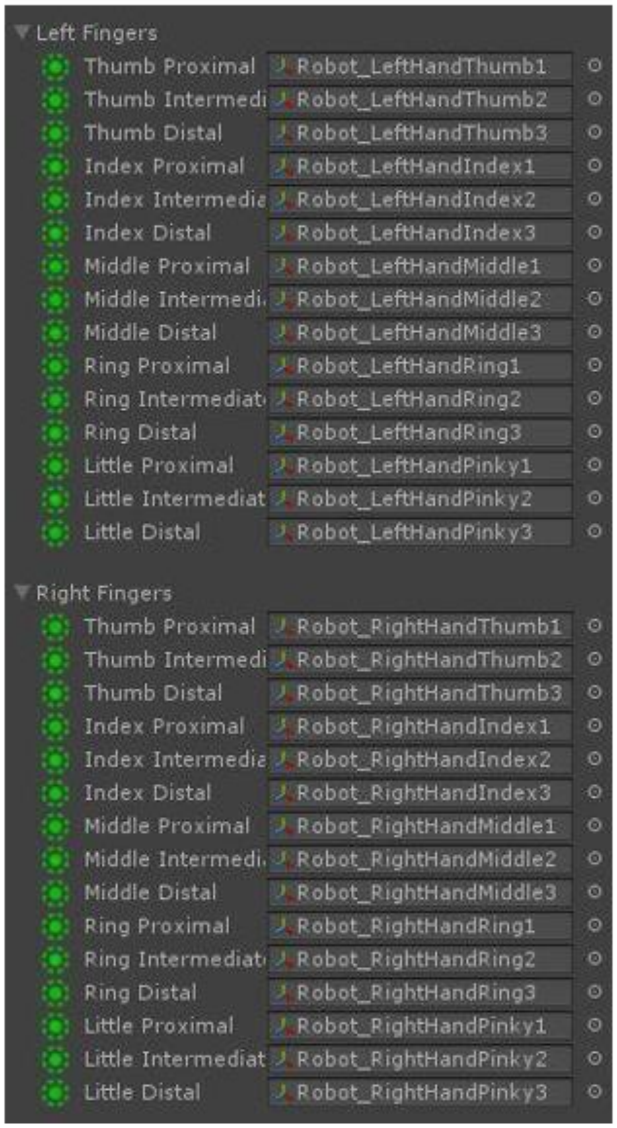
Head	
Neck	Neck (Transform)
Head	Head (Transform)
Left Eye	None (Transform)
Right Eye	None (Transform)
Jaw	None (Transform)
Body	
Hips	Hips (Transform)
Spine	Spine (Transform)
Chest	Spine1 (Transform)
Upper Chest	Spine2 (Transform)
Left Arm	
Shoulder	LeftShoulder (Transform)
Upper Arm	LeftArm (Transform)
Lower Arm	LeftForeArm (Transform)
Hand	LeftHand (Transform)
Right Arm	
Shoulder	RightShoulder (Transform)
Upper Arm	RightArm (Transform)
Lower Arm	RightForeArm (Transform)
Hand	RightHand (Transform)
Lower Arm	RightForeArm (Transform)
Hand	RightHand (Transform)
Left Leg	
Upper Leg	LeftUpLeg (Transform)
Lower Leg	LeftLeg (Transform)
Foot	LeftFoot (Transform)
Toes	LeftFoot_End (Transform)
Right Leg	
Upper Leg	RightUpLeg (Transform)
Lower Leg	RightLeg (Transform)
Foot	RightFoot (Transform)
Toes	RightFoot_End (Transform)

Left Fingers	
Thumb Proximal	LeftHandThumb1 (Transform)
Thumb Intermediate	LeftHandThumb2 (Transform)
Thumb Distal	LeftHandThumb3 (Transform)
Index Proximal	LeftHandIndex1 (Transform)
Index Intermediate	LeftHandIndex2 (Transform)
Index Distal	LeftHandIndex3 (Transform)
Middle Proximal	LeftHandMiddle1 (Transform)
Middle Intermediate	LeftHandMiddle2 (Transform)
Middle Distal	LeftHandMiddle3 (Transform)
Ring Proximal	LeftHandRing1 (Transform)
Ring Intermediate	LeftHandRing2 (Transform)
Ring Distal	LeftHandRing3 (Transform)
Little Proximal	LeftHandPinky1 (Transform)
Little Intermediate	LeftHandPinky2 (Transform)
Little Distal	LeftHandPinky3 (Transform)
Right Fingers	
Thumb Proximal	RightHandThumb1 (Transform)
Thumb Intermediate	RightHandThumb2 (Transform)
Thumb Distal	RightHandThumb3 (Transform)
Index Proximal	RightHandIndex1 (Transform)
Index Intermediate	RightHandIndex2 (Transform)
Index Distal	RightHandIndex3 (Transform)
Middle Proximal	RightHandMiddle1 (Transform)
Middle Intermediate	RightHandMiddle2 (Transform)
Middle Distal	RightHandMiddle3 (Transform)
Ring Proximal	RightHandRing1 (Transform)
Ring Intermediate	RightHandRing2 (Transform)
Ring Distal	RightHandRing3 (Transform)
Little Proximal	RightHandPinky1 (Transform)
Little Intermediate	RightHandPinky2 (Transform)
Little Distal	RightHandPinky3 (Transform)

PN Studio skeleton structure mapping

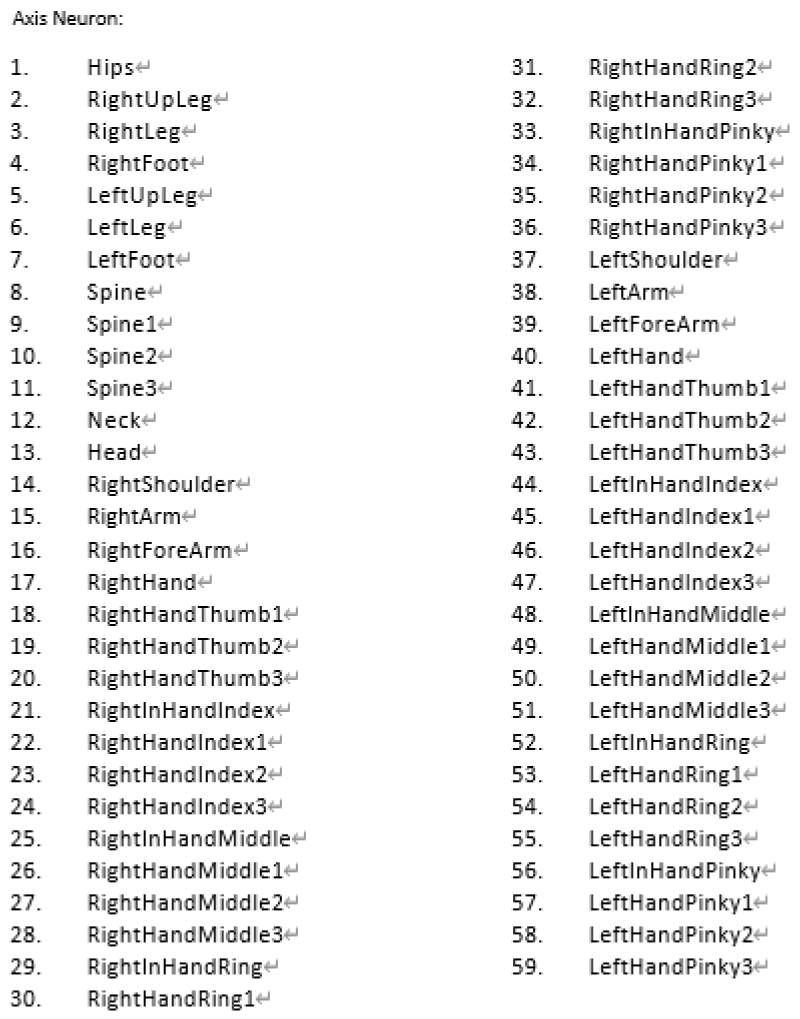


Body	
Hips	Robot_Hips
Spine	Robot_Spine
Chest	Robot_Spine3
Left Arm	
Shoulder	Robot_LeftShoulder
Upper Arm	Robot_LeftArm
Lower Arm	Robot_LeftForeArm
Hand	Robot_LeftHand
Right Arm	
Shoulder	Robot_RightShoulder
Upper Arm	Robot_RightArm
Lower Arm	Robot_RightForeArm
Hand	Robot_RightHand
Left Leg	
Upper Leg	Robot_LeftUpLeg
Lower Leg	Robot_LeftLeg
Foot	Robot_LeftFoot
Toes	Robot_LeftToeBase
Right Leg	
Upper Leg	Robot_RightUpLeg
Lower Leg	Robot_RightLeg
Foot	Robot_RightFoot
Toes	Robot_RightToeBase
Head	
Neck	Robot_Neck
Head	Robot_Head
Left Eye	None (Transform)
Right Eye	None (Transform)
Jaw	None (Transform)



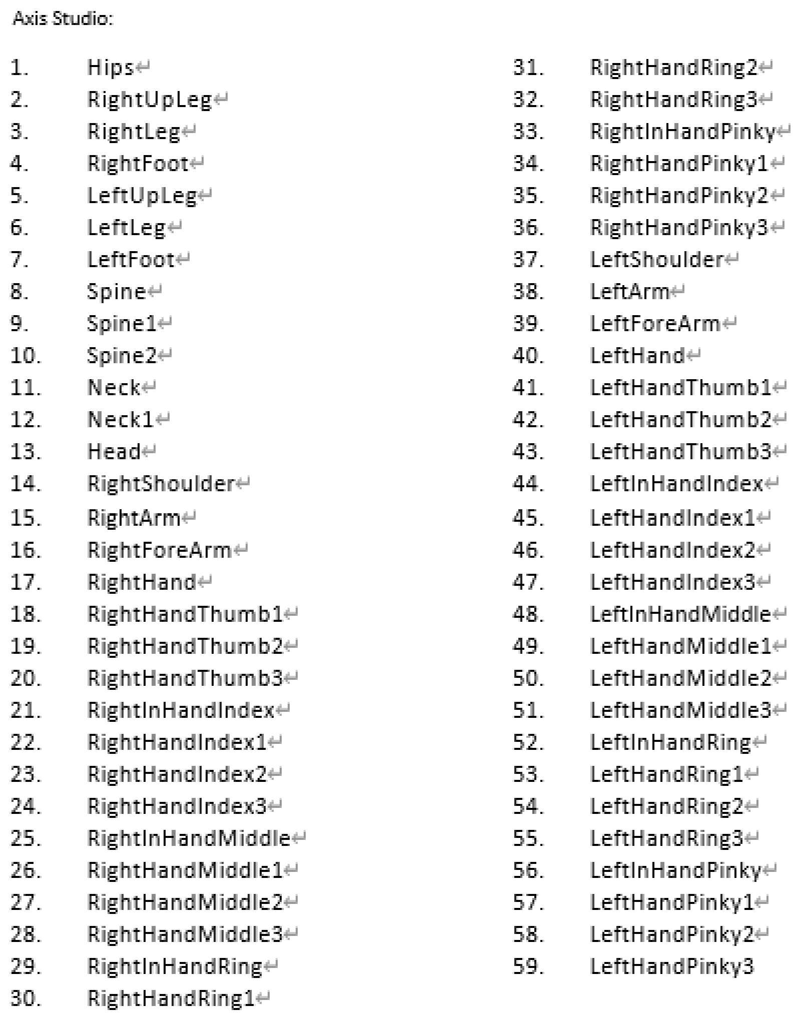
Left Fingers	
Thumb Proximal	Robot_LeftHandThumb1
Thumb Intermedi	Robot_LeftHandThumb2
Thumb Distal	Robot_LeftHandThumb3
Index Proximal	Robot_LeftHandIndex1
Index Intermedia	Robot_LeftHandIndex2
Index Distal	Robot_LeftHandIndex3
Middle Proximal	Robot_LeftHandMiddle1
Middle Intermedi	Robot_LeftHandMiddle2
Middle Distal	Robot_LeftHandMiddle3
Ring Proximal	Robot_LeftHandRing1
Ring Intermediat	Robot_LeftHandRing2
Ring Distal	Robot_LeftHandRing3
Little Proximal	Robot_LeftHandPinky1
Little Intermediat	Robot_LeftHandPinky2
Little Distal	Robot_LeftHandPinky3
Right Fingers	
Thumb Proximal	Robot_RightHandThumb1
Thumb Intermedi	Robot_RightHandThumb2
Thumb Distal	Robot_RightHandThumb3
Index Proximal	Robot_RightHandIndex1
Index Intermedia	Robot_RightHandIndex2
Index Distal	Robot_RightHandIndex3
Middle Proximal	Robot_RightHandMiddle1
Middle Intermedi	Robot_RightHandMiddle2
Middle Distal	Robot_RightHandMiddle3
Ring Proximal	Robot_RightHandRing1
Ring Intermediat	Robot_RightHandRing2
Ring Distal	Robot_RightHandRing3
Little Proximal	Robot_RightHandPinky1
Little Intermediat	Robot_RightHandPinky2
Little Distal	Robot_RightHandPinky3

APPENDIX B: SKELETON BONES



Axis Neuron:

1. Hips
2. RightUpLeg
3. RightLeg
4. RightFoot
5. LeftUpLeg
6. LeftLeg
7. LeftFoot
8. Spine
9. Spine1
10. Spine2
11. Spine3
12. Neck
13. Head
14. RightShoulder
15. RightArm
16. RightForeArm
17. RightHand
18. RightHandThumb1
19. RightHandThumb2
20. RightHandThumb3
21. RightInHandIndex
22. RightHandIndex1
23. RightHandIndex2
24. RightHandIndex3
25. RightInHandMiddle
26. RightHandMiddle1
27. RightHandMiddle2
28. RightHandMiddle3
29. RightInHandRing
30. RightHandRing1
31. RightHandRing2
32. RightHandRing3
33. RightInHandPinky
34. RightHandPinky1
35. RightHandPinky2
36. RightHandPinky3
37. LeftShoulder
38. LeftArm
39. LeftForeArm
40. LeftHand
41. LeftHandThumb1
42. LeftHandThumb2
43. LeftHandThumb3
44. LeftInHandIndex
45. LeftHandIndex1
46. LeftHandIndex2
47. LeftHandIndex3
48. LeftInHandMiddle
49. LeftHandMiddle1
50. LeftHandMiddle2
51. LeftHandMiddle3
52. LeftInHandRing
53. LeftHandRing1
54. LeftHandRing2
55. LeftHandRing3
56. LeftInHandPinky
57. LeftHandPinky1
58. LeftHandPinky2
59. LeftHandPinky3


Axis Studio:

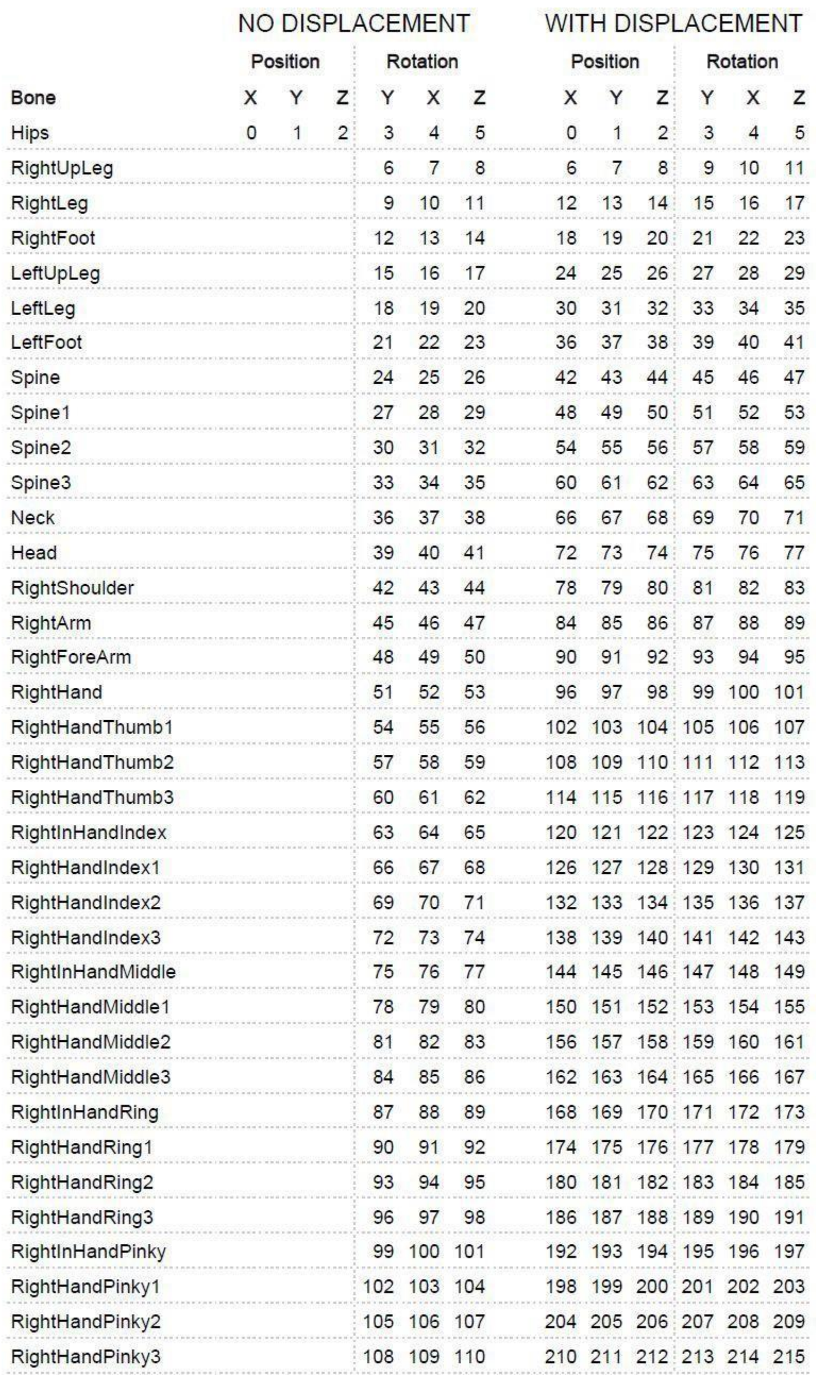
1. Hips
2. RightUpLeg
3. RightLeg
4. RightFoot
5. LeftUpLeg
6. LeftLeg
7. LeftFoot
8. Spine
9. Spine1
10. Spine2
11. Neck
12. Neck1
13. Head
14. RightShoulder
15. RightArm
16. RightForeArm
17. RightHand
18. RightHandThumb1
19. RightHandThumb2
20. RightHandThumb3
21. RightInHandIndex
22. RightHandIndex1
23. RightHandIndex2
24. RightHandIndex3
25. RightInHandMiddle
26. RightHandMiddle1
27. RightHandMiddle2
28. RightHandMiddle3
29. RightInHandRing
30. RightHandRing1
31. RightHandRing2
32. RightHandRing3
33. RightInHandPinky
34. RightHandPinky1
35. RightHandPinky2
36. RightHandPinky3
37. LeftShoulder
38. LeftArm
39. LeftForeArm
40. LeftHand
41. LeftHandThumb1
42. LeftHandThumb2
43. LeftHandThumb3
44. LeftInHandIndex
45. LeftHandIndex1
46. LeftHandIndex2
47. LeftHandIndex3
48. LeftInHandMiddle
49. LeftHandMiddle1
50. LeftHandMiddle2
51. LeftHandMiddle3
52. LeftInHandRing
53. LeftHandRing1
54. LeftHandRing2
55. LeftHandRing3
56. LeftInHandPinky
57. LeftHandPinky1
58. LeftHandPinky2
59. LeftHandPinky3

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.

APPENDIX C: BINARY DATA SEQUENCE

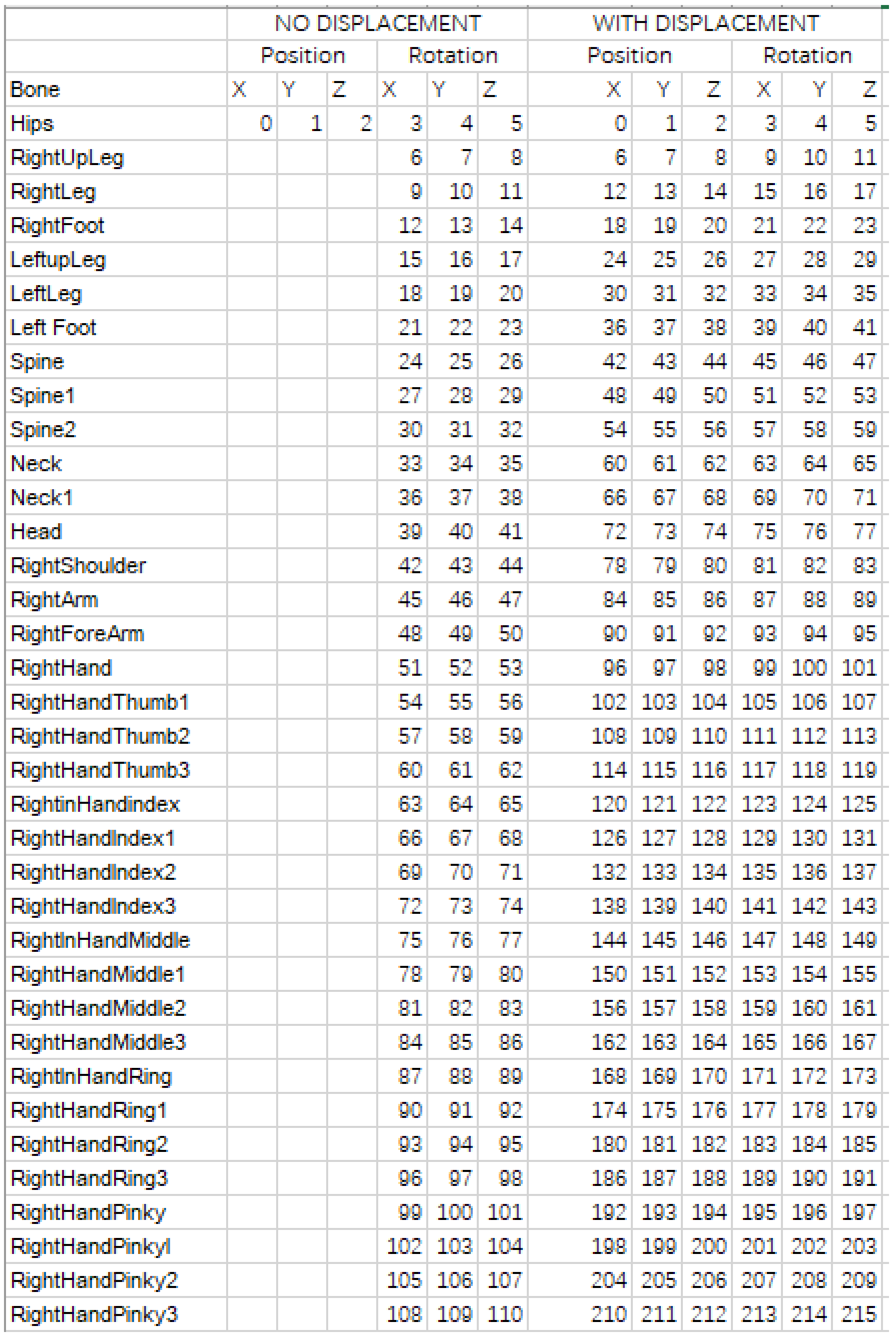
On the next two pages you'll find a complete graph of the whole sequence of the binary data received from Axis Neuron. It is a one-dimensional float array with different ordering and length depending on whether you're using displacement data or not.

Axis Neuron



	NO DISPLACEMENT						WITH DISPLACEMENT					
	Position			Rotation			Position			Rotation		
Bone	X	Y	Z	Y	X	Z	X	Y	Z	Y	X	Z
Hips	0	1	2	3	4	5	0	1	2	3	4	5
RightUpLeg				6	7	8	6	7	8	9	10	11
RightLeg				9	10	11	12	13	14	15	16	17
RightFoot				12	13	14	18	19	20	21	22	23
LeftUpLeg				15	16	17	24	25	26	27	28	29
LeftLeg				18	19	20	30	31	32	33	34	35
LeftFoot				21	22	23	36	37	38	39	40	41
Spine				24	25	26	42	43	44	45	46	47
Spine1				27	28	29	48	49	50	51	52	53
Spine2				30	31	32	54	55	56	57	58	59
Spine3				33	34	35	60	61	62	63	64	65
Neck				36	37	38	66	67	68	69	70	71
Head				39	40	41	72	73	74	75	76	77
RightShoulder				42	43	44	78	79	80	81	82	83
RightArm				45	46	47	84	85	86	87	88	89
RightForeArm				48	49	50	90	91	92	93	94	95
RightHand				51	52	53	96	97	98	99	100	101
RightHandThumb1				54	55	56	102	103	104	105	106	107
RightHandThumb2				57	58	59	108	109	110	111	112	113
RightHandThumb3				60	61	62	114	115	116	117	118	119
RightInHandIndex				63	64	65	120	121	122	123	124	125
RightHandIndex1				66	67	68	126	127	128	129	130	131
RightHandIndex2				69	70	71	132	133	134	135	136	137
RightHandIndex3				72	73	74	138	139	140	141	142	143
RightInHandMiddle				75	76	77	144	145	146	147	148	149
RightHandMiddle1				78	79	80	150	151	152	153	154	155
RightHandMiddle2				81	82	83	156	157	158	159	160	161
RightHandMiddle3				84	85	86	162	163	164	165	166	167
RightInHandRing				87	88	89	168	169	170	171	172	173
RightHandRing1				90	91	92	174	175	176	177	178	179
RightHandRing2				93	94	95	180	181	182	183	184	185
RightHandRing3				96	97	98	186	187	188	189	190	191
RightInHandPinky				99	100	101	192	193	194	195	196	197
RightHandPinky1				102	103	104	198	199	200	201	202	203
RightHandPinky2				105	106	107	204	205	206	207	208	209
RightHandPinky3				108	109	110	210	211	212	213	214	215

AxisStudio



	NO DISPLACEMENT						WITH DISPLACEMENT					
	Position			Rotation			Position			Rotation		
Bone	X	Y	Z	X	Y	Z	X	Y	Z	X	Y	Z
Hips	0	1	2	3	4	5	0	1	2	3	4	5
RightUpLeg				6	7	8	6	7	8	9	10	11
RightLeg				9	10	11	12	13	14	15	16	17
RightFoot				12	13	14	18	19	20	21	22	23
LeftupLeg				15	16	17	24	25	26	27	28	29
LeftLeg				18	19	20	30	31	32	33	34	35
Left Foot				21	22	23	36	37	38	39	40	41
Spine				24	25	26	42	43	44	45	46	47
Spine1				27	28	29	48	49	50	51	52	53
Spine2				30	31	32	54	55	56	57	58	59
Neck				33	34	35	60	61	62	63	64	65
Neck1				36	37	38	66	67	68	69	70	71
Head				39	40	41	72	73	74	75	76	77
RightShoulder				42	43	44	78	79	80	81	82	83
RightArm				45	46	47	84	85	86	87	88	89
RightForeArm				48	49	50	90	91	92	93	94	95
RightHand				51	52	53	96	97	98	99	100	101
RightHandThumb1				54	55	56	102	103	104	105	106	107
RightHandThumb2				57	58	59	108	109	110	111	112	113
RightHandThumb3				60	61	62	114	115	116	117	118	119
RightinHandindex				63	64	65	120	121	122	123	124	125
RightHandIndex1				66	67	68	126	127	128	129	130	131
RightHandIndex2				69	70	71	132	133	134	135	136	137
RightHandIndex3				72	73	74	138	139	140	141	142	143
RightInHandMiddle				75	76	77	144	145	146	147	148	149
RightHandMiddle1				78	79	80	150	151	152	153	154	155
RightHandMiddle2				81	82	83	156	157	158	159	160	161
RightHandMiddle3				84	85	86	162	163	164	165	166	167
RightInHandRing				87	88	89	168	169	170	171	172	173
RightHandRing1				90	91	92	174	175	176	177	178	179
RightHandRing2				93	94	95	180	181	182	183	184	185
RightHandRing3				96	97	98	186	187	188	189	190	191
RightHandPinky				99	100	101	192	193	194	195	196	197
RightHandPinkyl				102	103	104	198	199	200	201	202	203
RightHandPinky2				105	106	107	204	205	206	207	208	209
RightHandPinky3				108	109	110	210	211	212	213	214	215