**Tutorial: Applying Bone Rotations in NIUI**

This is a short tutorial on how to setup your current skeletal mesh to recieve bone rotations through the NIUI API.

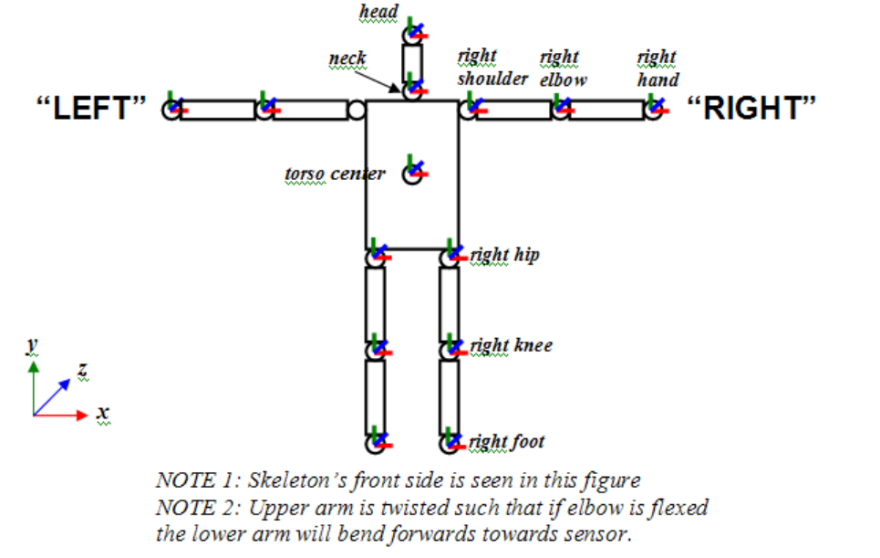
This tutorial assumes the following knowledge:

* Able to use the SkeletalMesh viewer.
* Knowledge of how to build Anim-Trees
* Basic Knowledge of UnrealScript

Throughout this tutorial I will be using the skeletal mesh (UDK\_ProtoRig\_03) that comes with the NIUI API in the demo content.

Before starting this tutorial, I strongly recommend that you look at the sample project to learn how the API is structured. I also recommend that you play around with the sample project files

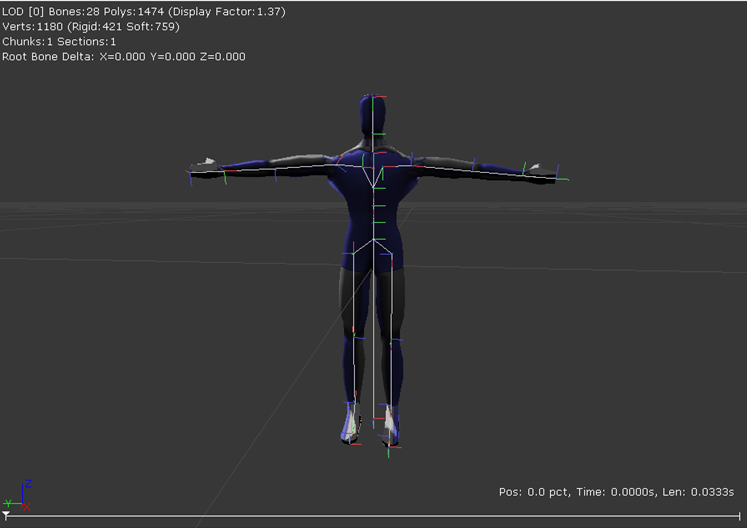
**A bit of house-keeping:**

Firstly, I need to just state the requirements for your skeletal mesh when working with NIUI. In order to apply rotations to bones correctly, NIUI needs to cache the bone rotations from the skeletal mesh when it is in an OpenNI compliant T-Pose (See image to right).

The best way to do this (and the way I recommend) is to create a single frame animation for your mesh so that it is in this pose. You should then force this animation to play while OpenNI is calibrating so that when the user is calibrated NIUI can cache these and use as a basis for applying rotations.

The advantage to using a T-Pose animation is that you can play/blend animations underneath the bone overriding and the modellers are not restricted to modelling the mesh in the OpenNI compliant pose (which can create deformations when animating if not skinned properly).

Also, a forward warning that I have found overly complex skeleton setups don’t tend to work well when you want to blend bone overriding and animations. For best results, I recommend that you keep your skeletal setups reasonably simple.

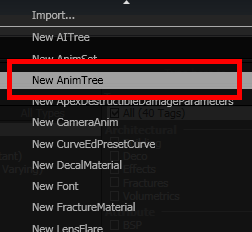


As you can see in the above image, ProtoRig has a single frame T-Pose animation that is played when a user is calibrating.

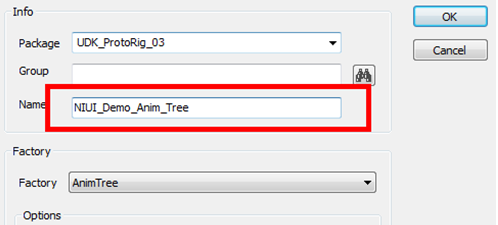
In future, I am hoping to extend the API so that you can store these rotations offline and remove the need to force this animation while in game. The need for the T-Pose animation will stay between versions however (for the reasons stated above).

**Building the Anim-Tree:**

Open up UDK and create a new Anim-Tree in your desired package by right clicking in the content browser and selecting **New Anim-Tree** in the pop-up menu. This will build us an empty animation tree that we are going to use to set up NIUI to override a meshes bones rotations.

****

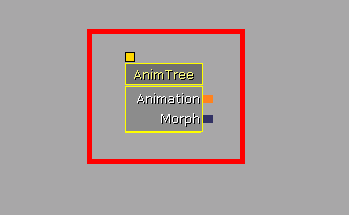
A small window should appear like the one below. For this example let’s just call the new Anim-Tree **NIUI\_Demo\_Anim\_Tree**.



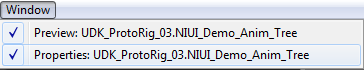
Be sure to leave the anim-tree in the same package as your skeletal mesh as it will make things a lot easier later on.

You should now have an empty anim-tree work space.

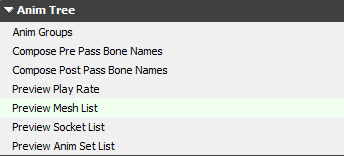
The first thing that we are going to do is link up the skeletal mesh with the anim-tree as this will allow us to add bone controllers to the root animation node. We do this through clicking on the root node.



When you have selected this you should see the properties window change to reflect all the editable settings in the node. If you don’t have access to the properties, click on the **Window** menu at the top of the window and then select the **Properties** selection.

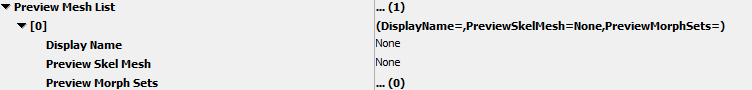


In the properties windows, find the **Anim-Tree** tab and then the **Preview Mesh List** property. This holds the list of meshes that are available for previewing within the animation tree editor.



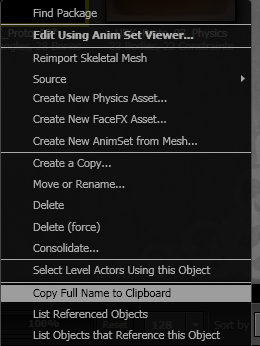
On the far right side of the window, click on the small green **** icon to add a new mesh to the preview selection.

Expand the **Preview Mesh List** property to see an empty mesh setting like the one below.

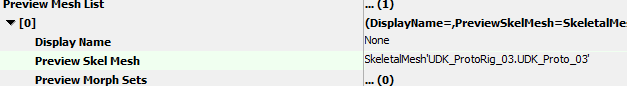


From here we are going to change the **Preview Skel Mesh** to reference to the target skeletal mesh for the bone control overriding. So go back to the generic browser and find your skeletal mesh and right click on it.

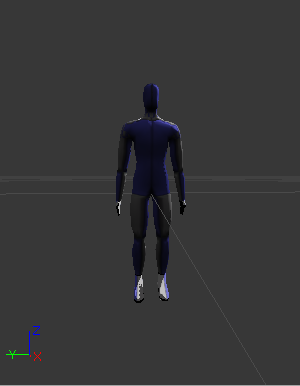
You should see a small pop-up menu like the one below.



Select **Copy Full Name to Clipboard** to copy the full reference path to the mesh. Now go back to the properties within the Anim-Tree editor and paste the reference path into the **Preview Skel Mesh** property.

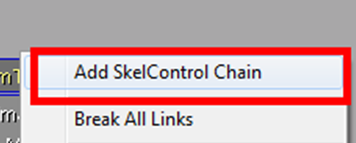


Two things should happen here: The mesh name should now reside in the **Preview Skel Mesh** property (see above) and the mesh should appear in the small preview window next to the Anim-Tree Editors workspace (see image below).

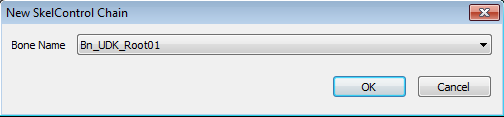


Now we will start setting up the bone controls for our mesh by creating NIUI\_SkelControlSingleBone nodes. For this tutorial I am going to keep it simple and just make an override for the Spine bone.

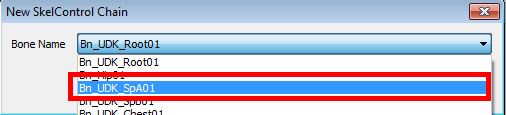
Lets start of by setting up a control chain for the spine bone. Right click on the root node and select **Add SkelControl Chain**. This adds an output to the root node that will allow us to connect BoneControllers that will affect the bone linked to it.



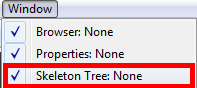
You should now see a window that allows us to select which bone to create an output for the skeletal control chain (see window below). For this we are going to



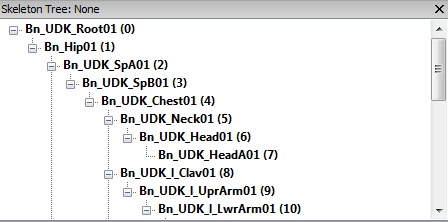
For this we are going to choose the spine bone that will best suit the NIUI\_JOINT\_TORSO input. For ProtoRig, this is the lower spine bone, **Bn\_UDK\_SpA01**. Click ok to add a output channel for the bone.



On a side note, to help decide which bones will be best for inheriting the NIUI rotations for your mesh, you should perform a visual test in the skeletal mesh viewer. Do this by opening up your mesh in the viewer and opening up the **Skeleton Tree** window under the **Window** menu option.

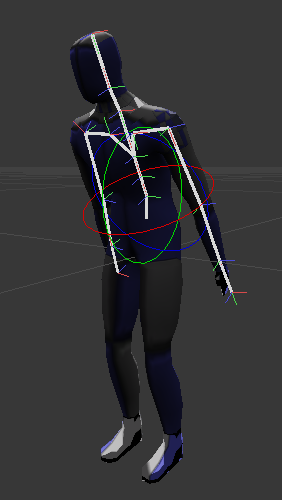
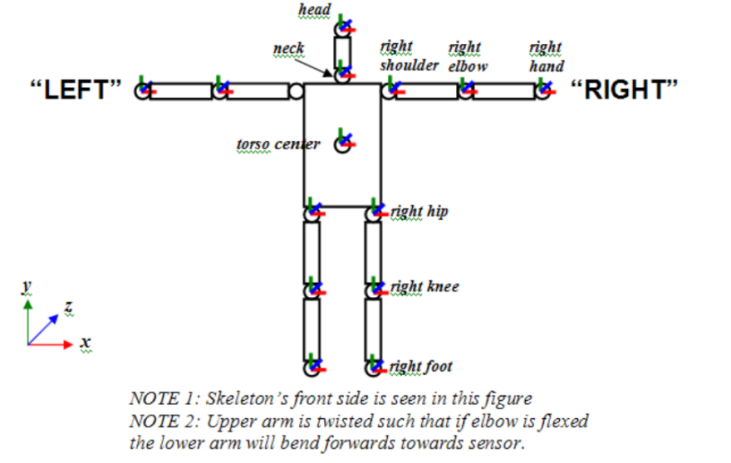


This should open up a window that looks like the one below.

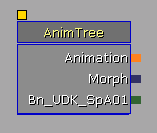


Within this window we have access to all the bones within the skeleton hierarchy.

The best way to decide which bone is most appropriate for inheriting the rotations from NIUI, compare the bones locations to the OpenNI T-Pose and also rotate the bone to make sure it deforms properly.



Once you have chosen which bone will be most suitable to inherit the torso rotation, jump back into your **Anim-Tree Editor** workspace. If you havent already done so, add an output for the target bone. You should now have an AnimTree similar to the one below.



We are now going to create the bone control node and set it up to override the bone with the OpenNI rotations.

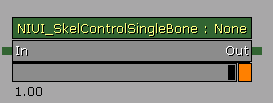
Right click in the **Anim-Tree Editor** workspace and a pop-up menu should appear. From within the menu, select **New Skeletal Control** then **Single Bone** and finally **NIUI\_SkelControlSingleBone**.



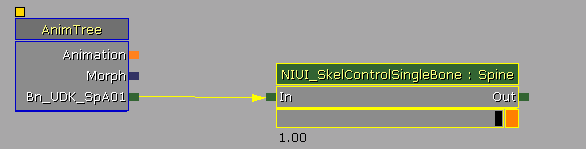




You should now have an empty control node like the one below.

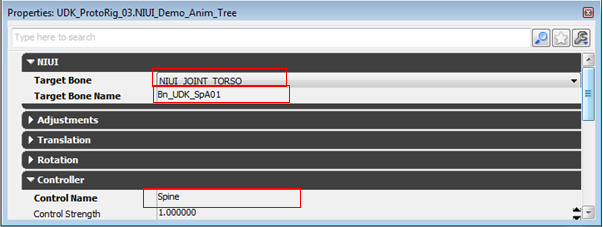


Link this up to your bone channel by clicking on the small green square to the left of the **In** wording in the and dragging to the small green square next to your bone name in the root node.



You should now have a connection like the one above.

Now we will configure the **NIUI\_SkelControlSingleBone** to pass the rotations from the NIUI bone to the connected bone. Under the NIUI tab, change the **Target Bone** field to **NIUI\_JOINT\_TORSO**. This configures the node to retrieve the rotations from the torso joint and then relay onto the connected bone.



We also need to put the name of the connected bone in the **Target Bone Name** field as this allows the control to cache the reference rotation just before starting to apply the rotation to the bone.

Lastly, make sure you name the controller something recognisable and clear.

NOTE: The name of the node’s are needed by the **NIUI\_SkeletalController** in order to cache the bone control nodes. Look in the sample project for the NIUI\_SamplePawn class to see the TargetBoneNames variable and the **PostInitAnimTree** function and how they are used to configure the skeletal controller.

The above steps are the same for all bones, with the exception of differing bone control channels and target NIUI bones within the nodes.

And that is the end of this tutorial. I will build another more thorough tutorial for this in the future, but for now this should be enough to get you started. As I said at the start of the document, I strongly recommend you play around with the sample project before doing this yourself so you can get an idea of how it all fits together.