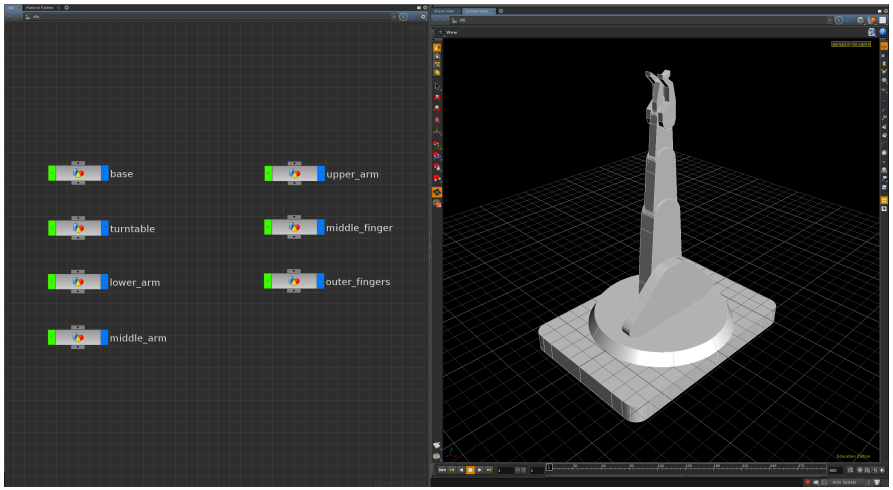
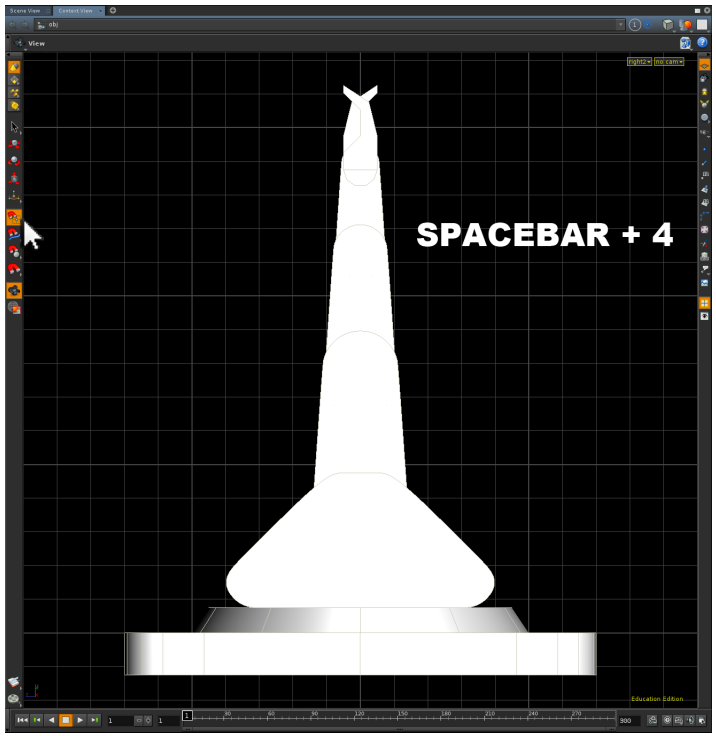
CREATING BONES

Bones can move objects, either through direct parenting of objects to bones (utilised for hard surface characters such as robots) or by geometry association (also known as Capturing or Skinning). The latter is utilised for any organic character whose movement depends upon the deformation of the character's geometry. The creation of bones is however universal for both techniques. Open the scene **robot_arm_begin.hipnc**



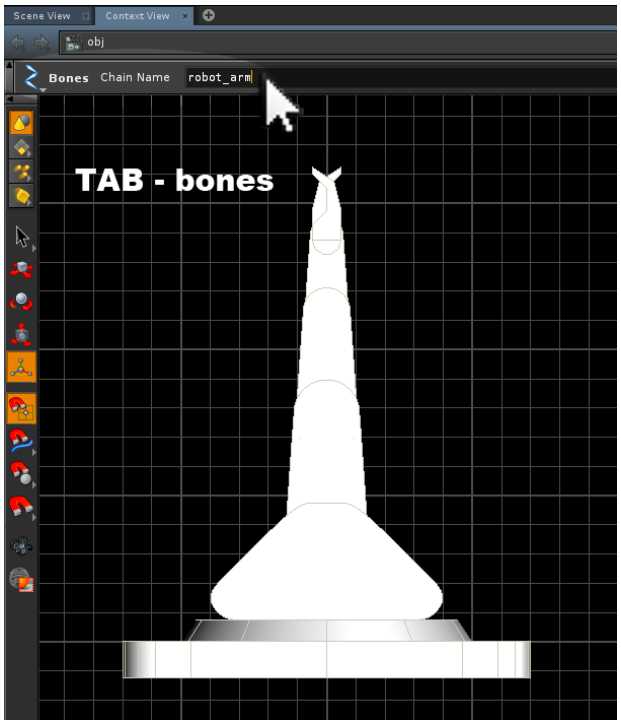
This scene contains a simple robot arm, which can be utilised for demonstrating the creation of bones.

Switch the **Viewer** to the **Right Orthographic View** (**SPACEBAR + 4**), and ensure that **Grid Snapping** is activated, and that the **Construction Plane** is displayed.



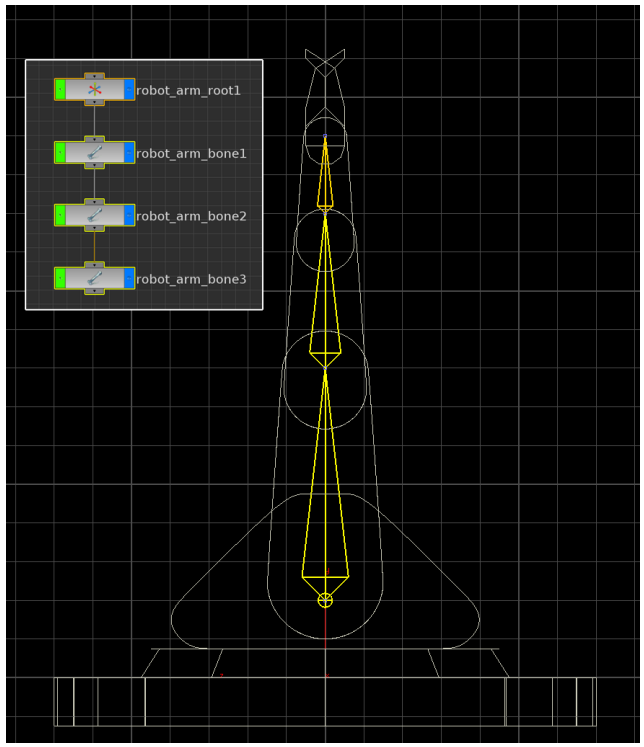
Bones should always be drawn using an Orthographic View, so their correct initial positioning is ensured.

With the mouse over the **Viewer** press **TAB** and type **bones**. This will activate the **Bones Tool**. The Bones Tool works in the same way as the Curve Tools in Houdini.



Before beginning to draw the bones, set the **Chain Name parameter** located at the top of the View Pane as **robot_arm**. This will prefix any bones created with the name robot_arm.

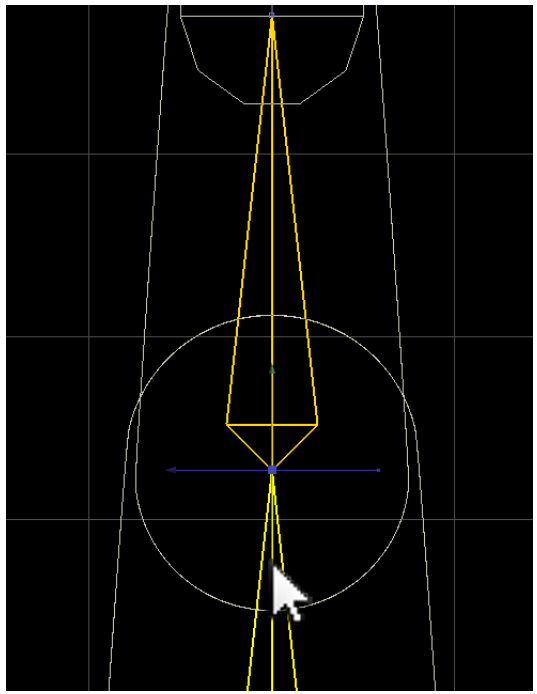
Set the **Viewer display** to **Wireframe**, and **LMB** draw the first three bones of the robot arm along the centre Y-axis, and press **ENTER** to confirm the operation.



This will automatically create a series of parented **Bone Objects** in the **Network View**. Note that a **Null Object** called **root** is utilized to specify the start of the bone chain.

REFINING BONE POSITIONS

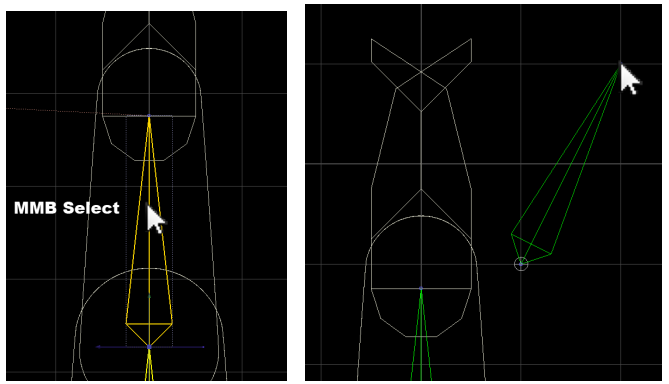
With the initial bone chain complete, **deactivate Grid Snapping**. The position of each bone can be adjusted by selecting the **small blue square** at its beginning or end.



Adjust the **Y position** of each bone, so that each joint is centered relative to the geometry. The position of each bone will equate to a pivot point when the geometry is moved by it.

CREATING PARENTED BONES

Still in Bones Tool mode, reactivate **Grid Snapping**, and set the **Chain Name** parameter of the **Bones Tool** to **robot_finger**. **CTRL + MMB select** the **top arm bone**. This will draw a **dashed blue box** around it indicating that it will become the **parent** for any new bones drawn.

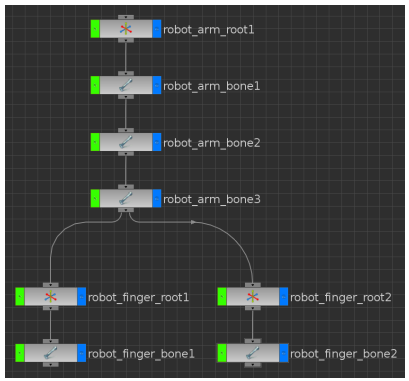


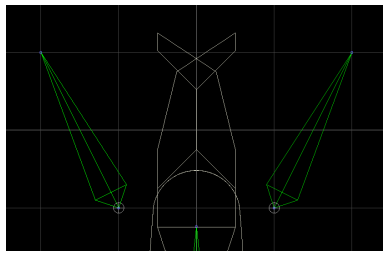
Slightly away from the robot arm, create a **single bone** for the **right robot finger** and press **ENTER** to confirm the operation.

NOTE: If necessary, **turn off** the **Display** of the **Robot Arm** geometry to make the creation of parented bones simpler.

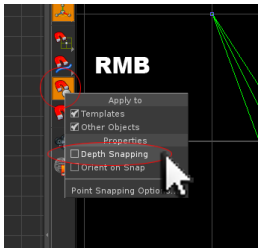
Repeat the **parenting bone operation** to draw a **second finger bone** for the **left robot finger**. Again press **ENTER** to confirm the operation.

Examine the **Network View** to see how the new Bone Objects have been configured in the bone chain.



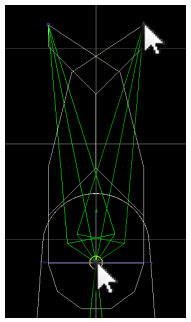


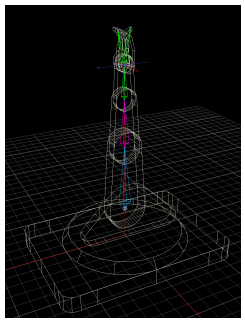
In the **Viewer**, deactivate Grid Snapping and instead activate **Point Snapping**. **RMB** on the **Point Snapping Button** to **deactivate Depth Snapping**.



This will ensure any repositioning of bones will happen on the plane they were drawn.

With Point Snapping activated, reposition the **robot_finger Bones** relative to the **Geometry finger Objects**.

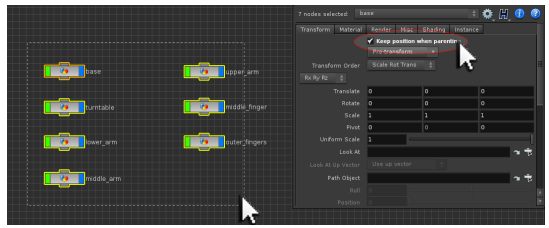




With this complete **deactivate Point Snapping**. The bones are now ready for the geometry to be parented to them.

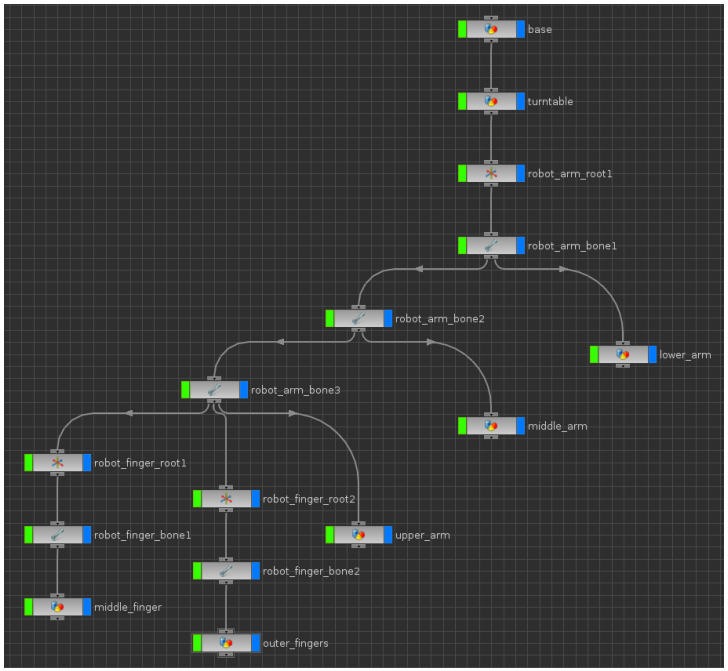
PARENTING THE GEOMETRY

In the **Network View** select all of the robot arm Geometry Objects, and in the parameters for this selection, activate the **Keep Position When Parenting** option.



Parent each robot arm **Geometry Object** (the child) to their corresponding **Bone Object** (the parent). This can be done manually in the **Network View** or interactively in the **View Pane** using the **Parent Tool**.

The **base Object** and the **turntable Object** should be parents to the **robot_arm_root1 Object**. The final network should look like this:



See **robot_arm_stage1.hipnc**

ADDING KINEMATICS TO DRIVE BONE MOVEMENT

At present the interactive **Pose Tool** can be utilised in the **Viewer** to position bones on a per bone basis. While this is useful for checking that the parenting has worked as expected, it is not practical for animation purposes.

Instead Kinematics are utilised within animation software for these tasks. There are two types of **Kinematics** that can be assigned to bones; **Forward Kinematics** (**FK**) and **Inverse Kinematics** (**IK**).

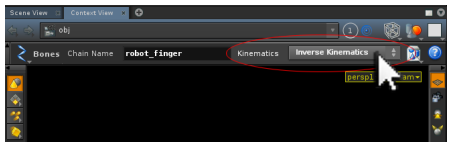
Forward Kinematics retains the positioning of bones on a per bone basis; however it simply adds a useful control handle to the bone to do this (in the form of a Null Object).

Inverse Kinematics creates a Null Object that automatically drives the position of an entire bone chain, and can be useful for animating legs and arms.

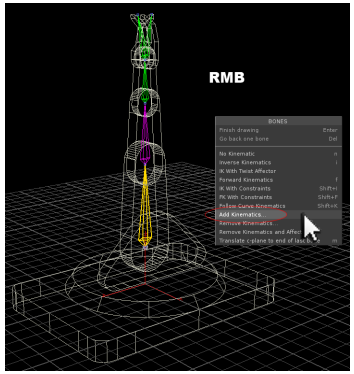
Either type of Kinematics can be assigned to Bone Objects before or after their creation. Adding Inverse Kinematics to a single Bone Object is the same as adding Forward Kinematics to a single Bone Object.

INVERSE KINEMATICS (IK)

In the **Network Editor** select the **Bone Object** controlling the lower section of the robot arm, and with the mouse over the **Viewer** press **TAB** and type **bones**. This will reactivate the **Bones Tool**.



Using the Bones tool parameters located at the top of the Viewer; set the **Kinematics** drop down menu to **Inverse Kinematics**.



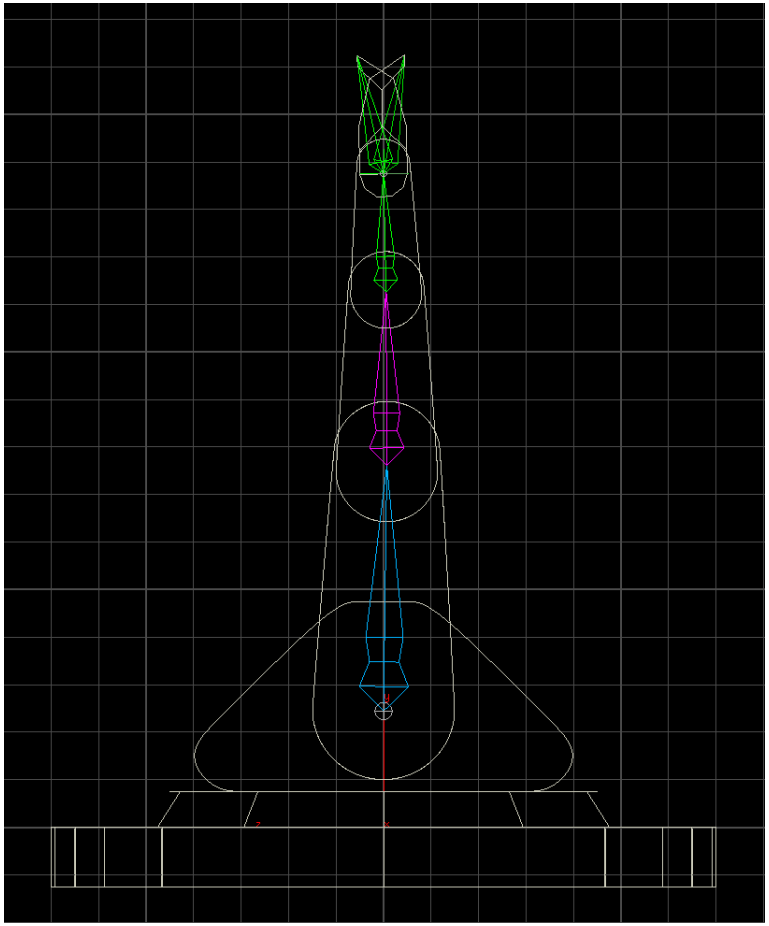
RMB on the **Viewer** to show the **Bones Tool Menu**, and select **Add Kinematics**.

As the first bone of the Inverse Kinematics chain was selected beforehand, the **Help Prompt** at the base of the Viewer will ask for the last bone of the Inverse Kinematics chain to be selected.

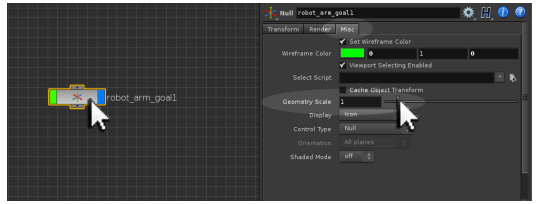
In the **Viewer** select the upper bone of the robot arm, and press **ENTER** to confirm the IK operation.

A Null Object called robot_arm_goal1 will be automatically created in the Network Editor, alongside a CHOP Network containing the IK Solver for the bone chain.

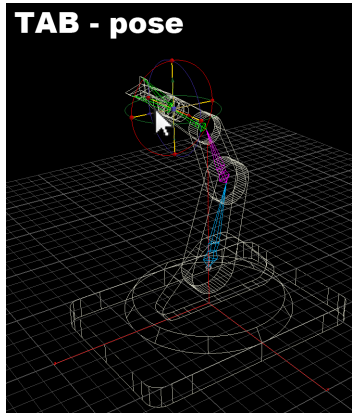
When the robot arm is examined in the **Right Orthographic View**, the robot arm curves slightly to the left. This is because an IK bone chain can only bend in one direction. The appearance of the bones has also changed indicating the inclusion of Kinematics.



NOTE: If a bend direction is specified when bones are drawn, Houdini will follow this when creating IK bone steup. If no bend direction is specified, Houdini will choose the bend direction.



In the **Network Editor**, select the **robot_arm_goal1 object**, and under the **Misc** section of its **parameters**, increase the **Geometry Scale** parameter to make the goal object easier to see in the Viewer.

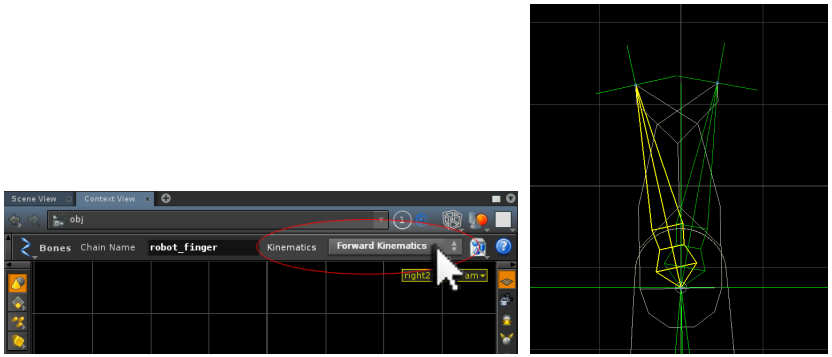


With the mouse over the **Viewer**, press **TAB** and type **pose**. This will activate the **Pose Tool** for positioning the goal object. The bones with Inverse Kinematics assigned to them follow the goal in the bend direction specified by the IK bone chain setup.

FORWARD KINEMATICS (FK)

Undo any **Pose Tool** changes to the position of the robot arm, and using the same Kinematics method as before, add **Forward Kinematics** to the two finger bones.

Select each bone in turn in the **Network Editor**, and reactivate the **Bones Tool** in the **Viewer**; this time specifying Forward Kinematics. **RMB** on the **Viewer** to activate the **Bones Tool Menu** and choose **Add Kinematics**. Press **ENTER** to confirm the same finger bone as the last bone in the FK chain.

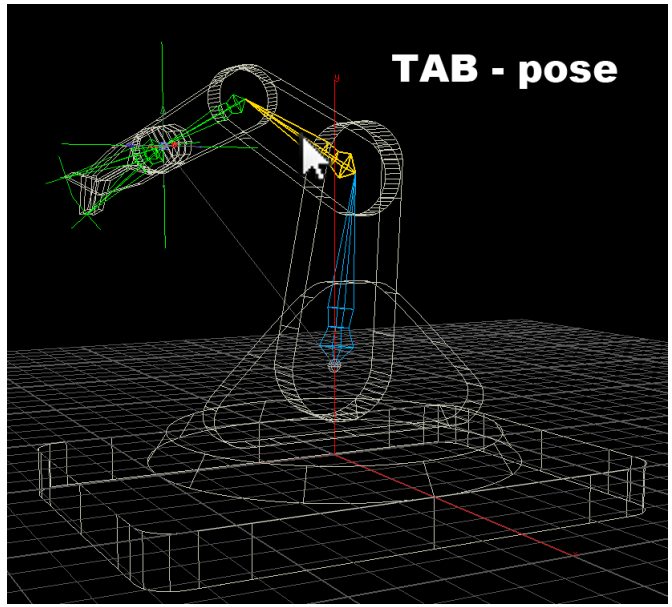


NOTE: When FK is assigned to bones, the goal Null Objects created are parented into the existing hierarchy, rather than appearing as separate objects in the Network Editor.

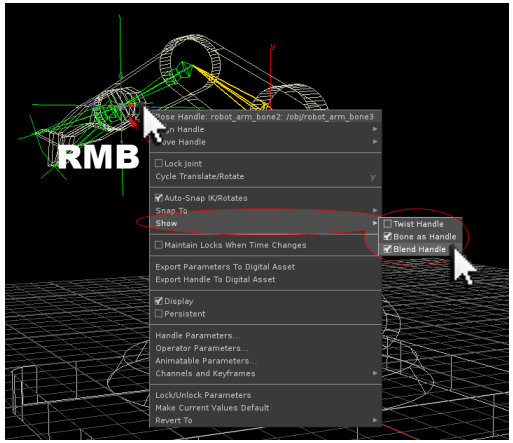
See robot_arm_stage2.hipnc

IK / FK BLENDING

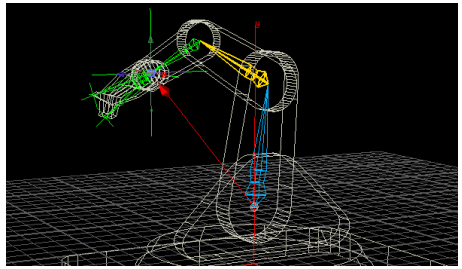
While the direction of the robot arm bend has been specified by the IK setup, it is possible to override this behavior allowing for greater animation flexibility without loss of IK functionality. Any IK bone chain can as a FK bone chain. This is known as IK/ FK Blending. Using the **Pose Tool** in the **Viewer**, position the **robot_arm_goal1 Null Object** to create a bend in the robot arm.



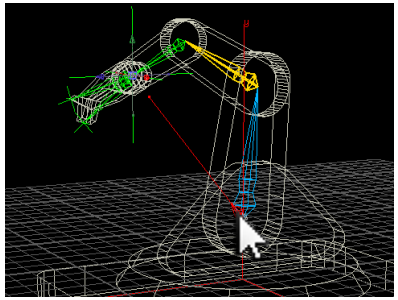
Whilst still in **Pose Tool Mode**, **select** one of the **arm bones**. A thin grey line will appear from the start of the IK setup to its end.

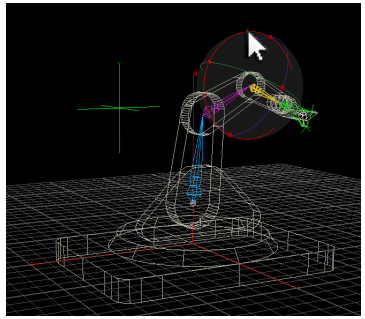


RMB on the Pose Tool Handle to reveal the contextual menu for it, and from the **Show** section, activate the **Blend Handle** option. The thin grey line will turn red with a red diamond slider attached to it.

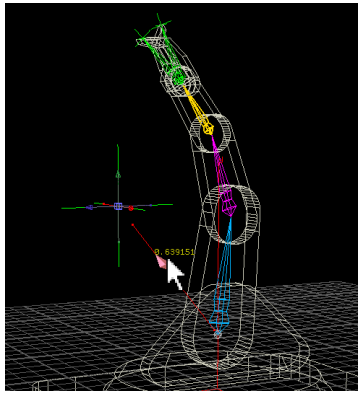


Move the red diamond slider down to the opposite end of the red line to turn off the influence of IK.

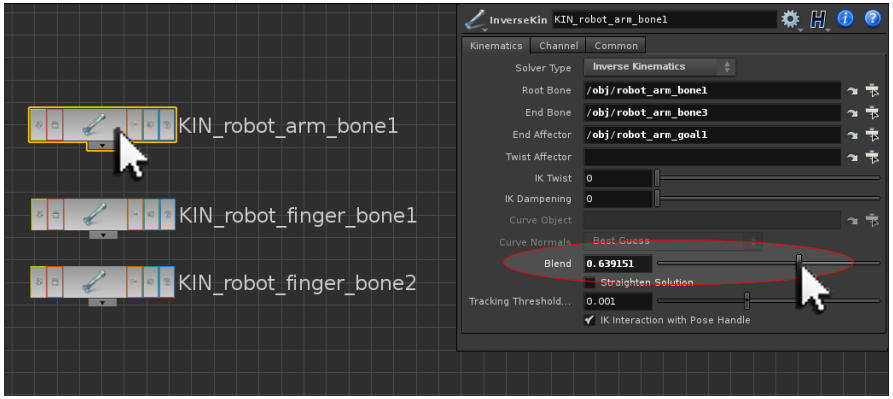




Press r with the mouse over the Viewer to activate the Rotate Tool. The bones can now be positioned so the bend of the robot arm goes in the other direction to the IK setup.



When the Show Blend Handle option is reactivated using the Pose Tool, moving the red diamond slider will blend between the two arm positions, going from IK to FX or back again.

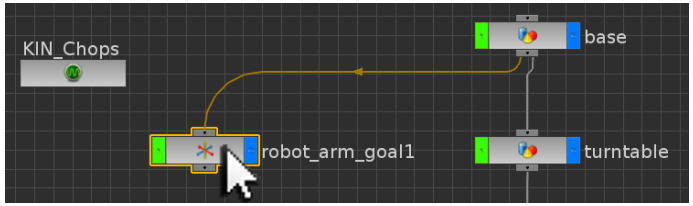


This blend value can also be key-framed. The blend parameter is located inside the CHOP Network automatically generated by the IK FK bone setup on the Inverse Kin CHOP associated with the robot arm.

See file **robot_arm_stage3.hipnc**

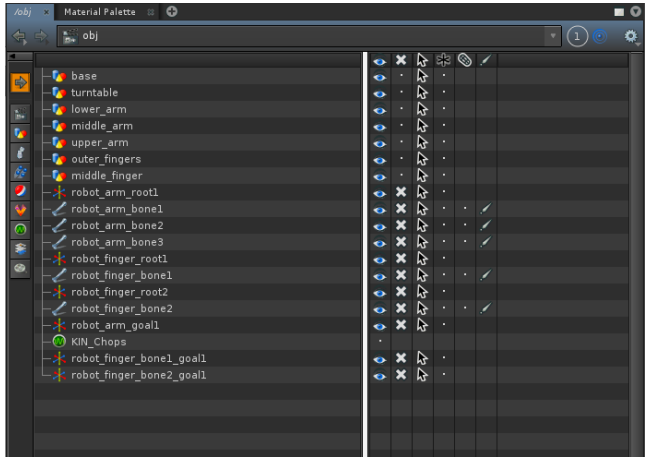
CHARACTER SETUP STRATEGIES

The base object of the robot arm can also serve as a Global Transform for the character setup. Parenting the robot_arm_goal1 Null Object to the base object will ensure that wherever the base is placed, the robot_arm_goal1 Null Object will also follow. The robot_arm_goal1 Null Object is however still independent allowing for animation to take place. Doing this would for example be a useful strategy if a factory line of robot arms were being created.



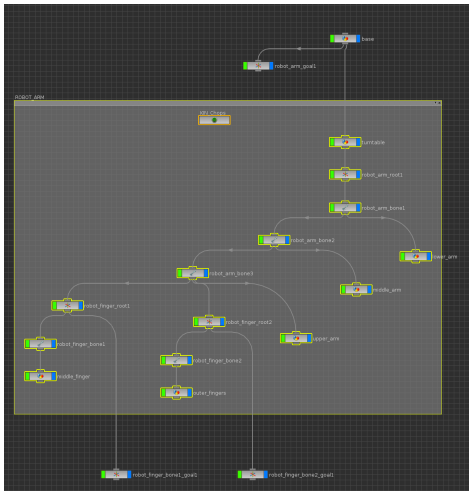
NETWORK EDITOR LIST VIEW

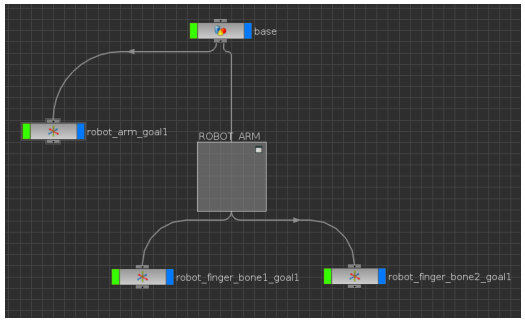
Pressing t with the mouse over the Network Editor will reveal a list view. This list view can make selection of certain object types simpler, as well as allowing for the (blue) display and (green) select flags of node selections to be disabled on enabled. For example, it may be prudent to disable selection ability for key parts of the robot arm setup that should not be interacted with at animation time; or for turning off the display of bones.



NETBOXES

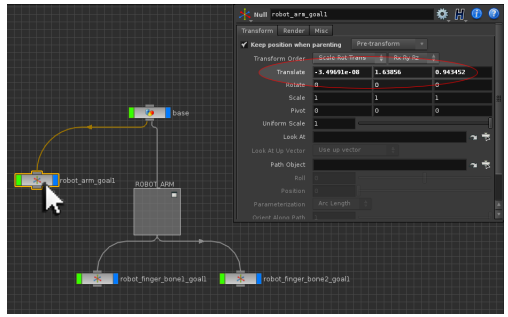
The nodes creating the robot arm setup can also be collapsed into a Netbox, by selecting all of the nodes apart from the controllers, and pressing CTRL + n with the mouse over the Network Editor. The Netbox can then be minimized to hide any non-essential nodes.



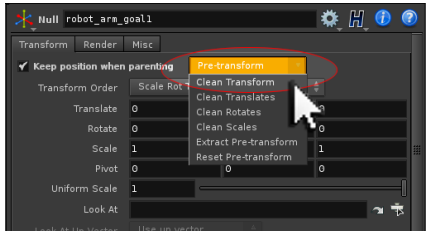


CLEANING TRANSFORM INFORMATION

Each of the robot arm controllers has transformation information assigned to it. For animation purposes, this can be reset to a clean state for animation by ‘cleaning’ the transform information. This resets all transform parameters to 0 without moving the object.



From the **Pre-transform drop** down menu of the parameters, select **Clean Transform** for each of the control objects in the scene.



See file robot_arm_end.hipnc

OTHER CHARACTER TOOL UTILITIES

CHARACTER DIGITAL ASSETS

When the Network configuration is complete, the entire network for the robot arm can be collapsed into a Sub Network (SHIFT + c) from which a Digital Asset could be generated. Key animation controls can then be ported up to the top level of the Digital Asset for animation purposes.

ALIGNING BONES

Bones can be positioned and aligned with their parent whilst in **Bones Tool** mode. Simply select the end of the bone to be aligned and **RMB** on the transform handle. From the resulting contextual menu choose **Align Handle > Align to Parent Bone**. The rotation tool can now be utilised to align the bones relative to each other. When aligning bones, start at the root and work outwards towards the end of the bone chain. Cleaning Transform information is also important after aligning bones. Aligning Bones should form part of the bone setup before the assignment of geometry to the skeleton.

MIRRORING BONES

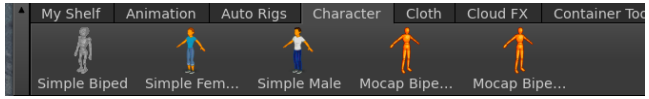
Bone setups including IK can be mirrored in the Viewer by utilising the interactive Mirror Tool. This means a single leg with IK can be created and mirrored to create its opposite leg. This can save time when generating characters.

AUTO RIGS

Houdini ships with a number of preconfigured bone setups that can be found on the **Auto Rigs Shelf**. These can be activated, combined and or modified if necessary. There is also a fully rigged and skinned **Toon Character** which can be useful for animatics, and proof of concepts where a character is asked to interact with other scene elements.

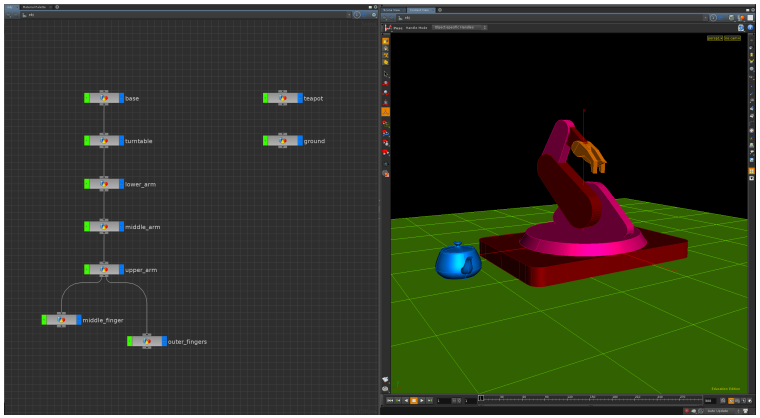
PRE MADE CHARACTERS

Other pre-made character rigs can also be found under the **Character Shelf**.

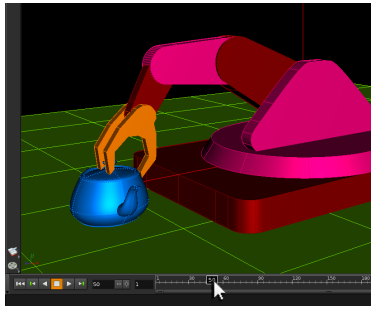


DYNAMIC PARENTING

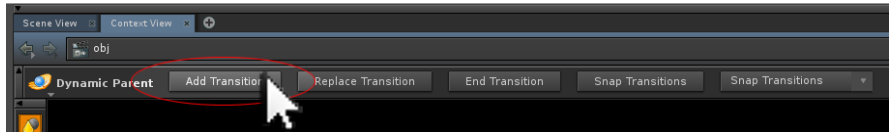
Dynamic Parenting allows for automated parenting/un-parenting of one object to another. Open the scene **dynamic_parenting_begin.hip**. This scene contains an animated **robot arm** (rigged using simple parenting rather than bones) and a **teapot object** for it to interact with. The animation on the robot arm emulates the arm picking up the teapot and placing it back down. It at present does not pick up the teapot object. Dynamic parenting will allow the teapot object to be parented to the arm only for the duration of the lift.



With the mouse over the **Viewer**, press **TAB** and type **dynamicparent** to activate the Dynamic Parent tool. Select the **child** object (the **teapot object**), and move the timeline onto frame **50** (where the robot arm first makes contact with the teapot).

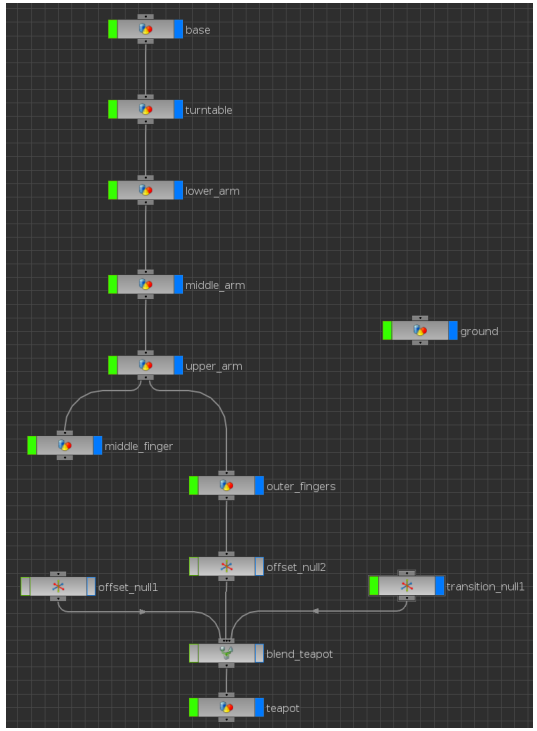


At frame **50**, press the **Add Transition** button located at the top of the **Viewer**.



Upon the request for the selection of a **parent object**, select the robot's **outer_fingers**. Now when the timeline is moved forward, the teapot moves with the robot arm as if it has been picked up.

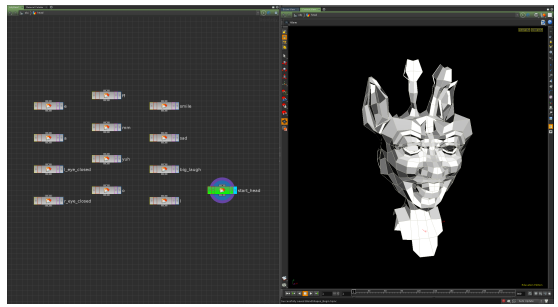
Move the timeline onto frame **125** (where the robot arm appears to put the teapot OBJ down), and press the **End Transition** button. The teapot will now be released from the grip of the robot arm.



Extra nodes have also been automatically created in the Network Editor as a result of this Dynamic Parent operation. Essentially, a Blend Object is being utilized to switch between 3 parent Null Objects for the teapot. The first parent Null Object locates the teapot on the ground, in the position before it has been picked up. The second parent Null Object is linked directly to the outer_fingers object. The third parent Null Object locates the teapot at its destination point. **See file dynamic_parenting_end.hipnc**

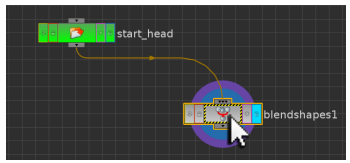
BLEND SHAPES

Blend Shapes can be used to **morph** between pieces of existing geometry that have the same topology. One if their uses are for creating facial expressions. Open the scene **BlendShapes_Begin.hip**

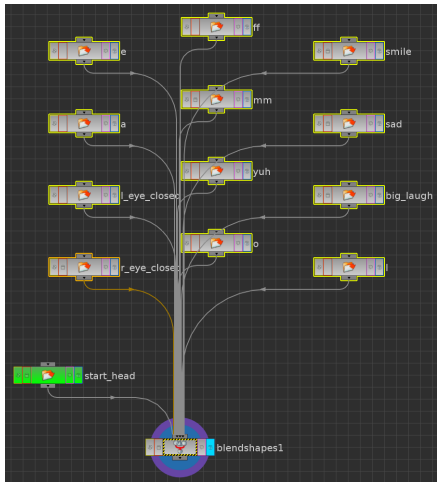


This scene contains a series of File SOPs all referencing in slightly different versions of a same character face. Activating the Display Flag for each File SOP in turn will display the facial different expressions.

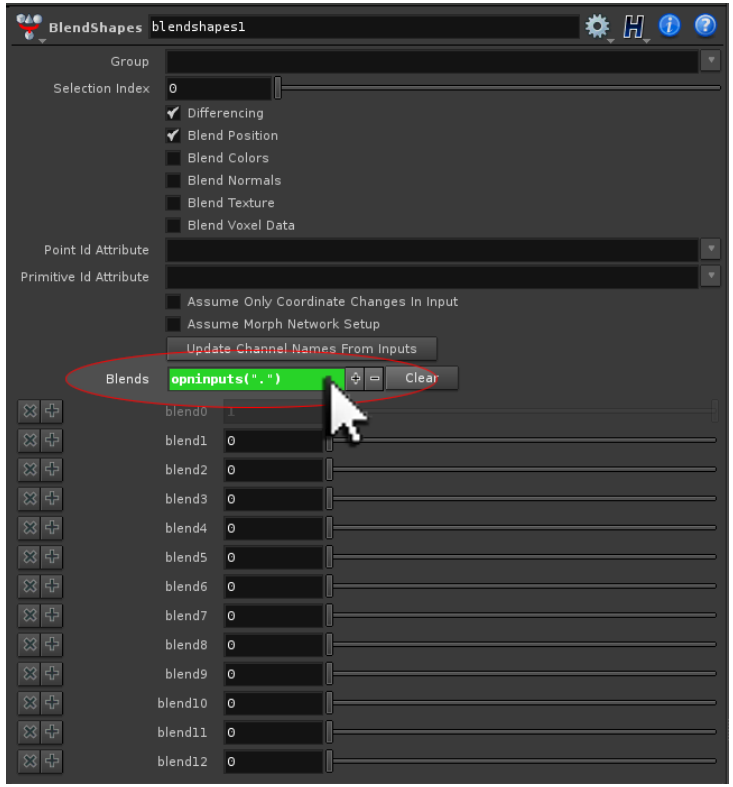
These individual geometries can be consolidated by utilising a **BlendShapes SOP**. This behaves in a similar way to a Merge SOP with regard to multiple inputs but will also allow for the creation of controls to blend between each face shape. **See also the Sequence Blend SOP.**



Wire the output of the **start_head File SOP** into a **Blendshapes SOP**. This will ensure that the neutral default head shape is the one used as the basis for the blending operation.

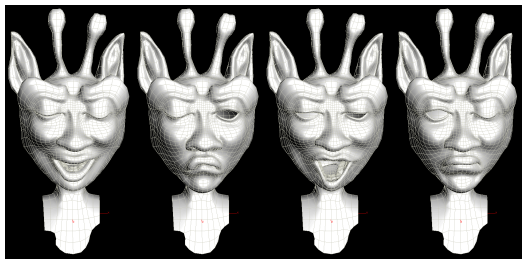


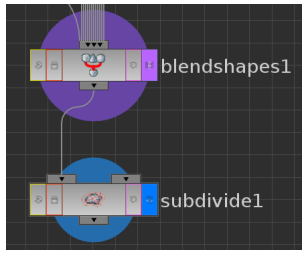
Select **all of the remaining nodes**, and **multi-wire** them into the **Blendshapes SOP**.



In the **parameters** for the **Blendshapes SOP**, add **opninputs(“.”)** to the **Blends parameter** to automatically calculate the number of blends to be created.

The blend values can now be increased or decreased, altering the shape of the face, to form character. These controls can be key framed and also be added into a Digital Asset.

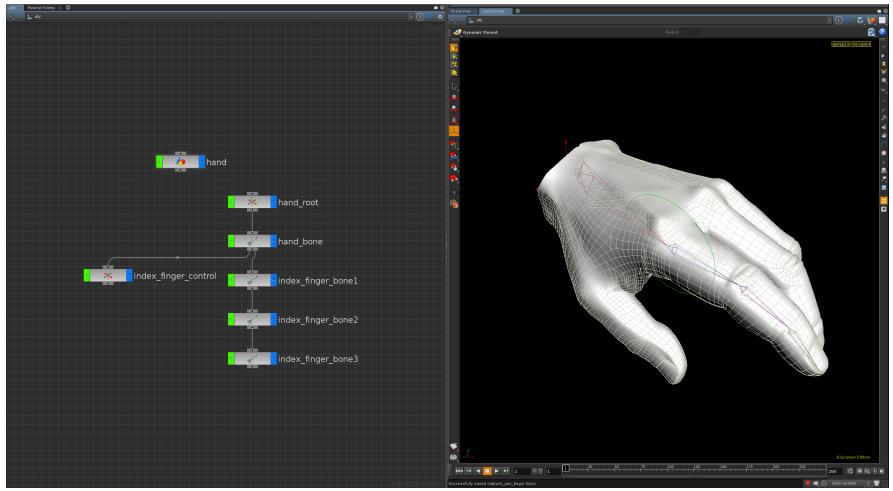




A **Subdivide SOP** can also be utilized for the purposes of displaying a smoothed version of the character. Setting the **Render** Flag to the **BlendShapes SOP** will render out the low-resolution geometry unless render **Polygons as Subdivision Surfaces** is specified at **OBJ Level**. If this is set, **Mantra** will handle the subdivision at render time instead. **See file BlendShapes_End.hipnc**

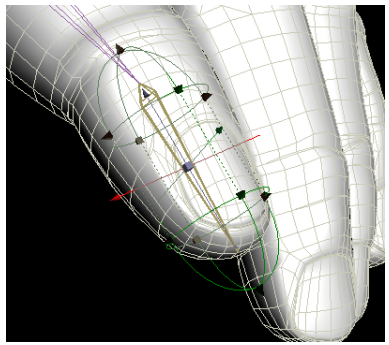
CAPTURING GEOMETRY

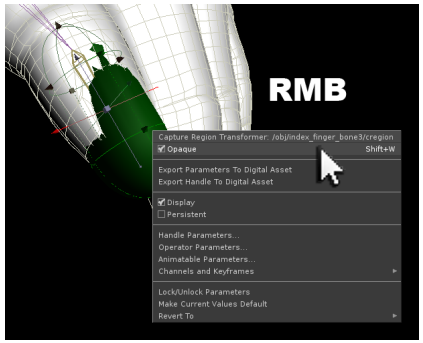
In order to create geometry that deforms alongside the bone movement, the geometry itself has to be captured and its position associated with the bones. This is done at Object Level. **Open the scene caputure_geo_begin.hipnc.**



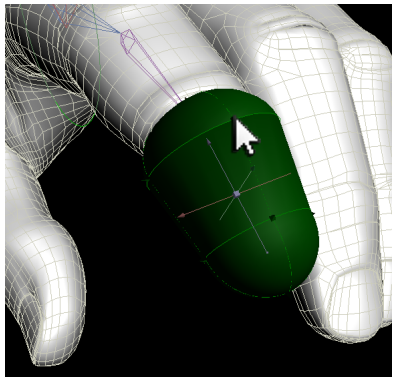
This scene contains geometry of a hand, and a currently unassociated underlying bone structure. A custom finger rotation control has been added to the bone setup to drive its movement. In order to capture the geometry effectively, the Capture Regions of each bone should first be edited. This will give an area-based region for the initial capture to be set to.

With the mouse over the **View Pane**, press **TAB** and type **editcaptureregions** in order to initiate the **Edit Capture Regions** tool. When prompted, select the associated with the fingertip.





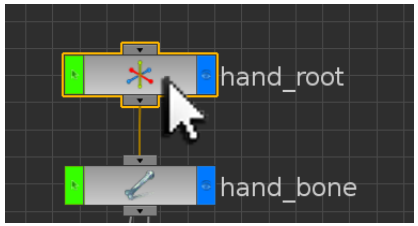
RMB on its Capture Region Handle will allow for Opaque visualisation of the region making it easier to work with.



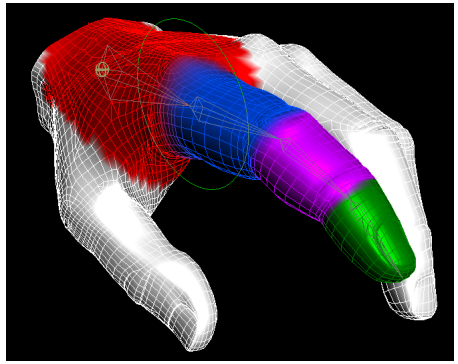
Modify the interactive region handles so the capture region encompasses all of the geometry associated with the fingertip. Repeat this operation for all of the other finger bones; ensuring that each capture region covers the area of the hand geometry it should. Very often, good capture regions are sufficient enough for controlling geometry deformations without the need for painting capture weights; so it is important to get the capture regions as correct as possible to minimize the work necessary later on.

CAPTURING GEOMETRY

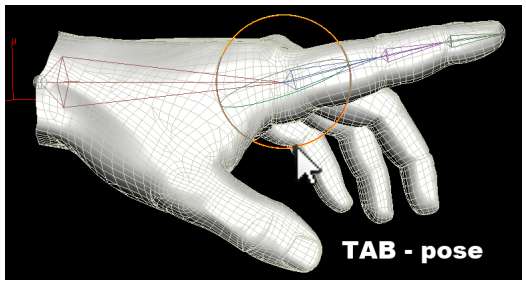
With the mouse over the Viewer press **TAB** and type **Capture Geometry**. Select the hand geometry and press **ENTER** to confirm the selection.



The **Help Prompt** will now ask for the root object of the bone network to be selected. Selecting the root object node in the Network Editor is the simplest way to do this. With the root object selected, press ENTER once again with the mouse over the Viewer. This will complete the capture process, passing the colour of the bone onto the associated geometry.



When the i**ndex_finger_control Null Object** is modified using the **Pose Tool**, the finger geometry moves with the bones. Any areas not successfully captured will however remain in their original position.

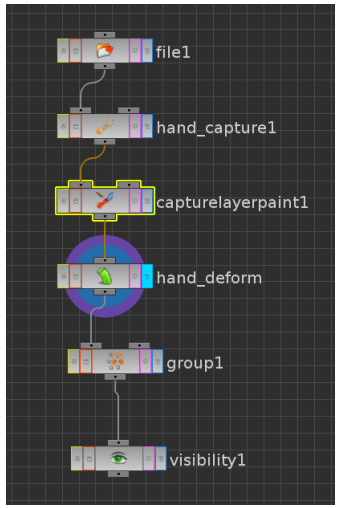


NOTE: Edit Capture Regions can also be invoked after the geometry has been captured. This will allow for any un-captured areas of geometry to be recaptured.

PAINTING CAPTURE WEIGHTS

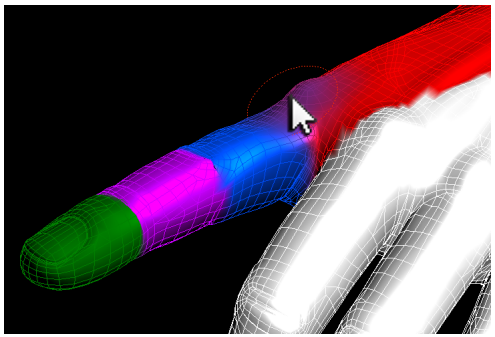
In order to better control the deformation of the finger geometry, the Capture Weights can be painted. This allows for more bespoke association of geometry to the finger bones.

With the mouse over the **Viewer**, press **TAB** and type **Paint Capture Layer**. Select the hand geometry and press **ENTER** to confirm the selection.



The interface will switch to the Geometry Level of the hand geometry, where additional nodes have been automatically created to capture and deform the geometry. A **Capture Layer Paint SOP** has been inserted before the hand_deform node.

Use **CTRL + MMB** to select the bone whose geometry associations are currently incorrect. The geometry can now be painted. Paint options for this tool can be activated by **RMB** in the Viewer.

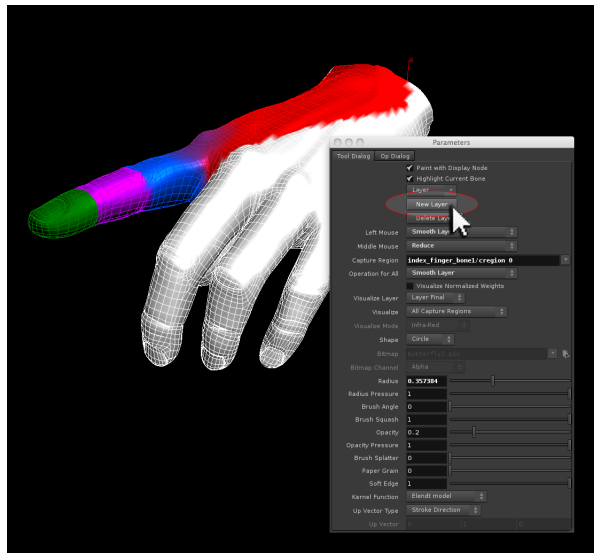


Capture weights can be also be smoothed to help create better deformations around the joints of the fingers. A good tactic when painting capture weights is to continually reposition the finger bone, so all deformation possibilities can be accommodated in the painting process.

CREATING ADDITIONAL CAPTURE PAINT LAYERS

Rather than painting all of the weights information in one go, layers can be assigned to aid the painting process. This means capture paint detail can be incrementally layered in new layers. If the painting of one layer goes wrong, it can simply be deleted without affecting the capture paintwork already done.

New Capture Paint Layers can be created by pressing p with the mouse over the View Pane. This will activate a floating Parameters window that also contains the dialog for the Paint Capture Layer Tool itself.

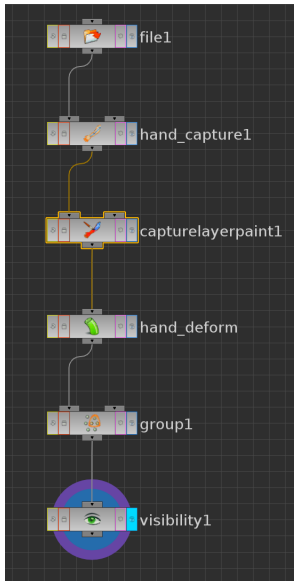


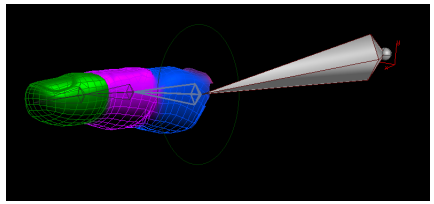
TEMPORARILY REMOVING BONE INFLUENCE

Sometimes it is necessary to view the geometry in its un-deformed state. Rather than reseting any bone controllers, bone influence can be deactivated by going to the main menu **Edit > Objects**, and from its sub menu choosing one of the **Bone Kinematics Overrides**.

VISIBILITY DISPLAY

Another useful SOP for both modeling and painting capture weights is the **Visibility SOP**. This operator will allow for the visibility of geometry to be hidden. This does not delete the geometry or affect the rendering of the object. It simply hides geometry in the Viewer.

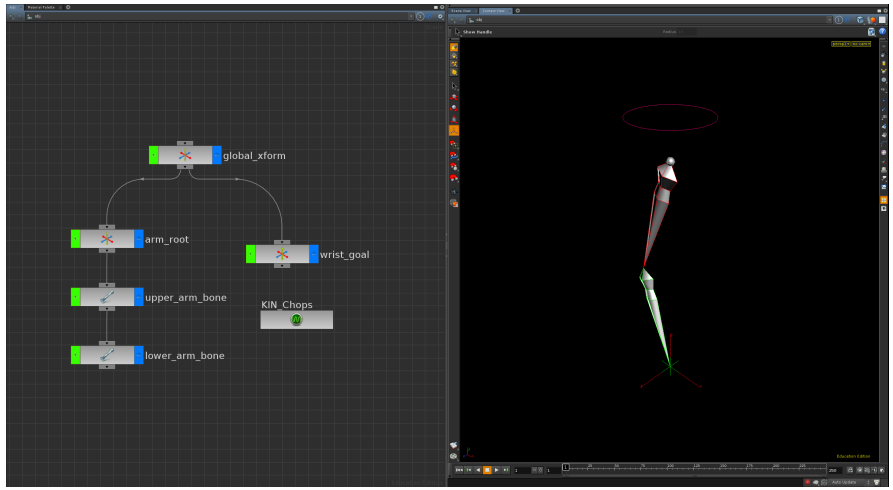




In this example a Visibility SOP can be utilized to hide all of the geometry apart from the index finger. This allows for greater precision when painting weights, and minimizes error when painting weights in hard to reach areas. **See file capture_geo_end.hipnc**

MUSCLE SYSTEMS

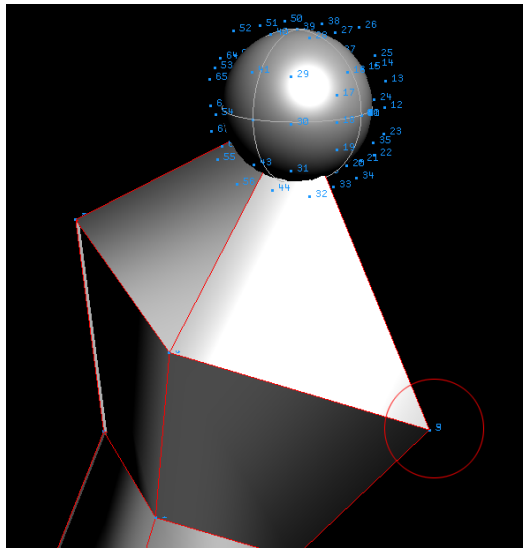
Houdini is capable of creating dynamic muscles for use with characters. If muscle systems are used, they can only be done so as a different capturing technique to capturing weights. It is not currently possible to mix muscles with captured weights.



Open the scene muscles_begin.hipnc. This scene contains a bone arm setup. Inverse Kinematics have been assigned to the bones, and the arm is controlled by the **wrist_goal** Null Object. This example will look at creating a bicep muscle system and procedurally deforming it relative to arm bone movement.

CREATING MUSCLE ATTACHMENT POINTS

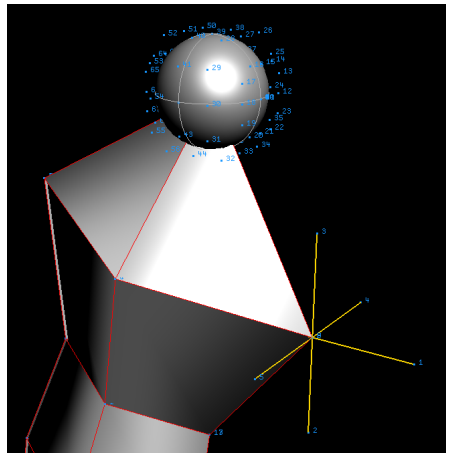
Muscles can either be attached to characters through the positioning of independent Null Objects parented to bones or through point attachment to the bone geometry itself. In this example, direct point attachment is utilized, to allow for bones and muscles to kept separately in the Network Editor.



When points and point numbers are displayed in the Viewer, examination of the **upper_arm_bone** reveals point number **9** as an optimum anchor point for the bicep muscle. An anchor point can be created at this location by utilizing a **Rivet Object**.

In the **Network Editor** create a **Rivet Object**. Initially it will return an error until its parameters are configured. In the **parameters** for the **rivet1** object specify:

Rivet Geometry **../upper_arm_bone**
Point Group **9**

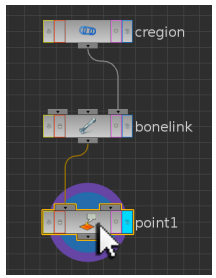


This will lock a null object to the specified point. Rename this object from **rivet1** to **bicep_rivet_upper.**

Repeat this process to create a **second Rivet Object** at the same point location for the **lower_arm_bone**. Rename this Rivet Object to **bicep_rivet_lower**.

CREATING A MID MUSCLE BULGE

Head into the **Geometry Level** of the **upper_arm_bone**, and append a **Point SOP** to the Network chain. This will be used to generate attributes on the arm bone geometry that can help orientate the direction of Rivet Objects.



In its **parameters** specify:

Standard >

Add Normal

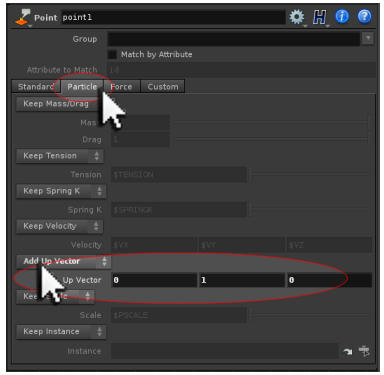
Normal $NX $NY $NZ

Particle >

Add Up Vector

Up Vector 0 1 0

NOTE: In H15, the default Point SOP normal declarations of @N.x, @N.y and @N.z will set all the normal information to 0 unless a Normal SOP is appended before it. Using $NX, $NY, $NZ will however automatically assign normal based on the incoming geometry.



Doing both of these operations will allow for a third Rivet Object to not only be positioned relative to the bone, but also orientated to it.

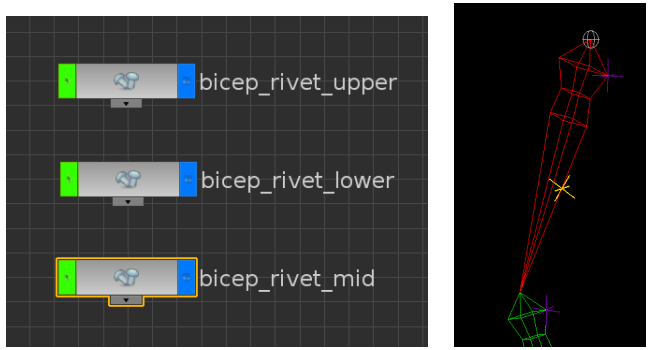
Return to Object Level, and create a third Rivet Object. In the **parameters** specify:

Rivet Geometry ../upper_arm_bone

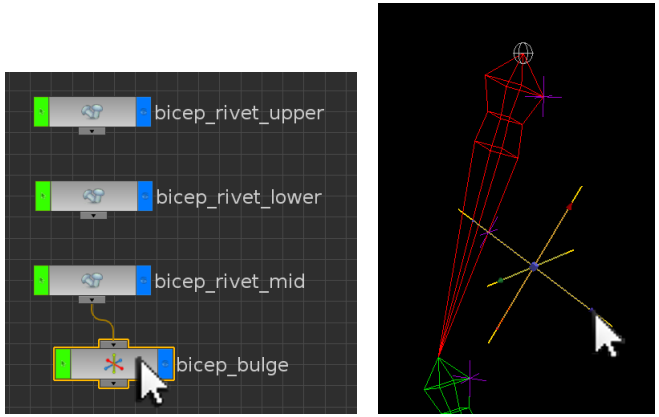
Point Group 9 25

☑ **Use Point Vector Attributes for Rivet Frame**

This will locate this Rivet Object halfway between points 9 and 25 (the upper region and tip of the upper_arm_bone), and orientate it approximately to the bone geometry. **Rename** this **Rivet Object** to **bicep_rivet_mid**.



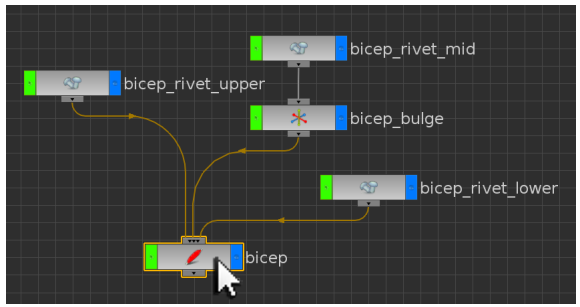
As a final step, append a **Null Object** to the **bicep_rivet_mid** object, and in the **Viewer** translate it slightly away from the rivet location. Rename this **Null Object** to **bicep_bulge**.



See muscles_stage1.hipnc

CREATING THE BICEP MUSCLE

In the Network Editor create a **Muscle Object** and rename it to **bicep**.



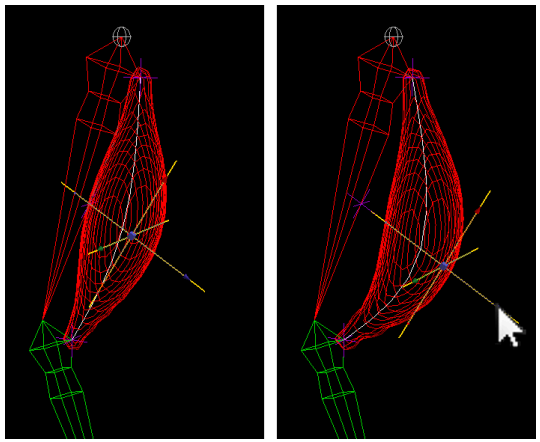
Wire the output of the **bicep_rivet_upper** as its **first input**; the output of the **bicep_bulge** as its **second input**; and the **output** of the **bicep_rivet_lower** as its third input.

In the **parameters** for the **Muscle Object** specify:

Muscle Name $OS

This will ensure that the internal muscle name matches the name of the muscle object node.

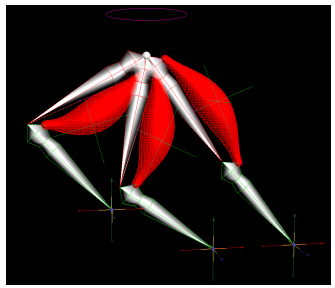
In the **Viewer**, a bicep muscle will have been created. The muscle is attached between the upper and lower anchors, and its shape is controlled by **bicep_bulge Null Object**.



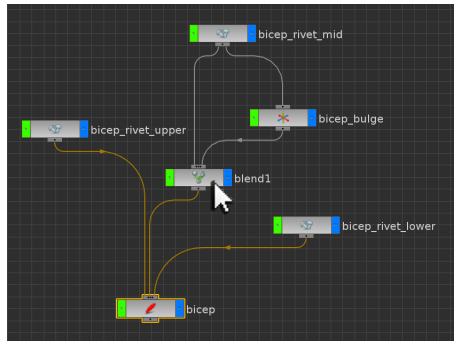
The Muscle Object is simply a series of Metaballs swept along a curve and scaled accordingly.

CREATING MUSCLE DEFINITION

At present the bicep muscle remains a relatively uniform shape when the arm is moved around using the **wrist_goal**. Some muscle deformation occurs; however it is visually minimal. A more naturalistic muscle behavior would be to have bulging occurring as the muscle is contracted, and bulging relaxing occurs when the muscle is stretched.

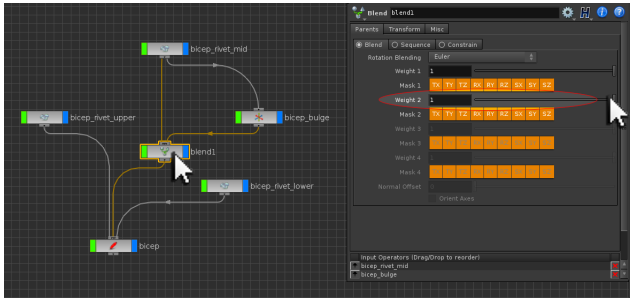


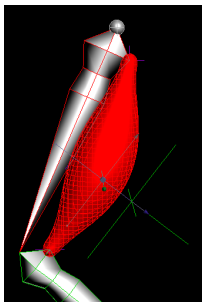
More naturalistic muscle deformation can be procedurally achieved by utilising a **Blend Object**, to blend the muscle shape between location described by the **bicep_rivet_mid Object** and the location described by the **bicep_bulge Null Object**.

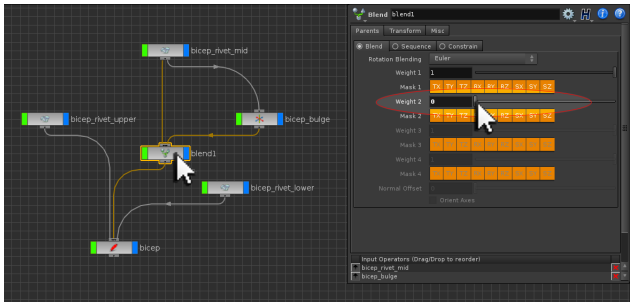


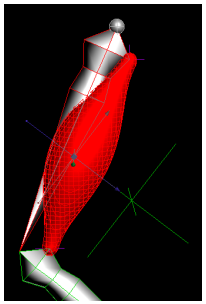
In the **Network Editor** append a **Blend Object**, to **the bicep_bulge Null Object**. Modify the inputs of the Blend Object so that the **bicep_rivet_mid** object is the **first input**, with the **bicep_bulge** object as its **second input**.

When the parameters of the **Blend Object** are adjusted, the bicep muscle blends between and contracted position and a relaxed position.









See muscles_stage2.hipnc

MUSCLE SHAPE AND BLEND POSITION CONTROL

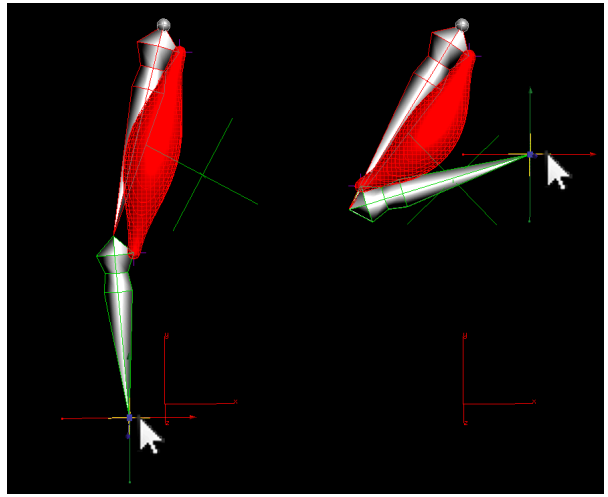
The shape and position of the bicep muscle can be procedurally controlled by calculating the angle between the upper and lower arm bones. This can be done by utilising the **boneangle()** expression function. As this angle will be utilised to drive the blend, it needs to be re-ranged between a value of 0 and 1. This can be done using the **fit() expression** function.

In the parameters for the **Blend Object** specify:

Weight 2

fit(boneangle("../upper_arm_bone","../lower_arm_bone"), 0, 180, 1, 0)

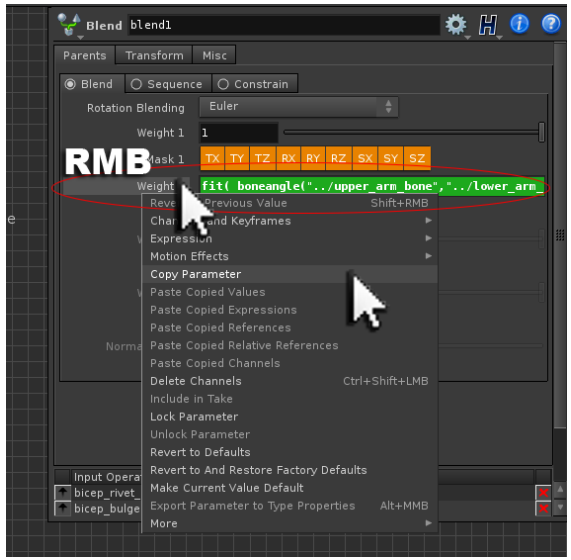
When the arm is fully outstretched, a bone angle difference of approximately 180 degrees is recorded by the boneangle() expression function, and re-ranged to a value of approximately 0 by the fit() expression function. This sets the muscle position to its relaxed state.



When the arm is fully bent, a bone angle difference of approximately 0 degrees is recorded by the boneangle() expression function, and re-ranged to a value of approximately 1 by the fit() expression function. This sets the muscle position to its contracted state.

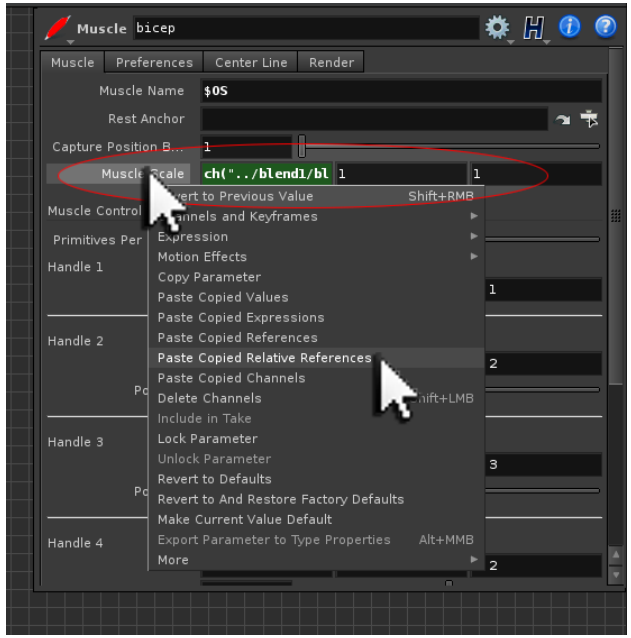
MUSCLE SCALE

This same procedural technique can also be applied to the scale of the muscle itself, where it becomes slightly larger if the muscle is contracted.



In the parameters for the Blend Object, **RMB** on the **Weight 2** parameter label and from the resulting contextual menu choose **Copy Parameter**.

Locate the **Muscle Scale** parameter located in the **Muscle Object**, and **RMB** on its parameter label. From the resulting contextual menu choose **Paste Copied Relative References**.



This will paste a channel reference into the x parameter of the Muscle Scale. Modify the expression to read:

Muscle Scale [x]

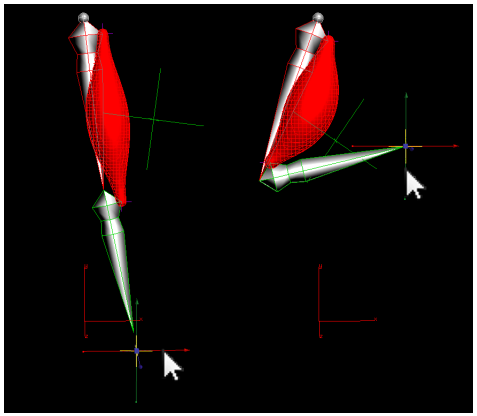
1 + (ch("../blend_location_as_arm_bends/blendw2")/5)

This expression can now be channel referenced into the remaining two Muscle Scale parameters:

Muscle Scale

1+(ch(etc... ch("musclescalex") ch("musclescalex")

Now when the bicep muscle is fully contracted, its scale is increased to approximately 1.15 in all three axes. When the bicep muscle is fully relaxed, its scale is approximately 1 in all three axes.



Visually this equates to a slight expansion in the bicep whenever the arm is bent. The aesthetic of the bicep can be further refined by modifying the position of the **bicep_bulge Null Object** and by the scaling factor assigned to the overall muscle scale. **See muscles_done.hip**

GEOMETRY CAPTURE AND MUSCLE SYSTEMS

At present it is only possible to **Capture Geometry** as either **Regions**, **Proximity** or **Metaballs**. It is not possible to create combinations of these techniques. This means that if a muscle-based solution for surface deformations is opted for, muscles will have to encompass the entire character or creature.