



Master Classes

Instructor Notes – Animator Friendly Rigging Part II

Jason Schleifer

## **Animator Friendly Rigging**

Creating animation rigs which solve problems, are fun to use, and don't cause nervous breakdowns.

**Jason Schleifer**



Master Classes

## Instructor Notes – Animator Friendly Rigging Part II

Jason Schleifer

### Example – Standard Bipedal Rig

In Animator Friendly Rigging Part I, we learned how to create a bouncy ball rig using a set of standards to ensure quality rigging techniques. We can take those same lessons and extrapolate them to help us create other types of rigs. One of the most common rigs needed in the industry today is a bipedal rig. That doesn't mean a rig that works on a bicycle, or a dual-pedestrian rig.. it's a character that stands on it's own two feet. A humanoid character. You know, "people"-ish.

So how do you start creating a bipedal rig that will work for your animators? The best way is to approach it the same way we looked at creating the ball rig: we looked at reference of what we expected the rig to be able to do.





## Master Classes

### Instructor Notes – Animator Friendly Rigging Part II

Jason Schleifer



*Figure 1 - Human character doing various "humanistic things".. walking, lifting, standing, pushing..*



*Figure 2 - More poses that are standard for humans. Notice the interactions between various parts of the body, elbows on knees, hands holding up the head, etc.*

As you can see from these images, the body can get into a number of various poses, and our control structure must handle as many of these situations as possible.

However, we're not just creating a rig which can simply be posed, we have to create a rig which will support the type of motion the animator will want to

create. So how do we determine what motion our rig will need to handle? The best way is to approach it the way an animator would: with *reference*.

#### Acquiring Reference

Go and grab yourself a video camera or web cam and shoot reference of the type of motion your character will be expected to create. If you don't have a video camera, search online for video. *Google Video*<sup>i</sup>, *You Tube*<sup>ii</sup>, and the *BBC Motion Gallery*<sup>iii</sup> have searchable video archives of motion just ripe for analyzing.

One of my favorite types of video to analyze involves the sport of Parkour, or free-running. Parkour is the art of running over, under, and through obstacles. It's great reference for seeing what the body can do, and what kinds of extreme motion can be achieved.

Of course, while watching all this amazing parkour reference out on the web I became jealous of the fact that these people had the ability to move with such grace and agility. So some friends<sup>iv</sup> and I went out and shot our *own* parkour video. But.. alas.. we are not so graceful.

You can view the movie for yourself by looking in the included *movies* folder. The file is called *parkour-H264.mov*.



Figure 3 - Two images from the parkour video



## Master Classes

### Instructor Notes – Animator Friendly Rigging Part II

Jason Schleifer

By analyzing the reference video, you can break down what type of motion will be necessary for your character. Take two or three frames of a movement and analyze it. Discover what is moving, what is staying still. Where to the moving parts pivot from? Where is the force of the motion? How would you want to move that bit of the body if you were to animate it? By doing this analysis, you can start to get an idea for what an animator does, and see how an animator thinks about motion.





*Figure 4 - the series of images demonstrate four poses for the action of pushing oneself up from a sitting position. Notice what part of the body moves while other parts stay still.*

The first thing you want to do is segment out the human into workable chunks. This can be done by thinking about how the character you're creating moves, what individual pieces are involved, and trying to determine the easiest way to dissect the character. A human is relatively simple due to its familiar structure. But this same trick can be applied to almost any creature or shape. Analyze the structure and determine clean break points based on the anatomical structure and on how the creature moves. It's imperative to start thinking about the creature as one that will be in motion even at this early point in the rigging process. Without the knowledge of how the creature is going to move and what types of actions will be required, it is impossible to rig a creature effectively.



## Master Classes

# Instructor Notes – Animator Friendly Rigging Part II

Jason Schleifer

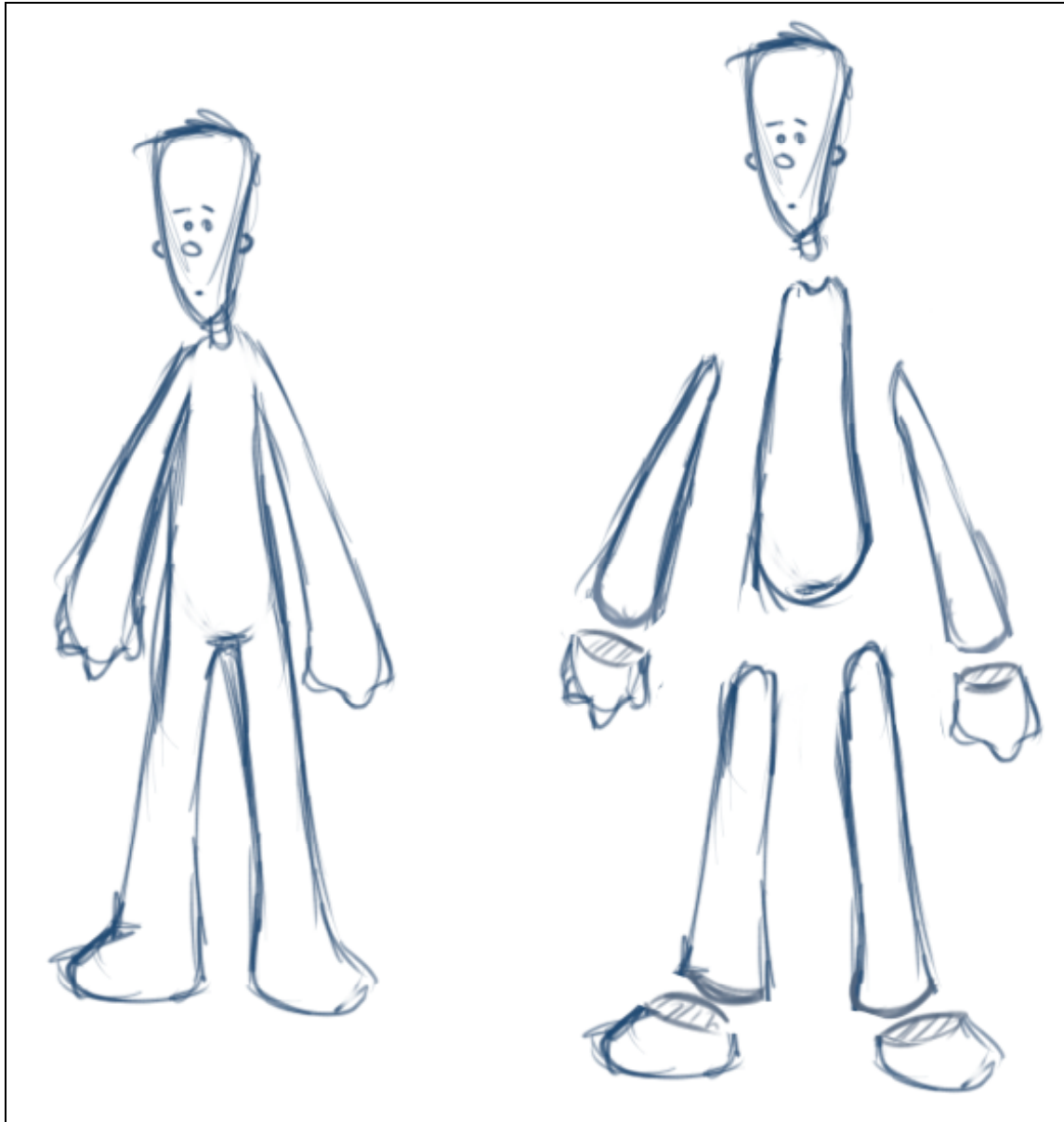
### Creating the rig - Where to start?

If you've had any experience with rigging before, or even talking about the body, think about how people talk about various body parts. "He put his arm on her shoulders." "She nodded her head". "He tweaked his back". "She had long legs". "That dude had wings, he must be a mutant".

If you analyze the way people *think* about the body, you can start to break the body up into convenient segments-- parts that are unique and make sense to the animator to be broken apart.

Another way to think about it is to break up the body until there's "one of each". What that means, is how many torso's are there? One. Arms? Two. Legs? Two. Head/neck? One. Looking at it this way, you can break apart the rig so you have *one of each*.





*Figure 5 - Imagine a "popping" noise as the body separates.*

Based on this, we can separate the body into the following sections:

- Head
- Torso
- Arm
- Hand
- Leg
- Foot

You can probably break these bits apart more, for example the hand can be broken apart into finger and thumb. However, for now this level of granularity will work fine for getting the general rig working correctly.

So where do we begin? We've got these six sections, what do we do first?

The best place to start is with the part of the body that causes the most motion, usually the part of the body with the greatest amount of mass. With a human, motion comes from the hips and torso. They affect every part of the body. So the torso is probably the best place to start when it comes to rigging.

## Rigging a Torso

But how do you start? Take a look at the reference drawings again and analyze how that part of the body is supposed to move. What is the range of motion? What type of motion is there? What are the limits? If you need more examples, get more reference.





There are four reference videos included with the documentation located in *movies/back/*. Watch each of the videos to begin analyzing the back.

If we look at some of the actions that we've drawn & how the torso relates, we can start to get an idea as to what type of movement is necessary.



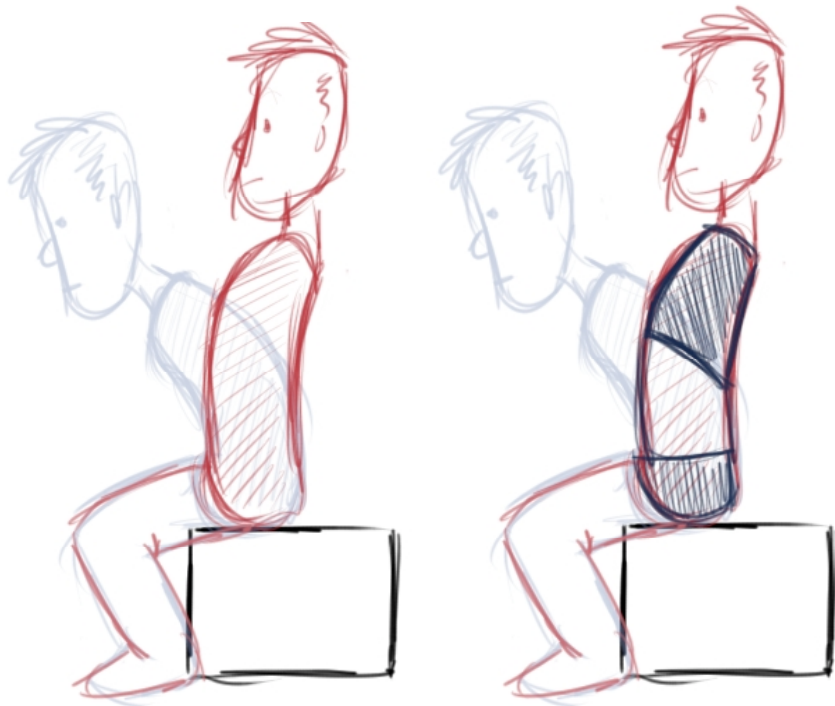
*Figure 6 - torso motion as part of some basic actions*

If we analyze these actions, we can start to abstract the motion into its various parts. The torso consists of two main parts.. the hips and the shoulders. If you

think about how you move around and how those two parts relate, we can extrapolate that they're the most important part of the back.

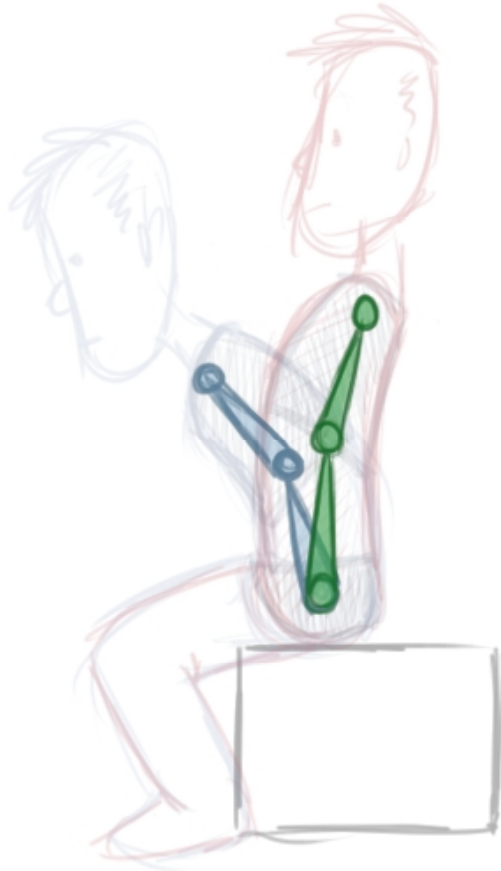
**It's the relationship between the hips and shoulders that help define almost any action.**

Let's take a look at one of the simplest animations involving the torso. That would be a character sitting down.



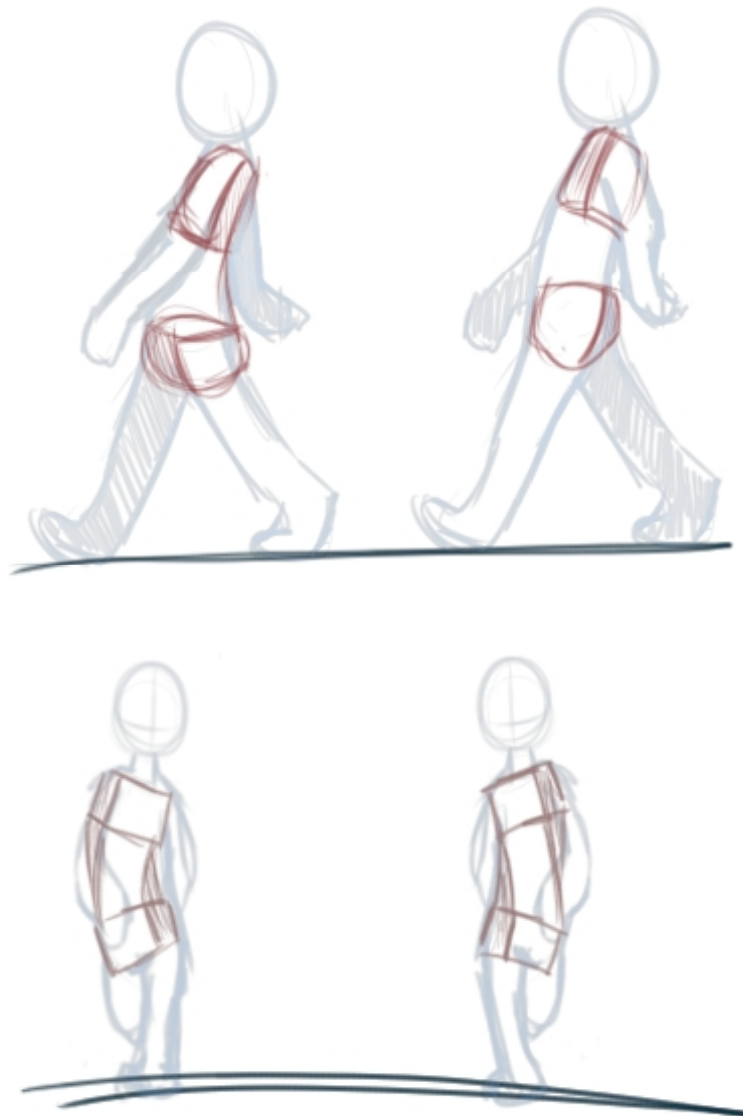
*Figure 7 - a character sitting down with hip/torso broken apart.*

The relationship here is quite simple, the torso is rotating about the hips. So if this were all the torso would need to do, we could simply get away with a very simple joint structure.



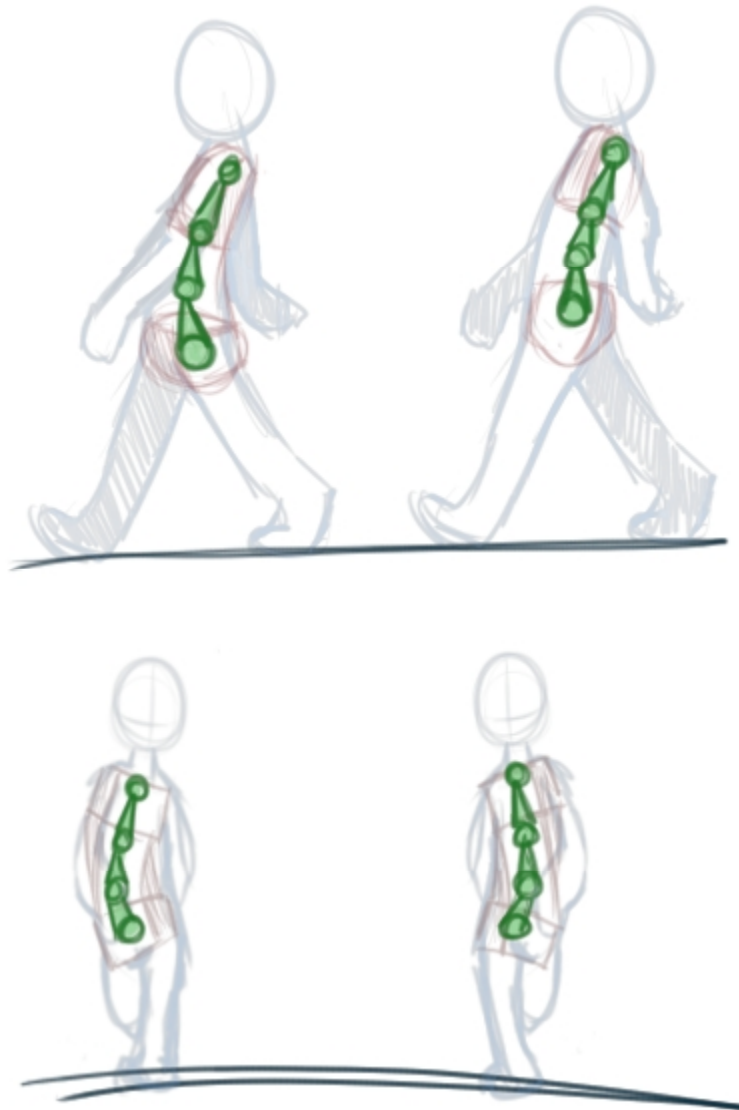
*Figure 8 - Simple joint structure for the back.*

Unfortunately, things are rarely that simple. Let's take a look at another more complicated example.. a walk.



*Figure 9 - hips/shoulders countering each other in a walk*

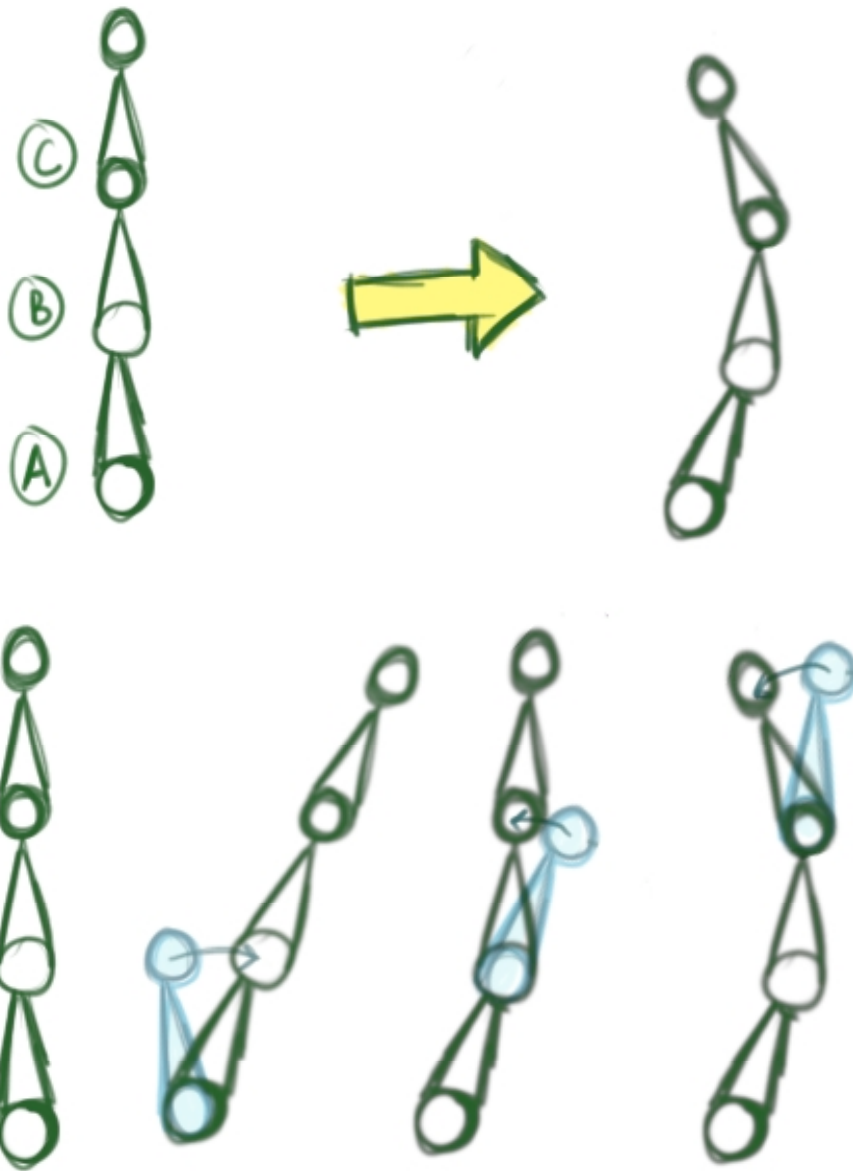
Here, the relationship is a little bit more complex. As you can see, the hips and shoulders counter each other. They need to rotate in three dimensions: front and back, side to side, and twisting. In this case, a joint chain can still be used, although it would be a bit more complicated.



*Figure 10 - joint chain to handle arcing of back in a walk*

While this seems like a perfectly reasonable solution, it begins to introduce some complexity when working out how these chains will be animated.

Without any complicated rigging, the joint chains will work as a **forward kinematics** solution. This means that each chain, when rotated, will affect the children.



*Figure 11 - to get the spine to look like the result in the top image, you must rotate each individual joint, as displayed in the bottom image.*

In the above image, in order for the chain to look like the result, it's necessary to rotate A, then B, then C. This is a pretty simple solution and works well, as long as you don't want to change the position of A without changing the position of C. As soon as you want to do that, then this solution breaks.

A great example is if you were animating somebody wrestling.



*Figure 12 - character in a classic "wrestling" pose. Notice he's not thrilled about wrestling, because it's a violent sport, and our character is a pacifist. But for the sake of instruction, he's willing to demonstrate the technique. What a trooper!*

If we've posed the character bending over, and then decide to raise the hips:



*Figure 13 - Hips raised*

This can be incredibly difficult, especially if you don't want to affect the position of the head and shoulders:





*Figure 14 - difference between hips lowered and raised. Notice the lack of motion in the head and shoulders.*

Thus, we now know that we need more than just a simple joint chain. We need a solution that does the following:

1. Allows for rotation of hips and shoulders
2. Allows for rotation in all axis – Bend, Side to Side ,and Twist
3. Allows for independent motion of shoulders and hips.

With these three needs, we can start talking about possible animation controls.

#### **Control possibility 1 – FK control, pivot at root.**

**File:** *torso\_fk.ma*

**Pros:** Predictable motion  
Small number of controls  
Similar to real-world counterpart  
Easy to move in arcs.

**Cons:** Difficult to make changes



## Master Classes

# Instructor Notes – Animator Friendly Rigging Part II

Jason Schleifer

Independent motion not easily achieved.  
Spine compression difficult.

### **Control possibility 2 – FK control, pivot at mid-back.**

**File:** *torso\_midBack.ma*

**Pros:** Easier to create some independent control  
Small number of controls  
Easy to move in arcs.  
Spine compression easier.

**Cons:** Still doesn't handle total control independence  
Can be even more confusing than simple FK chain.

### **Control possibility 3 – Independent translation/rotation control of hips and shoulders**

**File:** *torso\_ik.ma*

**Pros:** Total independent control of hips and shoulders  
Easy to tweak  
Small number of controls.  
Allows for unique cartoony poses.  
Allows for stretching, if required.  
Spine compression piece of cake.

**Cons:** Arcs can be more difficult to achieve

### **Control possibility 4 – combination fk control and ik control.**

**File:** *torso\_both.ma*

**Pros:** all the pros of control possibility 3.  
Arcs are much easier to achieve.

**Cons:** Once you move the spine away from the controls, it's harder to create arcs.

Great! So there are four possible controls. But do they cover every situation? They seem to cover most animation problems pretty well, anything from character's walking, to talking, to sitting, to getting up, to wrestling.

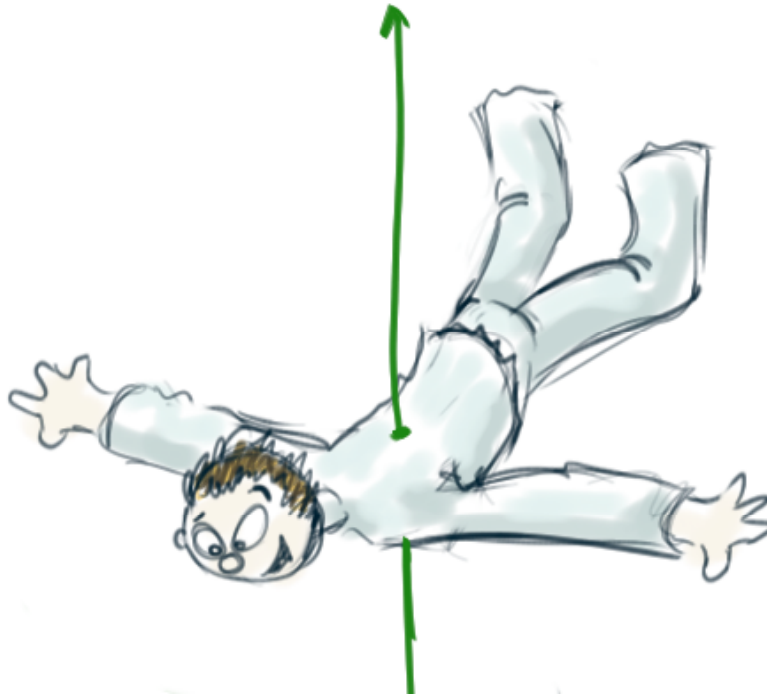
But what if our character is flying through the air?



*Figure 15 - character falling, where does it make the most sense to put the pivot?*

If you have a character animated flying through the air, does it make sense for their pivot point to be located around their hips? Probably not. If the pivot were located there, and you wanted them to orbit around, it would look odd. It would actually look like they were on a string being pulled around, and the animator would have to work extra hard to get it to move correctly.

What you actually want is the pivot to be located in the center of mass for the character.



*Figure 16- best place for the pivot when someone is in the air*

So what does this mean for our rig decision? Does it mean we should move the main pivot for the body to the center of mass of the character?

No, because that wouldn't work for a character who's sitting down. Every time they bent over, you'd have to counter-animate. In the case of a character sitting down, you want their pivot right at the base of their hips, where their buttocks is contacting the ground.

So it appears that we have another requirement for our rig: The ability to allow the animator to move the pivot of the main control to the location they need based on the shot's requirements.

Let's take a look at our final list of requirements for our torso control:

#### **Torso Animation Rig Requirements**

1. Allows for rotation of hips and shoulders



## Master Classes

# Instructor Notes – Animator Friendly Rigging Part II

Jason Schleifer

2. Allows for rotation in all axis – Bend, Side to Side ,and Twist
3. Allows for independent motion of shoulders and hips.
4. Allows for relocation of pivot.

## Torso Rigging Toolkit

Before starting on developing the back rig, I think it's important to actually put together a "toolbox" of techniques we're going to use. I find it extremely helpful to do this when working out new ideas, mainly because it helps solidify techniques before getting caught up in all the finite details of a particular rig.

So what are we going to need in our toolbox in order to work with the back rig?

1. Fk Joint control for the basic body rig.
2. Independent control for the shoulders and hips
3. Spline IK to guide objects between shoulders and hips.
4. Stretching of joints based on the shoulder and hip translation
5. Pivot relocation

Let's take a look at each of these tools, and make sure we understand how they work *before* getting involved in any great depth in our rig.

### FK Joint Control

Most joints and controls that you work with on your body are going to be based on FK rotation of joints. They work as one would expect any hierarchical relationship to work: Rotate/move the parent, it affects the children.

Let's take a look at some specifics on the joints and make sure we understand enough about them to know what we're going to be using them for.

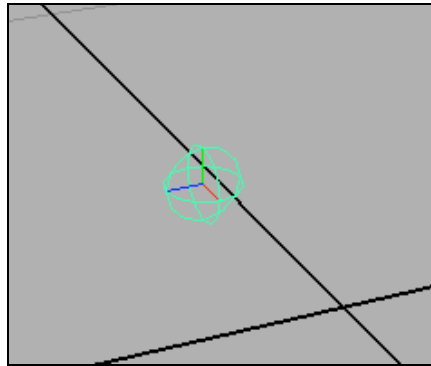
#### 1. Create a New Scene

- Choose **File > New**

#### 2. Use the Joint tool to create a joint

- Choose **Skeleton > Joint Tool**
- Click somewhere in the 3d view to create a joint

- Hit “q” to finish creating the joint and switch back to the **Select** tool.



*Figure 17 - Joint created*

What you have now is a joint. This is a special type of object in Maya that is unique in that it has a location, rotation, *and* an *orientation*. That means it has a position (translate), rotation value (rotate), and also an axis that it orients along (jointOrient).

By itself, it doesn't mean very much (or make much sense), but with a child joint, we can start to see how the orientation can be very important.

### 3. Create a New Scene

- Choose **File > New**

### 4. Use the Joint tool to create a joint segment

- Choose **Skeleton > Joint Tool**
- Click somewhere in the 3d view to create a joint
- Click again to create another joint.
- Hit “q” to finish creating the joint

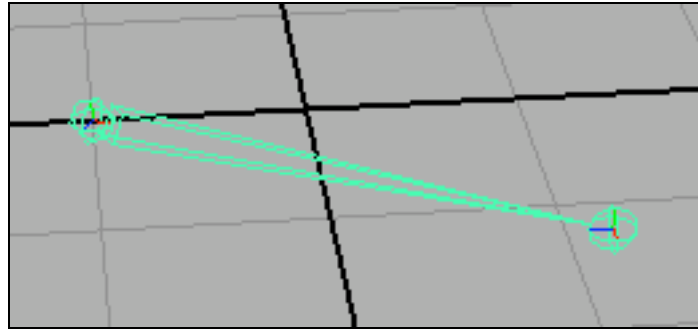


Figure 18 - a joint segment created, 1 joint parented under a second joint.

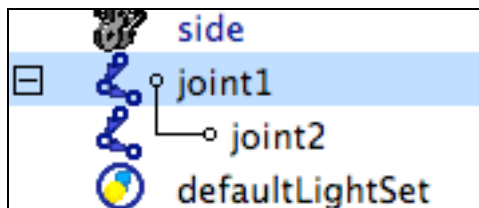


Figure 19 - As you can tell, there are only two joints in the scene, but Maya understands how to draw a joint segment between them

If you look in the outliner, you can see how only 2 joints exist in the scene, but Maya naturally draws a segment between the two joints. This gives us an idea of how these joints can work together to create a *skeleton* for our character.

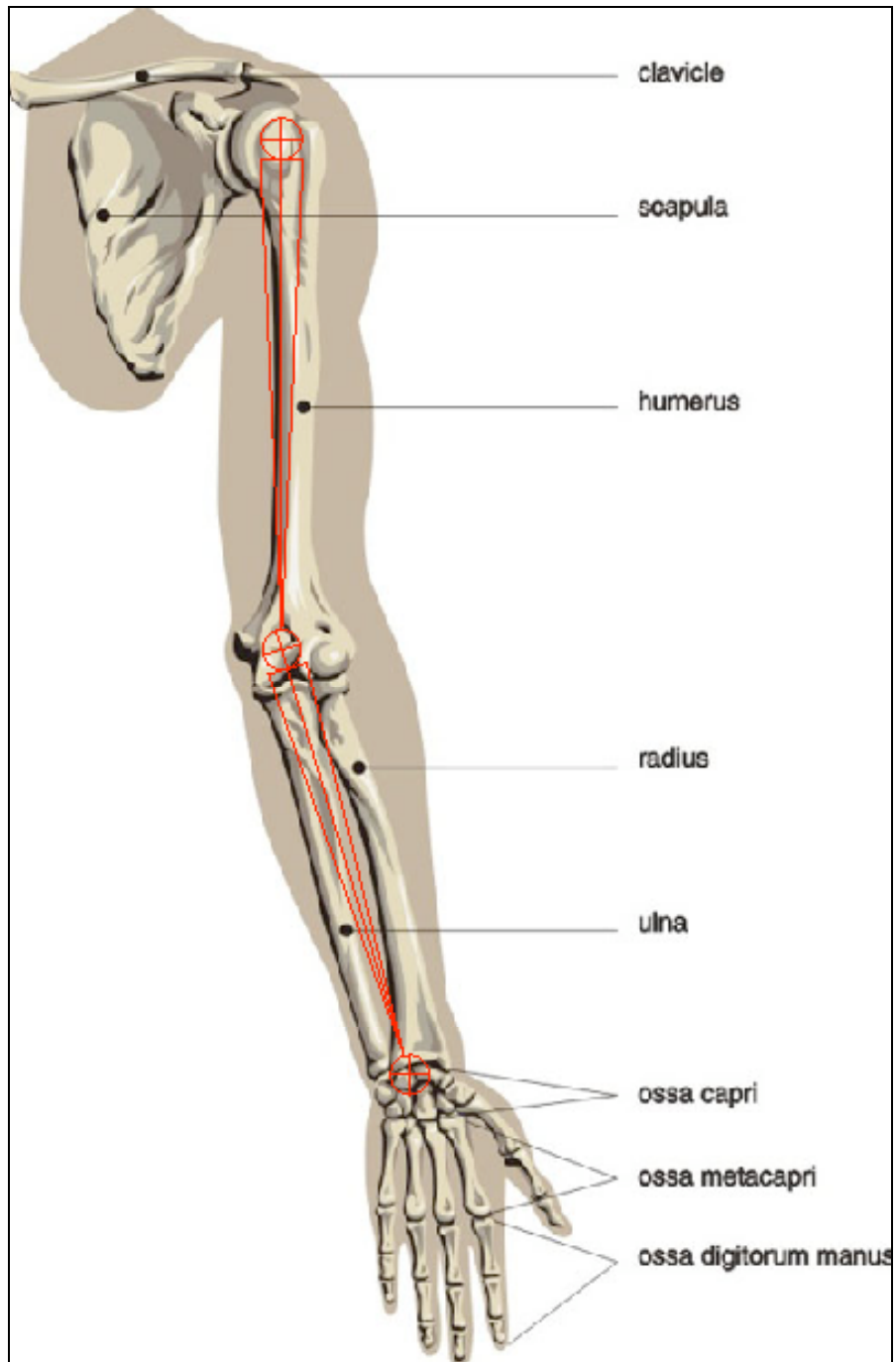


Figure 20 - joints lining up with a human skeleton. Image taken from <http://staticfiles.sabc.co.za/>



Let's take a look at how the joint's orientation affects the joints.

#### 5. Look at the Joint's Orientation

- Select **Joint1**
- Switch to the **Rotate** tool.
- Notice that even though joint1 has a rotation value of **0 0 0**, it's *oriented* towards the child joint. This means that the parent joint (joint1) is *pointing towards its child* (joint2).

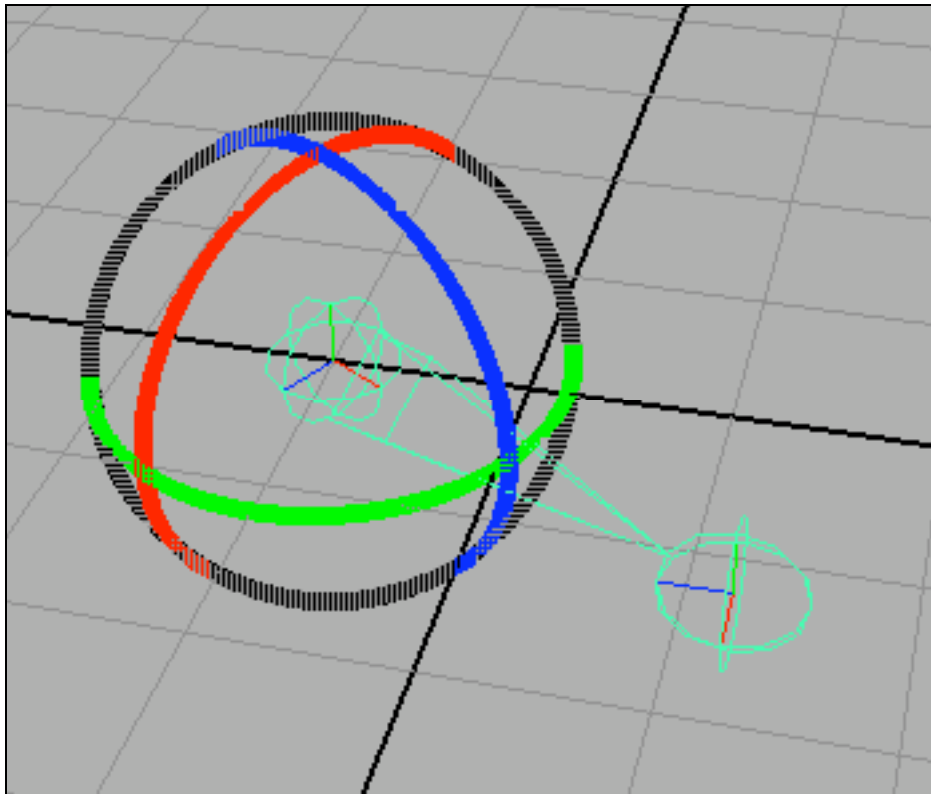


Figure 21 - Joint is oriented towards its child.

This orientation is extremely important, because it allows us to control how our skeleton is going to move. Let's take an example of an upper arm..

#### 6. Create a new scene

- Choose **File > New**

#### 7. Go to a Side View

- Choose **Panels > Orthographic > Side**

#### 8. Create an fk arm without proper orientation

- Choose **Skeleton > Joint Tool > Option Box**
- Set **Orientation** to **none**

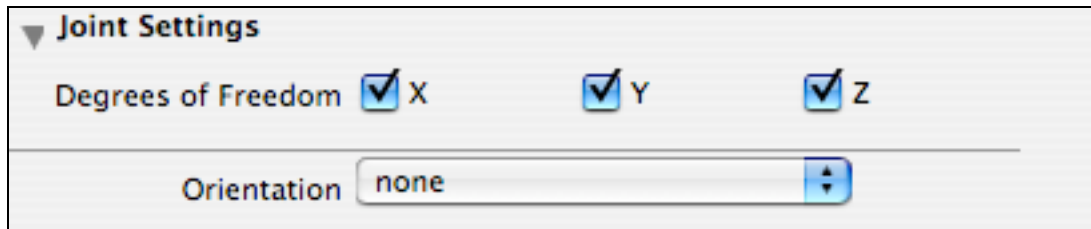
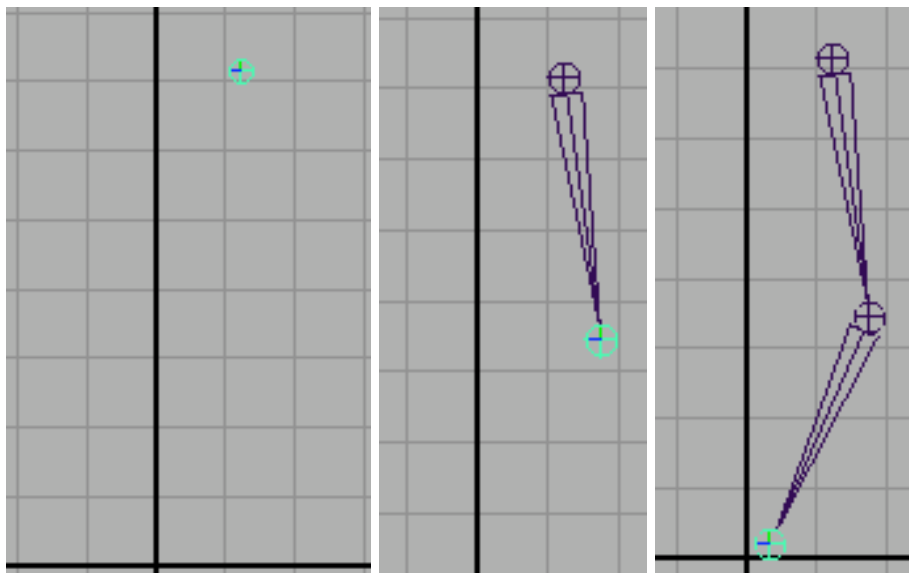


Figure 22 - joint orientation settings, orientation set to none

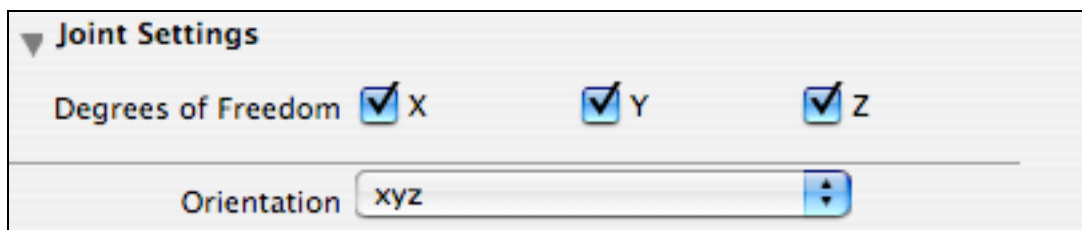
- Click three times in 3 different locations to create an “arm”.



*Figure 23 - Creating an "arm"*

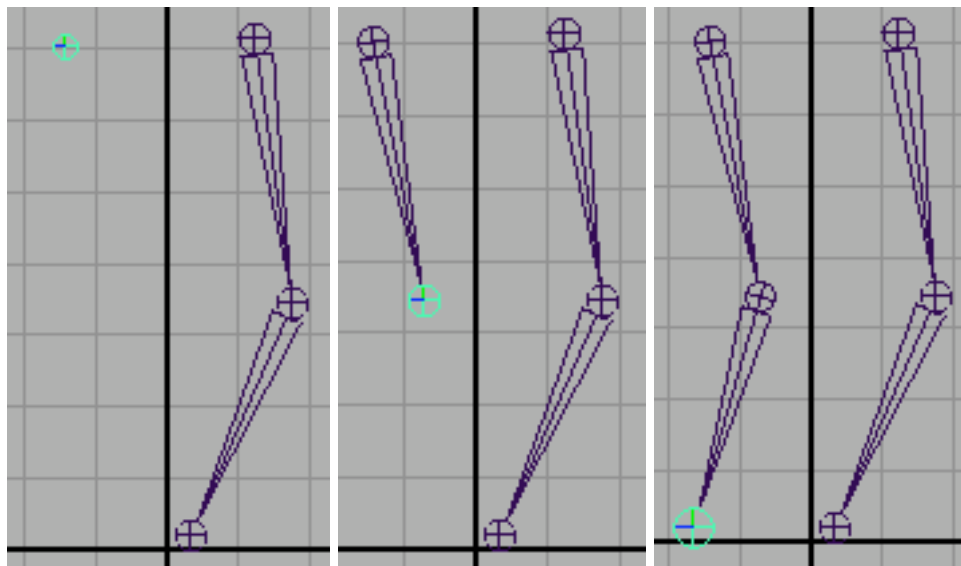
#### 9. Create an FK arm *with* proper orientation

- Choose **Skeleton > Joint Tool > Option Box**
- Set **Orientation** to **xyz**



*Figure 24 - joint orientation settings, orientation set to xyz*

- Click three times, as before, to create an “arm”.



*Figure 25 - clicking three times to create an oriented "arm"*

Let's take a look at the orientations of these joints. You'll be able to see exactly what the difference is between having an orientation of “none”, vs. the joint orienting towards it's child.

#### 10. Display the rotation axis of the “elbow” joints

- Select the two “elbow” joints

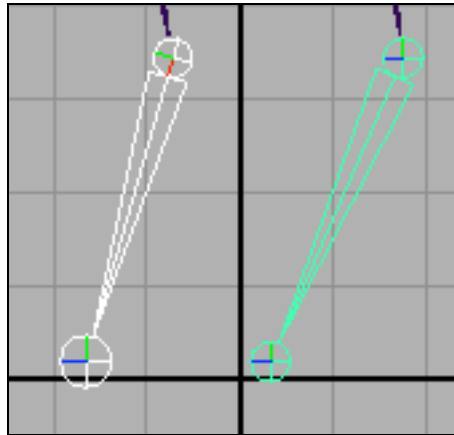


Figure 26 - the two "elbow" joints of the arms

- Choose **Display > Transform Display > Local Rotation Axis**

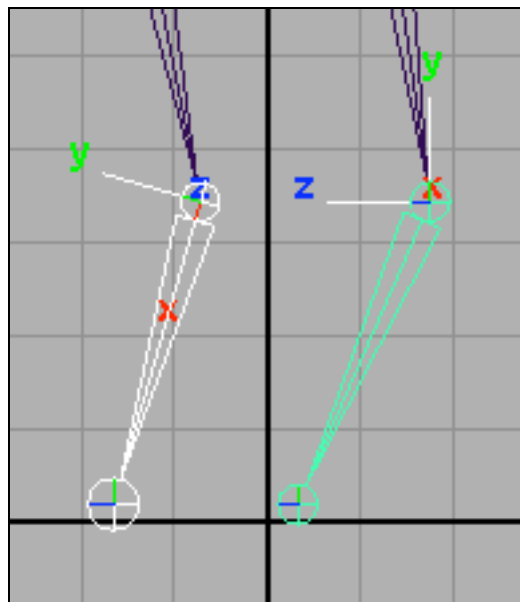


Figure 27 - local rotation axis turned on for each joint



## Master Classes

# Instructor Notes – Animator Friendly Rigging Part II

Jason Schleifer

See how the rotation axis is different for each joint? The one on the right has a rotation axis that matches the world XYZ. The one on the left is pointed down towards it's child.

But how does this really affect us? Is it that big of a deal to have control over the direction of the axis?

Try rotating the joints, and you'll notice a difference immediately, especially if you try and rotate the joint as if you were twisting the forearm. With the joint on the left, the twist happens exactly as you expect – straight down the arm. With the joint on the right, you actually can't get the joint to twist how you want. Try it. Tell me this wouldn't drive you nuts if you were animating with this control on a day-to-day basis. Go ahead. I dare you.

As you'll find throughout the course we'll be manipulating the axis for the joints based on what's the most appropriate for any given situation. Sometimes you *must* have the axis aim directly down the bone. Other times, it's best to aim the joint somewhere else. We'll get into greater specifics as we continue.

## Independent shoulder and hip control

Since independent control over the shoulders and hips for our character is important, it would make sense for us to test that type of control and see how it works.

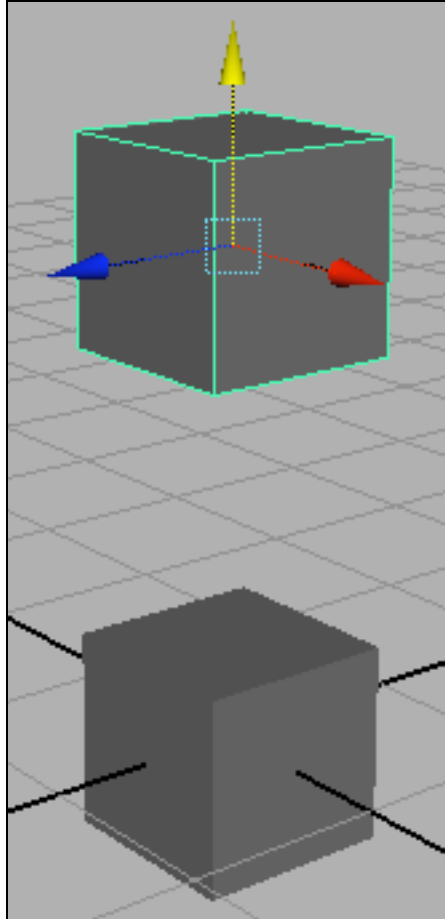
To do this, we'll create two cubes that we can use as examples for the shoulder and hip controls.

### 11. Create a new scene

- Choose **File > New Scene**

### 12. Create two poly cubes

- Choose **Create > Polygon Primitives > Cube**
- Choose **Edit > Duplicate**
- Move the duplicated cube up in y, so it sits above the first cube

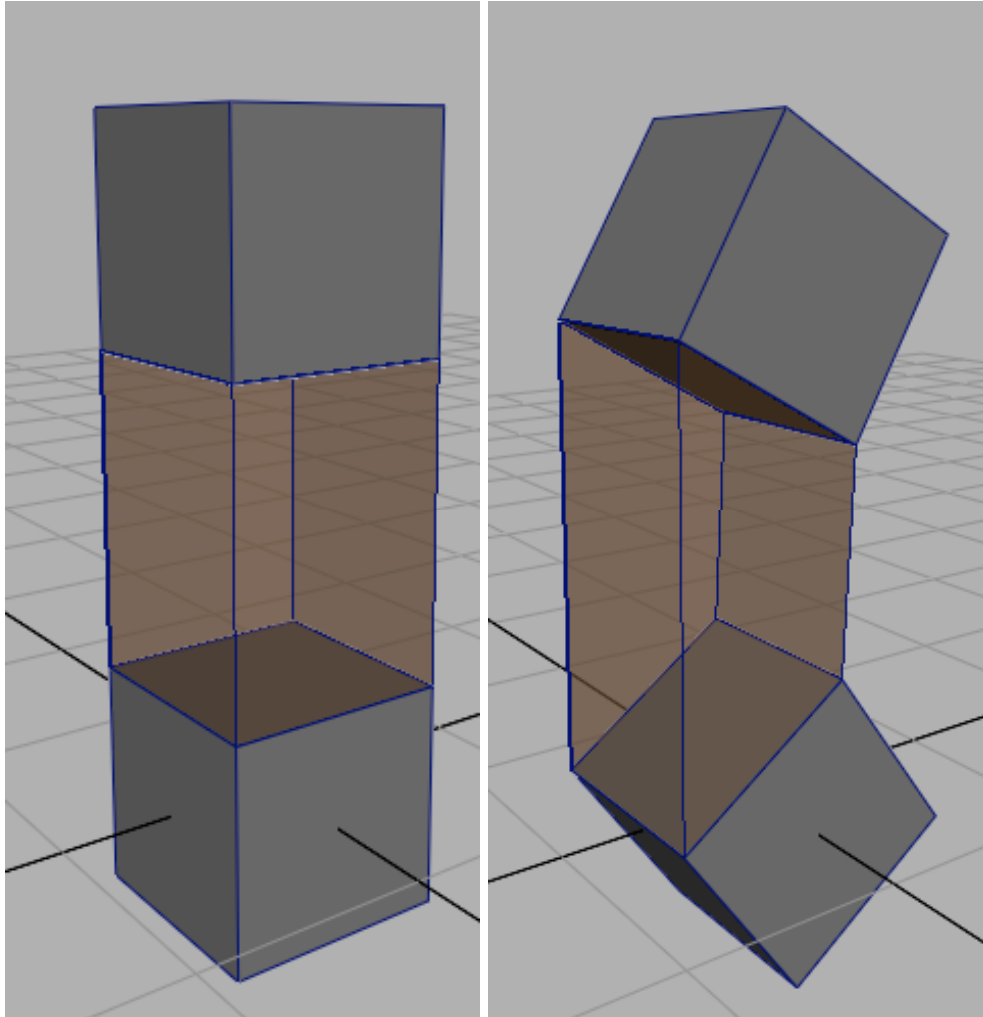


*Figure 28 - two cubes that will be used as temporary shoulder and hip controls.*

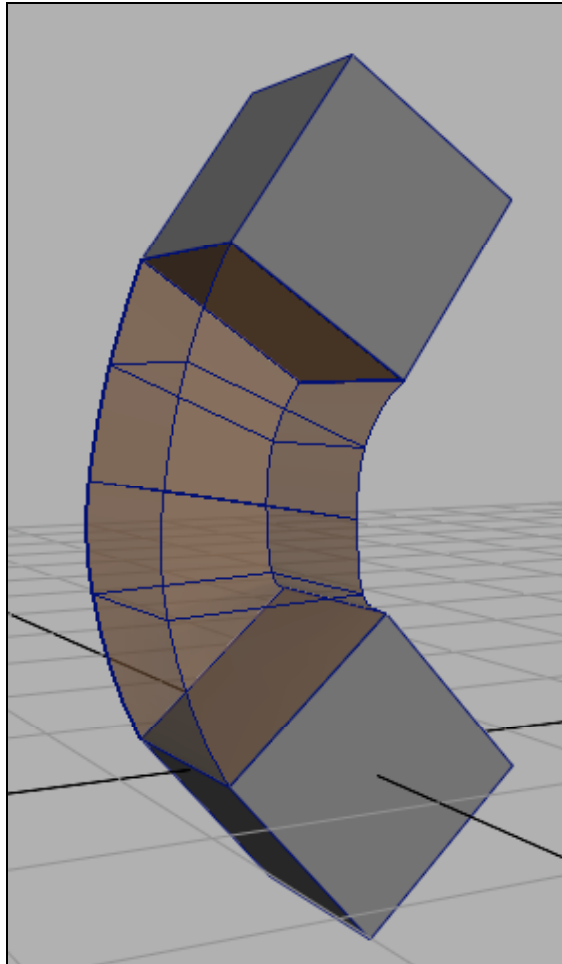
#### 13. Rename the cubes

- Select **pCube1** and rename it **hip\_anim**
- Select **pCube2** and rename it **shldr\_anim**

Move the hips and shoulders around. This action simulates what it would be like to manipulate the hips and shoulders on your character. As you can see, it's relatively straightforward. However, if you look at how a back moves, you know that there's more to it than just the hip and shoulder position. There is all the stuff *between* the hips and shoulders that we need to manipulate as well.



*Figure 29 - Imagine the transparent bit in-between the two cubes is the stuff between your shoulders and hips. A linear connection between the two isn't going to work, we need to control it in a way that looks more natural.*



*Figure 30- This is much more comfortable for your character's guts.*

The idea is that the natural angle of the hips and shoulders will cause everything else to move along with it all nice and easy.

You also want to have the rest of the body twist automatically with the shoulders and hips:



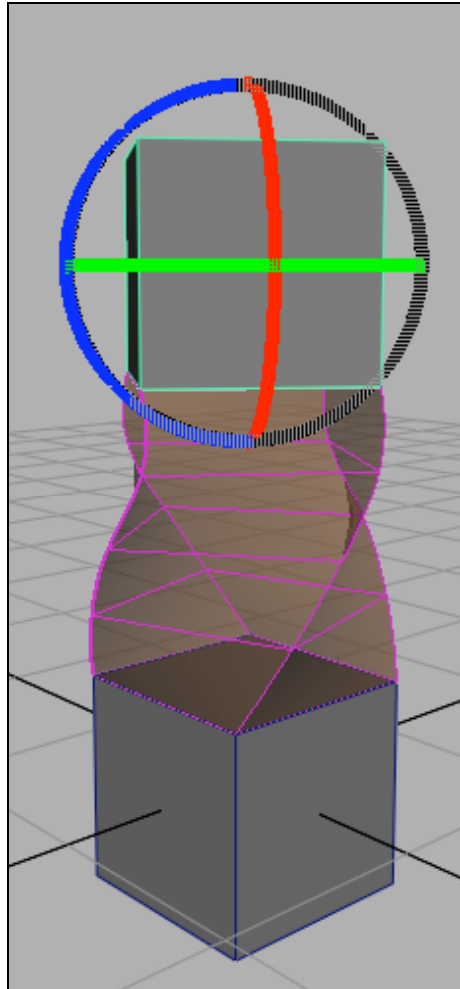


Figure 31 - guts twisting with the hips and shoulders

There are a few different techniques to achieve this effect. In this course I'll be demonstrating the use of the splinelk system, mainly because of it's speed and that it's easy to use. If you have another method of deforming the back, *you're welcome to use it!* There are no wrong answers here, as long as you are able to get the desired result.

Let's go ahead and create a skeleton structure that will go from our hip control to our shoulder control. We will use this example to simply get to know the tools we're going to be using to create the actual spine system later.

#### 14. Create a joint segment for the spine

- Choose **Panel > Orthographic > Side**
- Choose **Skeleton > Joint Tool**
- Click right above the **hip\_anim**
- Hold down **Shift** and click again just below the **shldr\_anim** *note: holding down shift while clicking will ensure that the joint is perfectly vertical*

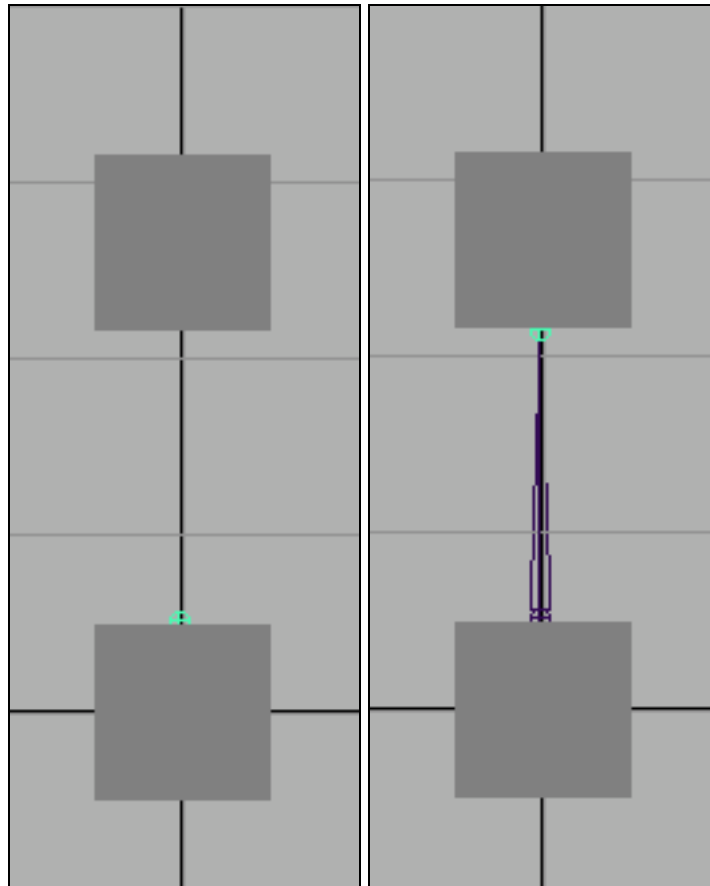


Figure 32 - creating a joint that goes from the top of the hip to the base of the shoulder

Obviously if we're going to create a spine that will bend, we need it to have more segments than just the single segment we've created here. You can go ahead and click a number of times to create the segments you want, or you can use the **js\_splitSelJoint<sup>vi</sup>** tool to do it automatically!

#### 15. Split the joint into 6 segments.

- Select **joint1**

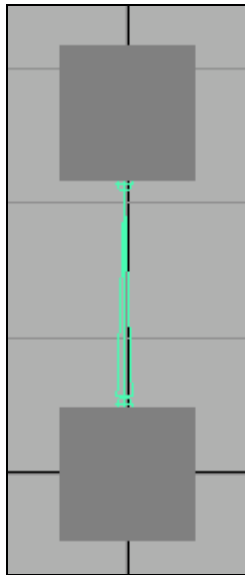


Figure 33 - Selecting joint1

- Click on the **Split Selected Joint** icon in the **Animator Friendly Rigging** shelf
- Choose **6 Segments**
- Click **Okay**.

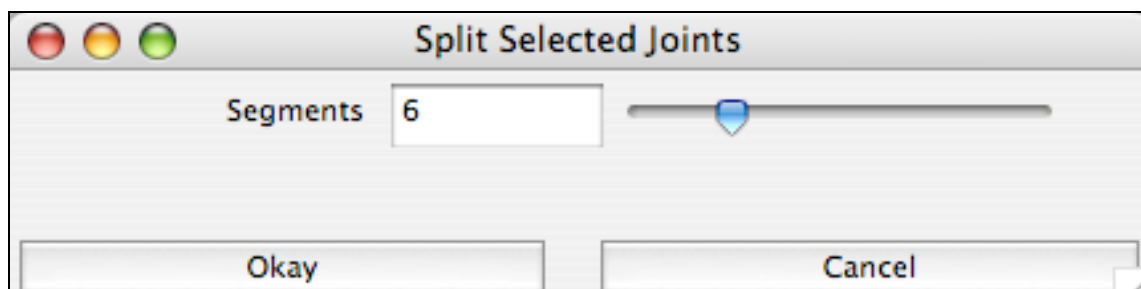


Figure 34 - Split Selected Joints interface

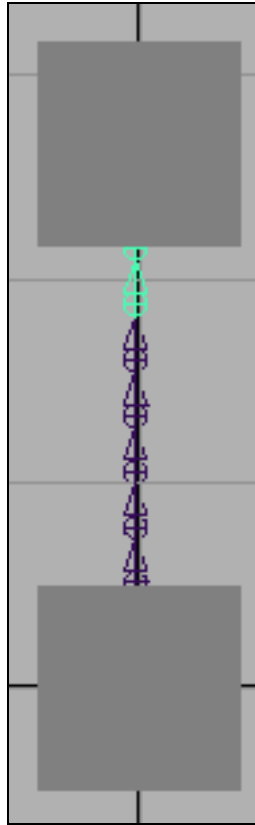


Figure 35 - Joint split into 6 segments

Now that we have our joints broken apart into 6 equal segments, it's going to be much easier to see what is going on if each of them have a piece of geometry under them.

You can create a polygon cube and parent it to each skeleton, or use the **js\_createSkelGeo<sup>vii</sup>** command to do it for you.

#### 16. Assign geometry to each joint

- Select all the joint segments, except for **joint2**

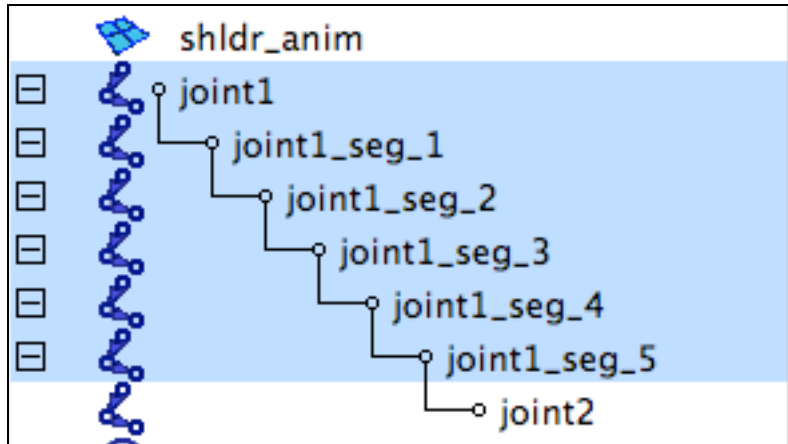
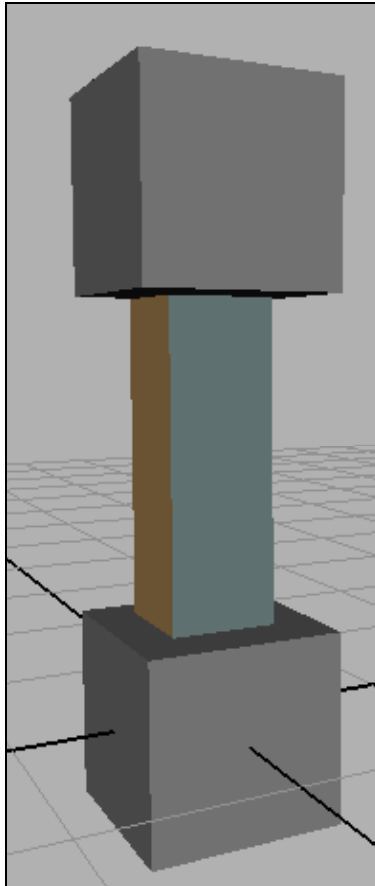


Figure 36- all the joints selected except for the last one.

- Click on the **Add Geometry to the Selected Joint** button in the **Animator Friendly Rigging Shelf**.





*Figure 37 - geometry automatically added*

Notice that the geometry has two shaders on it. It's much easier to see how the skeleton is twisting when this is done, so you can easily tell how things will look for your character.

The next step is to create a **splineIk** control that will go through all of our joints in the back. The **splineIk** takes the joints and snaps them to a **nurbsCurve**. Manipulate the curve, and it will pull the joints around. The great thing about the splineIk control is that you can have absolute control over the twisting of the joints very quickly and easily!

**Allow the joints to bend with the shoulder and hips**

#### **17. Create a splineIk control for the joints**

- Choose **Skeleton > Spline IK Handle Tool > Option Box**
- Set the options to look like the following image:

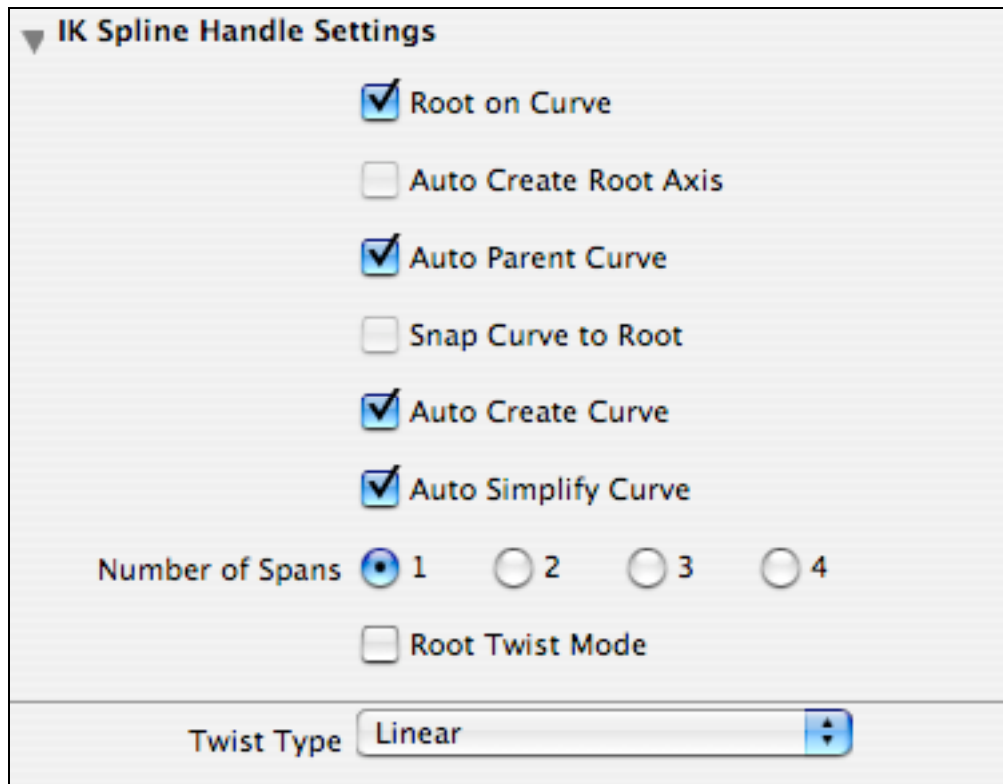


Figure 38 - spline ik handle tool options

- Click on the *first* joint – **joint1**
- Click on the *last* joint – **joint2**

A splineIK Control should be automatically created between these two joints.

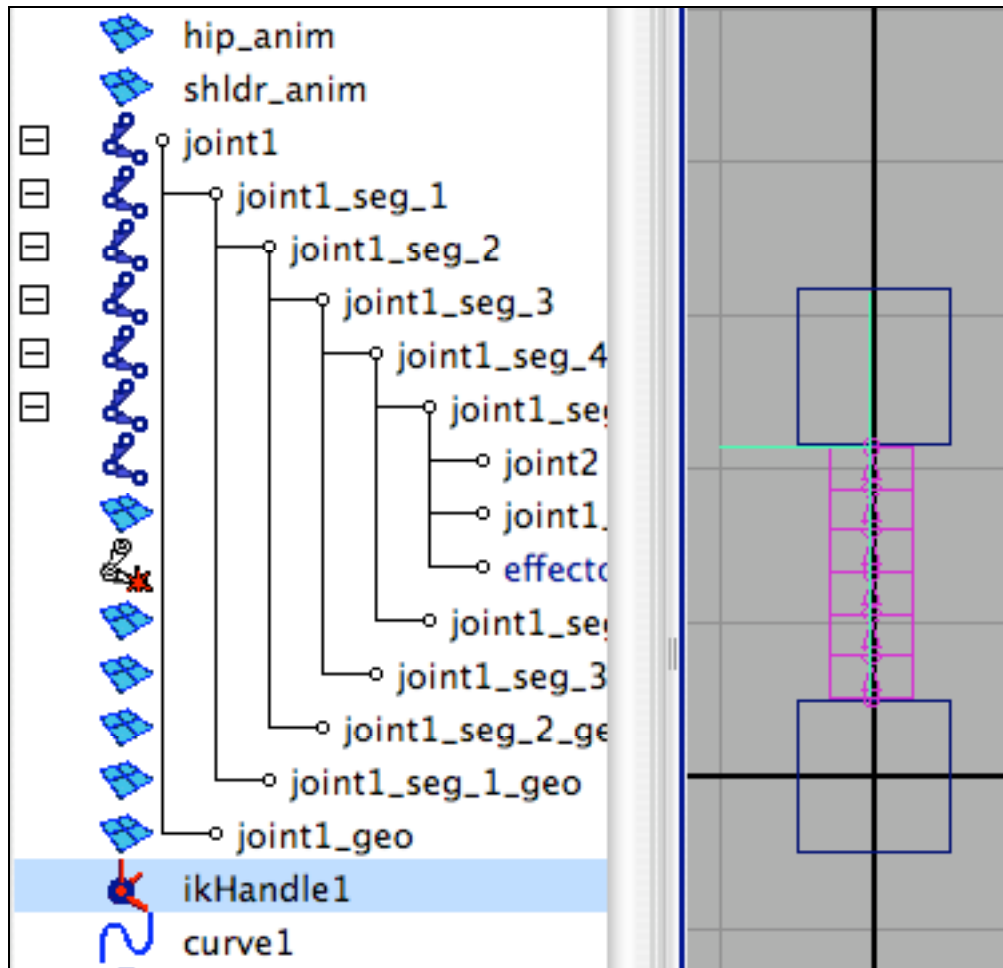


Figure 39 - spline ik control created through all the joints

Notice that there is now an **ikHandle1** and **curve1** node in the scene. The **ikHandle1** node is the spline ik handle. We will find all sorts of attributes on here that we'll be using to manipulate how the spline will work.

The **curve1** node is the nurbs curve that now *controls* the middle of the spine.

#### 18. Manipulate the nurbs curve to see how it works.

- Select **curve1**
- Hit **F8** – this will put you into **Component** mode. It allows you to manipulate the cvs on the curve.



- Click and drag with the **Left Mouse Button** over the top of the spine. When you release, you should have the top CV highlighted.

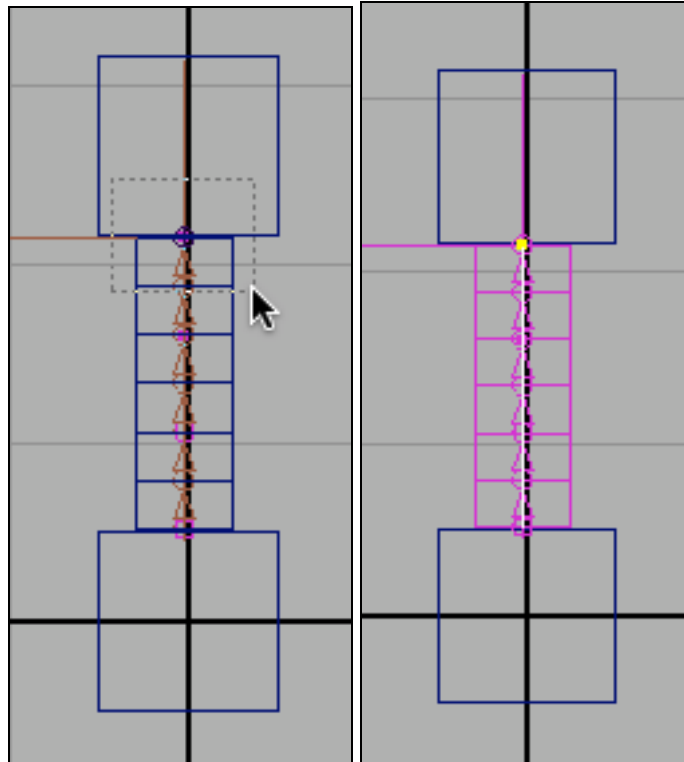


Figure 40- selecting the top cv in the curve (curve1.cv[3])

- Use the **Move Tool** (w) to move the cv around and see how it pulls the curve.

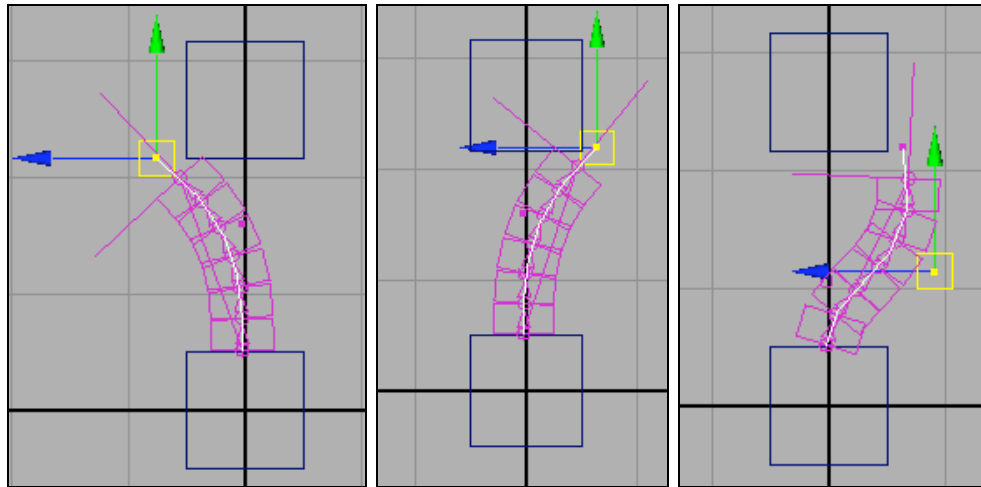


Figure 41 - moving cv's around on curve1 to see how it affects the spine

Now the goal is to actually have the points on the curve move with the hips and shoulder controls.

Again, there are a number of ways to do this using Maya. I'm going to demonstrate a technique using smooth skinning because it's simple, gives good results, and is quick to set up.

To do this, we're going to create a joint for the hip, and a joint for the shoulder. The curve will be smooth-skinned to each joint, causing it to move with them.

#### 19. Create two joints, one for the hip and one for the shoulders

- Choose **Skeleton > Joint Tool**
- Holding down the **v** key for point snap, click with the **Left Mouse Button** near the start joint on top of the hip, and drag *towards* the joint. This will create it and then snap it on the joint.

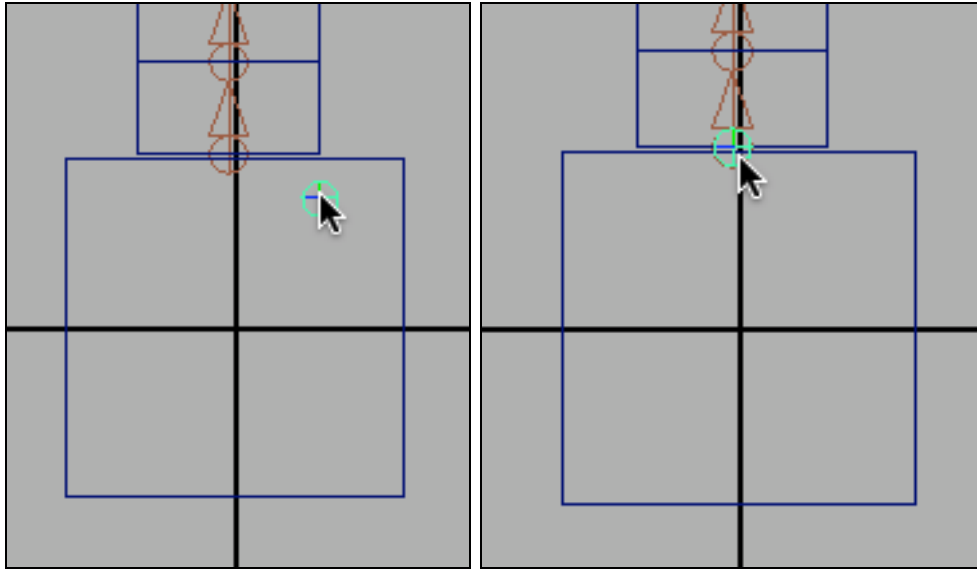


Figure 42 - creating a joint at the top of hip\_anim

- Hit the **y** hotkey. This will complete the tool, and put you back in the joint tool again for the next joint.
- Holding down the **v** hotkey for point snap, click with the **Left Mouse Button** near the end joint at the bottom of the shoulder, and drag *towards* the joint. This will create the joint and snap it in the same position as the other joint.

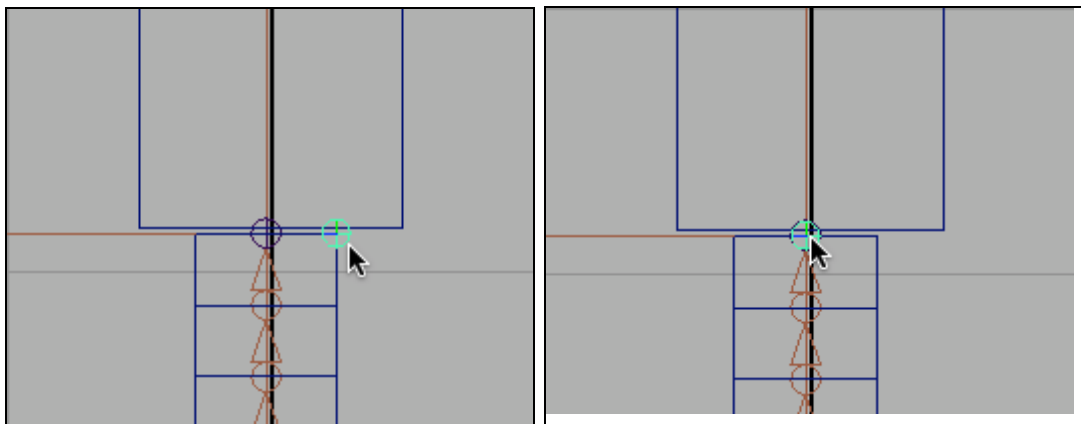


Figure 43 - Snapping the joint to the bottom of the shoulder control.

Now that we have two joints, one for the hips and one for the shoulders, it's a good idea to name them.

#### 20. Change the names of the joints and other items

- Change **joint1** to **back\_joint**
- Change **ikHandle1** to **back\_ikHandle**
- Change **curve1** to **back\_crv**
- Change **joint3** to **hip\_joint**
- Change **joint4** to **shldr\_joint**



Figure 44 - Renaming nodes so they make more sense

#### 21. Skin the curve to the joints

- Select **back\_crv**, **hip\_joint**, and **shldr\_joint**
- Choose **Skin > Bind Skin > Smooth Bind > Option Box**
- Make sure the settings match what is in the following image

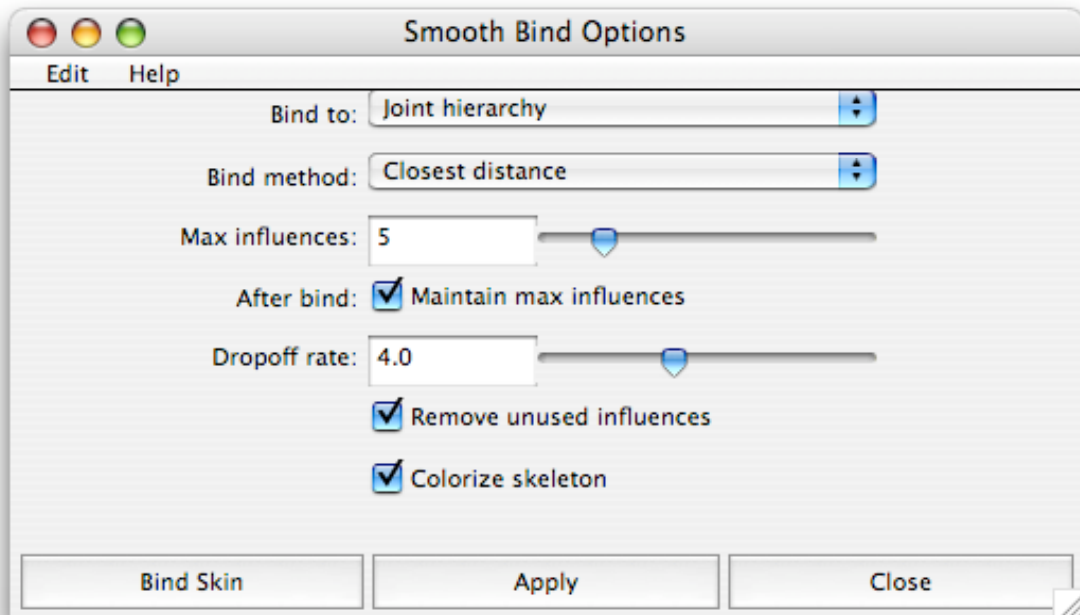


Figure 45 - smooth bind default options

- Hit **Bind Skin**

## 22. Parent the joints to the hips and shoulders

- In the **Outliner**, drag **hip\_joint** onto **hip\_anim** with the **Middle Mouse Button**
- Drag **shldr\_joint** onto **shldr\_anim** with the **Middle Mouse Button**

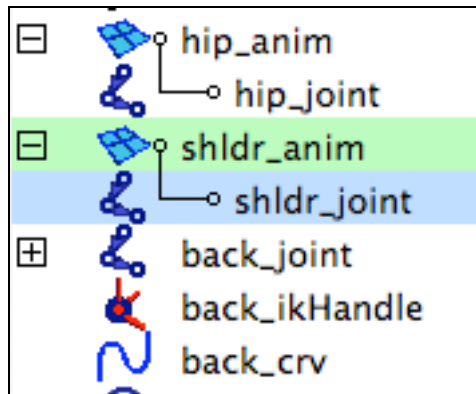


Figure 46- parenting the hip and shoulder joints to the hip and shoulder controls.

Now when you manipulate hip\_anim or shldr\_anim, you should see the back moving to match the position.

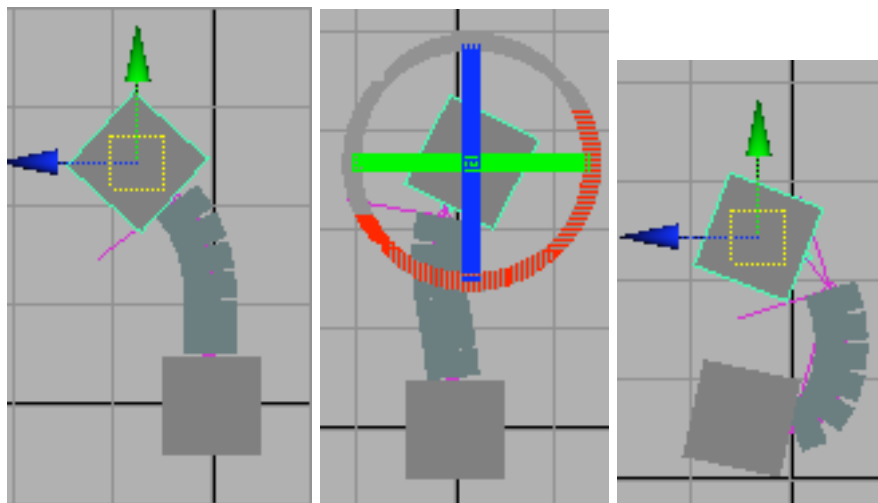


Figure 47 - moving the hips and shoulders

One thing you'll notice right away is that the back doesn't stretch to keep the same length as the shoulder. This doesn't give us the result we want, because we actually *want* the back to stretch in order to meet the hips and shoulders correctly.

I know what you're thinking: "In the real world, backs don't stretch! People aren't made of elastic material!"



## Master Classes

# Instructor Notes – Animator Friendly Rigging Part II

Jason Schleifer

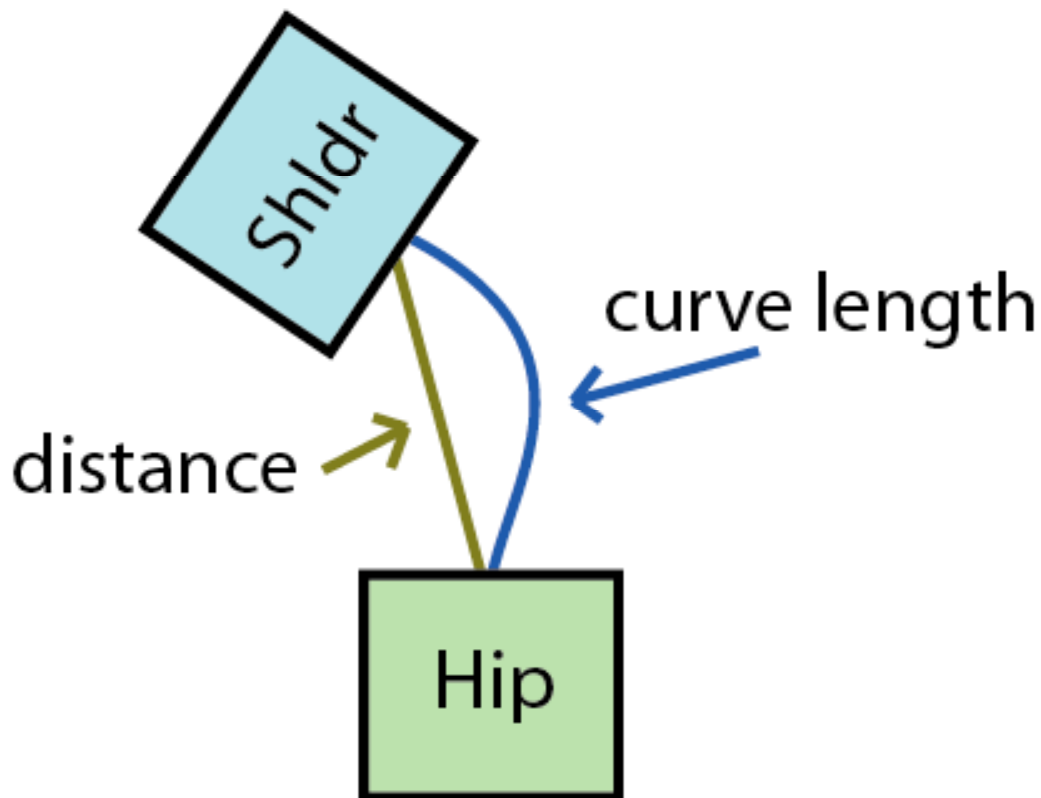
That's true. But we're not in the real world, we're in *animation*-land. Part of an animator's *job* is to keep their character on-model. This means that they're not going to stretch the character beyond any recognizable shape without good reason. It's *more* important to provide them with a control system that's easy to work with and doesn't cause them to fight the system.

If you don't allow the back to stretch, each time the animator adjusts the hips, they'll have to adjust the shoulders, and then they'll have to adjust the hips, and then the shoulders, and then the hips.. you get the idea. Painful!

Instead, let's make it easy for them. If they're going to grab the shoulders, let them move the shoulders *exactly where they want them*.

### Determining the scale

So how do we go about determining how to scale the joints? It's not just a matter of measuring how far the two controls are from each other, because that wouldn't take into account the curve of the spline.



*Figure 48 - the difference between the distance and the curve length*

To determine the length of the curve, we need to measure the **arc length** of the curve. We can grab that information with a node called **curveInfo**, which can be connected to the curve. Then, at any point we can query the **arcLength** attribute on the **curveInfo** node.

The quickest way to attach the **curveInfo** node to the curve itself, is to use the **arclen** command.

#### 23. Connect the curveInfo node to the back curve

- Select **back\_crv**



- Type the following command into the script editor:

```
arclen -ch 1;
```

If you look at the Hypergraph with the curve selected, you can see the **curveInfo** node clearly visible.

#### 24. Open the hypergraph

- Select **back\_crv**
- Choose **Window > Hypergraph Input and Output Connections**

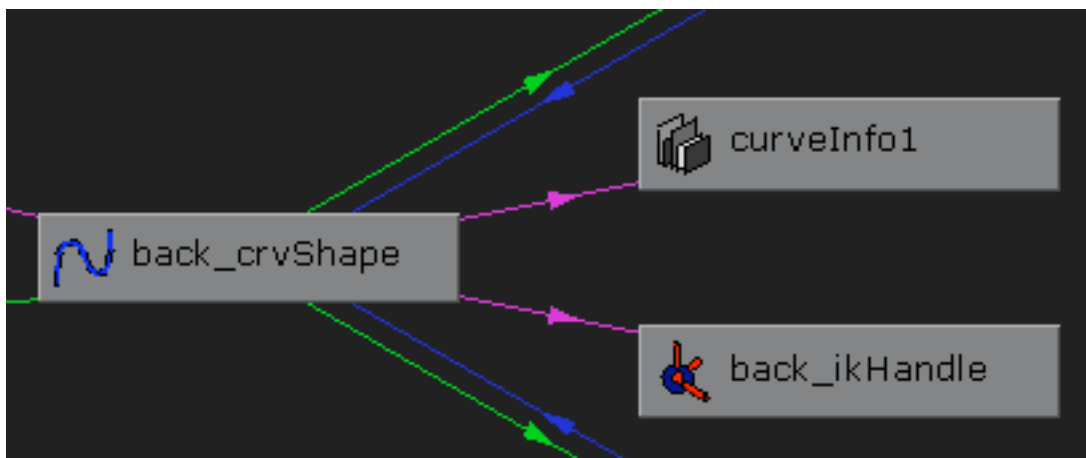


Figure 49 - curveInfo1 connected to the back\_crvShape

#### 25. Get the curve length

- Select **curveInfo1**
- Open the **Attribute Editor**
- Look at the **Arc Length** attribute.

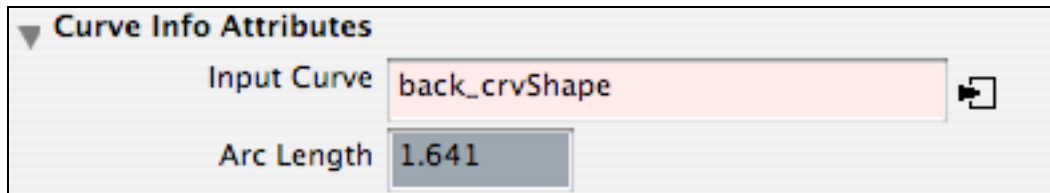


Figure 50 - Arc Length

If you move the shldr\_anim and hips\_anim controls, you'll notice that the Arc Length changes. It increases in value as the curve stretches, decreases as it shrinks.

So how do we use this information to scale the joints on the back properly? This is where the exciting world of math comes into play!

But don't worry.. it's easy math. I promise.

If you look at one of the joints that we want to scale, you'll notice that its scale value is **1**. If the back curve is at its default length, as it is when we don't move the controls, we want to *keep* the joints at a value of **1**. If we *double* the length of the curve, then we want to set the joints to a value of **2**. If we *halve* it, then the joints should have a scale of **.5**.

So in order to know what to do correctly, we want to measure the *difference* between what the curve *originally* was, vs what it *currently* is.

In effect, our expression for the joint's scale will look something like:

**Joint Scale = The amount that the curve scales.**

Let's pretend our initial curve length is a value of 2. And we want to set the scale of the curve to 1.

**1 (joint scale) = 2 (curve length) (and ...)**

Well, if the Joint Scale is 1.. and the curve length is 2.. what can we do with the curve length to give us a value of 1?

If you remember your fractions, you'll know that *any number divided by itself is equal to 1*.



## Master Classes

# Instructor Notes – Animator Friendly Rigging Part II

Jason Schleifer

We could say:

$$1 = 2 / 2$$

or

$$\text{Joint scale} = \text{curve length} / \text{curve length}$$

So that will work fine.. but what if the curve is scaled to twice it's normal length, so it's at a value of **4**? How would we get the joint scale to be a value of **2**?

$$2 = 4/4$$

or

$$\text{Joint scale} = \text{curve length} / \text{curve length}$$

Well, this doesn't work.. because **4/4** is *not* 2, it's **1**. And as we all know, 2 does *not* equal 1. So let's change our expression:

$$\text{Joint scale} = \text{current curve length} / \text{original curve length}$$

That would work, right?

$$2 (\text{joint scale}) = 4 (\text{current curve length}) / 2 (\text{original curve length})$$

What about if we halve the curve length? So instead of it being a value of 4 (double the original size), it's a value of 1 (half the size).

$$.5 = 1 / 2$$

That definitely works! See? Isn't math *fun*?

Now that we've figured out all this high level math, what is the best way to implement it?

I like to follow this rule:

### Expressions for testing Nodes for finessing

In fact, you could make a t-shirt out of it (but I get 10%). What it means is, when I'm testing out ideas for rigs, I like to use **Expressions** because they're easy to tweak, and very easy to read and understand what's going on. In fact, with



## Master Classes

# Instructor Notes – Animator Friendly Rigging Part II

Jason Schleifer

commenting it's *super* easy to know what you're doing because you can *write it in English* (or Spanish, French, arabic.. whatever floats your lingual boat).

Once you know what your planning on doing, you can then convert the expression on to **node** evaluations, because in general they evaluate *faster* than expressions. However, this isn't always the case, so it's important to try both methods and see which is faster for evaluation.

Here's a comparison between the two methods with the example we were just figuring out:

### Expression:

```
// Curve Scale Expression
//
float $scale = curveInfo1.arcLength / 1.641;
back_1_joint.sx = $scale;
back_2_joint.sx = $scale;
back_3_joint.sx = $scale;
back_4_joint.sx = $scale;
back_5_joint.sx = $scale;
back_6_joint.sx = $scale;
```

### Nodes:

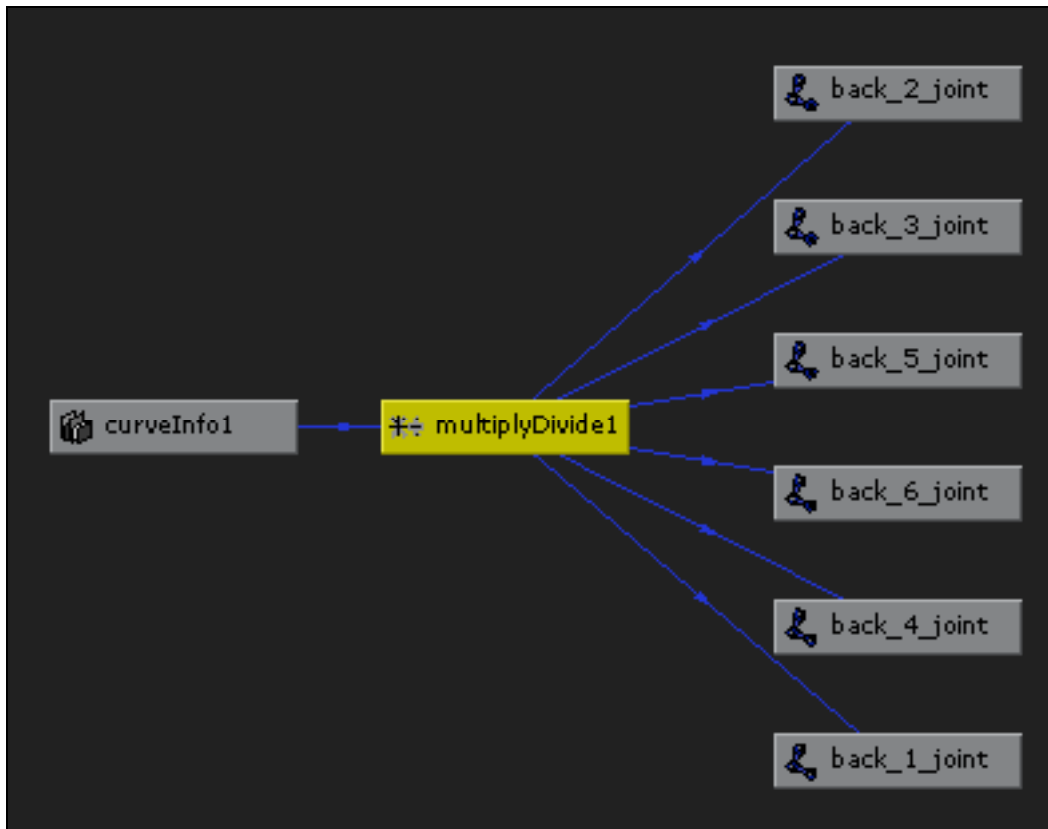


Figure 51 - Node implementation of scale

See how that works? The information from the curveInfo node, feeds into the multiplyDivide node, which sends the results into the back joints. It's a little tougher to read what's going on, and this is a *simple* graph. Imagine a complicated one with a few dozen multiply's, divides, and other functions all blending with each other. It can get messy very, very quickly.

As for speed.. well, the node based implementation here runs only slightly faster than the expression. So is it worth it? Let's find out when we're finished implementing our full back rig.

Let's implement this scale function in order to modify the length of the joints.

## 26. Rename the curveInfo node

- Select **curveInfo1**
- Rename it to **back\_curveInfo**

#### 27. Rename the back joints

- Change the name of each of the back segments so they're named something which is easier to keep track of. **back\_#\_joint** works just fine.

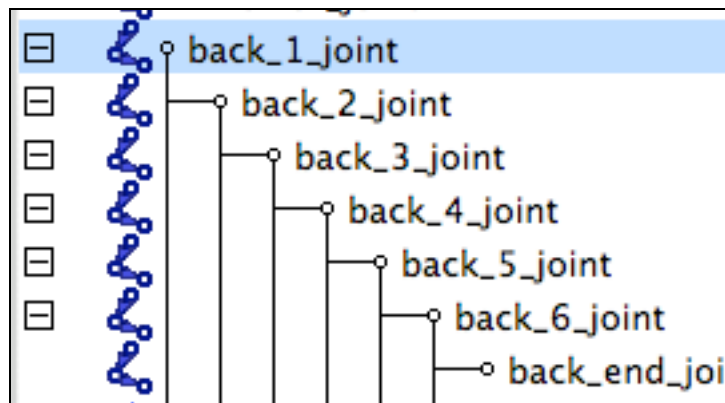


Figure 52 - renaming the joints

#### 28. Get the length of the curve

- In the script editor, type the following command:

```
getAttr back_curveInfo.arcLength;
```

You should get a result which looks something like:

```
// Result:1.641501//
```

Save this value, because it's the actual length of the curve.

#### 29. Create the expression

- Choose **Window > Animation Editors > Expression Editor**
- In the expression text field, enter the following expression

```
// Back Scaling Expression
```



## Master Classes

### Instructor Notes – Animator Friendly Rigging Part II

Jason Schleifer

```
//  
  
// Determine the scale of the back  
// The original length of the back is: 1.641501  
$scale = back_curveInfo.arcLength / 1.641501;  
  
// Now apply that scale to the scaleX attributes  
// on the joints.  ScaleX is the attribute  
// that points down the curve.  
back_1_joint.scaleX = $scale;  
back_2_joint.scaleX = $scale;  
back_3_joint.scaleX = $scale;  
back_4_joint.scaleX = $scale;  
back_5_joint.scaleX = $scale;  
back_6_joint.scaleX = $scale;
```

- Click **Edit**

Move the hips and shoulder controls around. You should see the joints stretching to keep the back the right length.

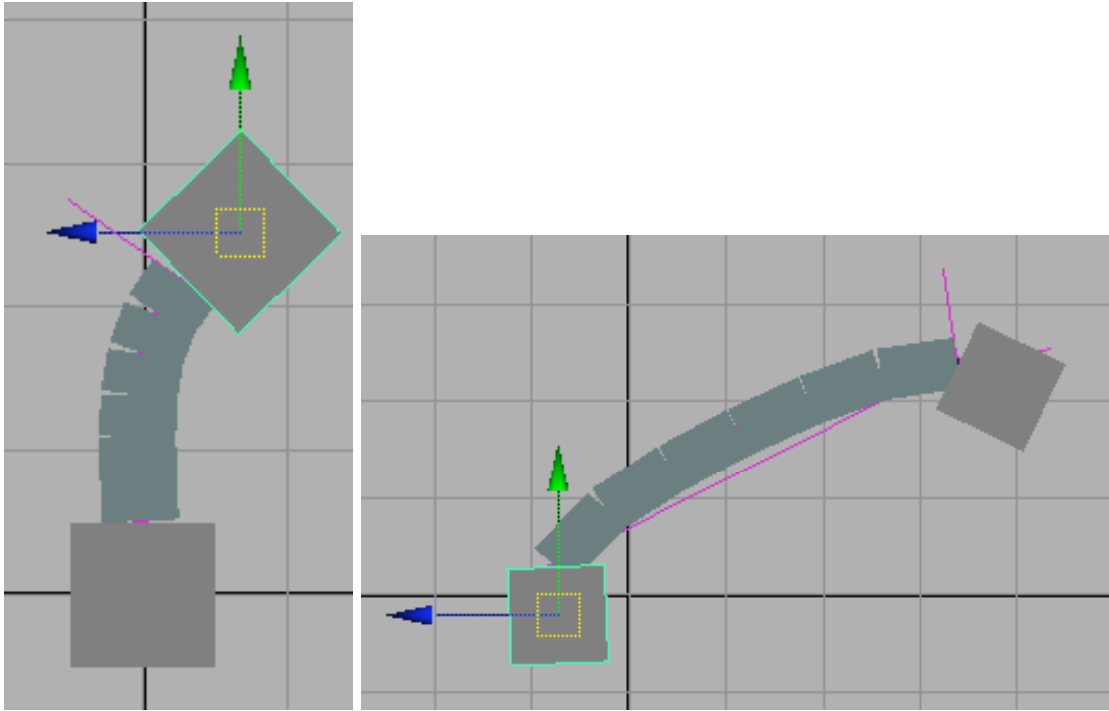


Figure 53 - back stretching

#### Allow for twisting

If you were to switch to a perspective view now, you'd notice that when you *twist* the controls in the **Y-axis**, nothing happens. The back doesn't twist with them! (Figure 54)



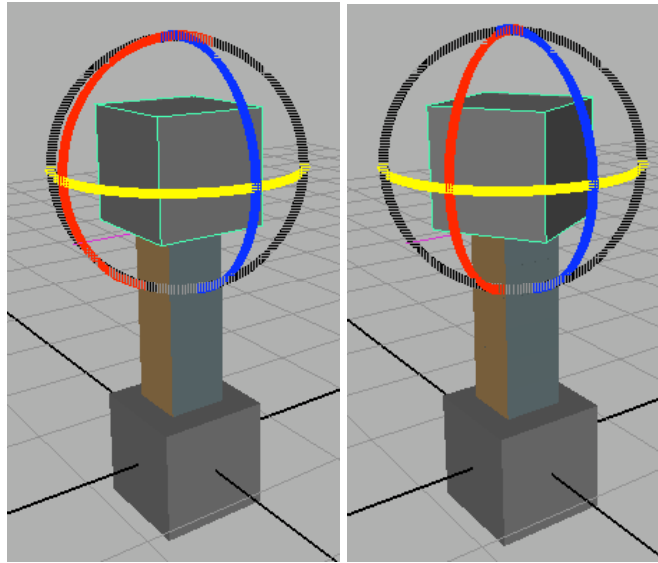


Figure 54 - spine doesn't twist with the hips and shoulders

What?? A back control that won't let the character *twist*? Why.. that's just pure insanity! We can't allow for something like that! And if you hand this off to your animator, why.. I can't even say what would happen. Let's just say it would be nasty, bloody, and you wouldn't want to write home about it.

The next step in our quest to a solid back rig is to hook up the twisting. To do that, it's important to understand how twisting works with a splinelk system.

### 30. Look at Twist and Roll

- Select **back\_ikHandle**
- In the **Channel Box** adjust **Twist** and the **Roll** attributes.

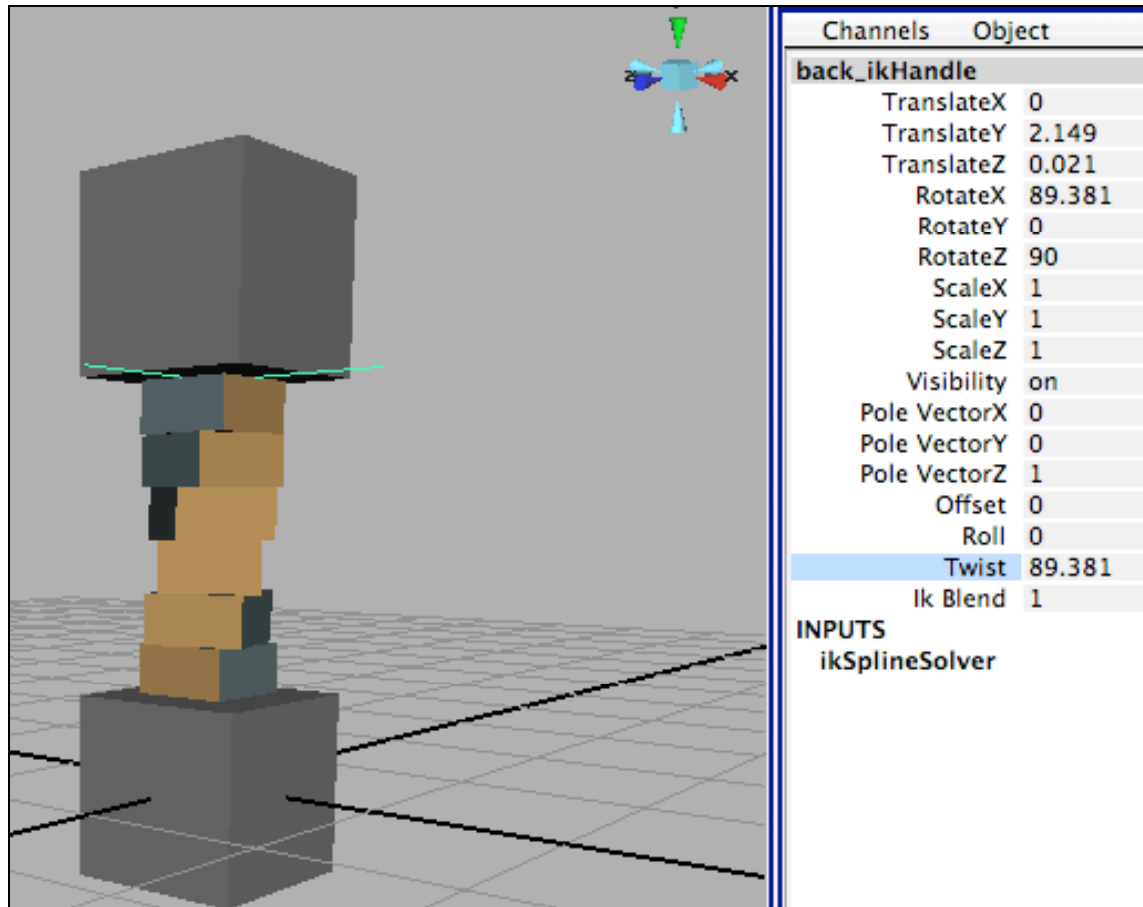


Figure 55 - adjusting the twist attribute on back\_ikHandle

Notice what happens with the back as you use the **twist** attribute. It does exactly what you'd think.. it **twists** the back. Each joint rotates just a tiny bit, causing the back to twist about.

When you adjust the **roll** attribute, it takes all the joints and just **rolls** them around the spine. It's sort of like a global offset to the rotation of the joints.

While these are very interesting controls, they're not the best controls we can have for managing the twisting of the back. As long as the animator is going to be rotating the hips and shoulders, we should just tell the back to match the twisting *to them*.

This way, the animator never has to think about the twisting of the spine. It just happens naturally, exactly as they would expect it to.



Master Classes

## Instructor Notes – Animator Friendly Rigging Part II

Jason Schleifer

### Using the Advanced Twist Controls

The **Advanced Twist Controls** on the splinelk handle will allow you to attach the twisting to a start and end control, and specify exactly how they will affect the joints. They're a bit confusing when you first look at them, so let's go through really quickly how to set them up to work with your back rig.

#### 31. Bring up the Advanced Twist Controls in the Attribute Editor

- Select **back\_ikHandle**
- Open the **Attribute Editor**
- Scroll down until you see **Ik Solver Attributes**
- Open Ik Solver Attributes, and scroll down until you see **Advanced Twist Controls**.
- Open Advanced Twist Controls.

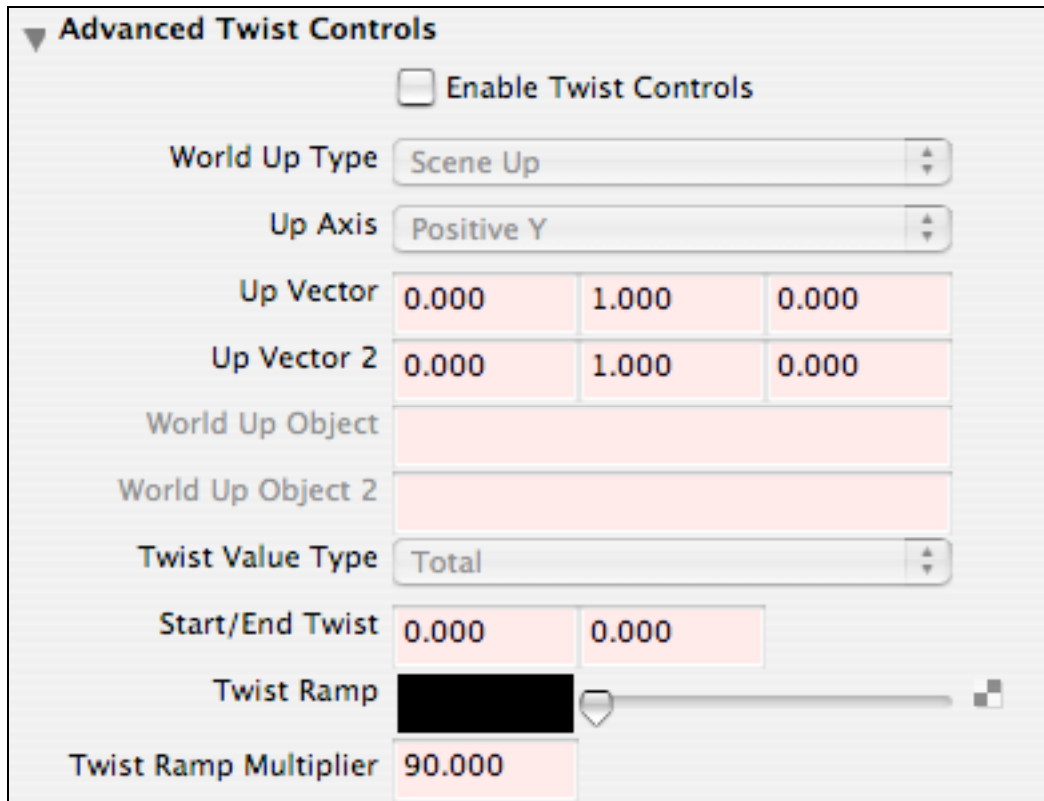


Figure 56- Advanced Twist Controls for the spline IK Solver.

As you can see, there are a number of options that at first appear very confusing and overwhelming. Don't worry, there are really only a few things we need to worry about:

- **World Up Type** – This attribute will let us tell the splineIK to follow the hips and shoulder controls
- **Up Axis** – The axis on the controls that we consider “up”
- **Up Vector** – What direction the START of the spine considers up
- **Up Vector2** – What direction the END of the spine considers up
- **World Up Object** – The start control (the hips)
- **World Up Object2** – The end control (the shoulders)

### 32.Enable Twist Controls

- Click **Enable Twist Controls** – this will enable the advanced twist controls on the spine.

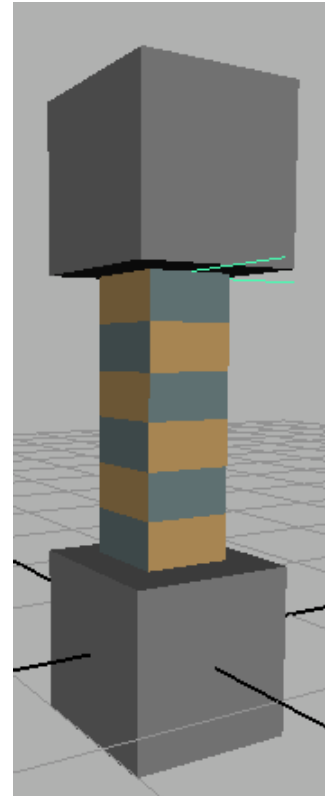
You will notice that as soon as the advanced twist controls are enabled, the spine twists all uncontrollably. This is because we haven't told the spine how we *want* it to twist yet!

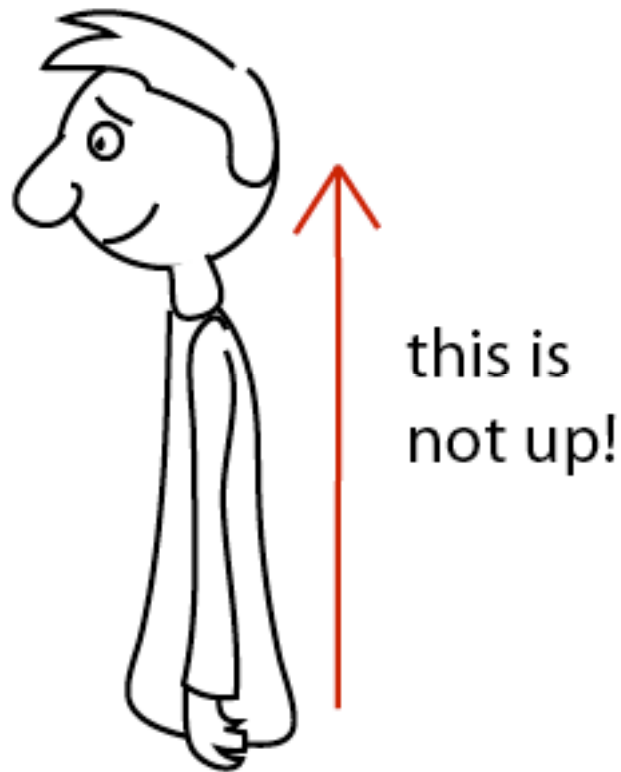
What we do now is step through the attributes and set them correctly so the spine doesn't twist unexpectedly.

#### 33. Set the World Up Type

This attribute allows us to specify *what* the spine uses for controlling the direction it considers *up*.

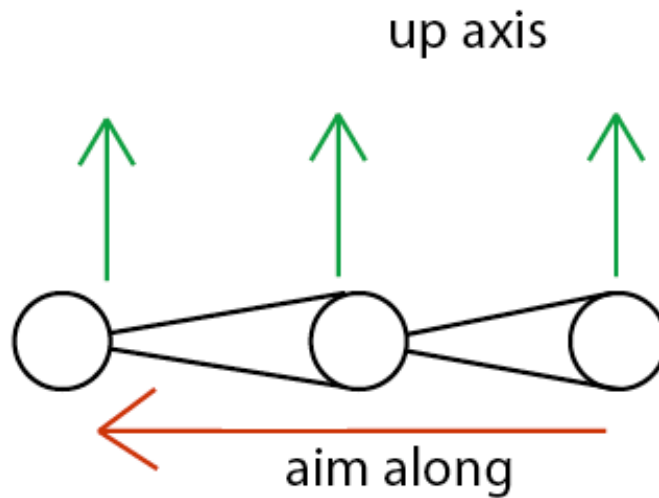
Up does *not necessarily* refer to the actual “up” direction we would assume. With a back, we would think “up” meant straight up, or up towards the head.





*Figure 57 - up does not refer to the world space "up" for our scene.*

That is not the way the “up” works on a splineIK system. For the splineIK, “up” is the direction we want the joints to angle as they’re aiming along the spine.



*Figure 58 - the "up" is the axis that keeps the joints oriented the same as the aim along the curve*

If you were to rotate that image, so the joints would go straight up and down like a true spine.. you would be able to see that the “up axis” is actually pointing back *behind* the joints.

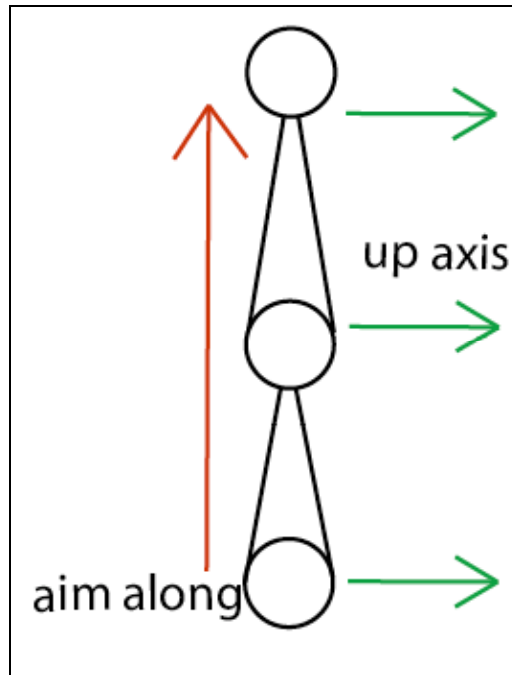


Figure 59 - re-orienting the joints to make more sense for a "spine"

*This* is what we want to be thinking about when we determine what is “up”. We want to determine *how we specify the axis that the joints will aim at while they’re moving with the curve.*

So what did we decide we wanted to determine our up axis? The twisting of the shoulders and hips, of course!

- Set **World Up Type** to **Object Rotation Up [Start|End]**

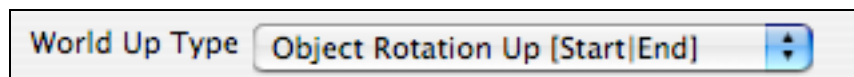


Figure 60 - World Up type set to Object Rotation Up [ Start| End]



#### 34. Set the Two Objects

It doesn't do us much good to have world up type specified as an object type without telling it which objects to use!

We're going to be using the hips and shoulders to control the up vector twisting, so let's specify them. The first object is the one at the *start* of the spline, and the second object is the one at the *end* of the spline. Since we drew the spine from the hips to the shoulders, hips\_anim is the first object, shldr\_anim is the second.

- Set **World Up Object** to **hip\_anim**
- Set **World Up Object2** to **Shldr\_anim**



Figure 61 - Setting the world up object 1 and 2

#### 35. Set the Up Axis

Now that we've determined that the hips and shoulders are going to control the up axis, we have to tell it *what axis* is actually going to control the up axis.

If you look at the following image, you can see that we're specifying the *back* of the character as "up". Now look at the world axis. Which direction is the same as the "up" we previously specified?

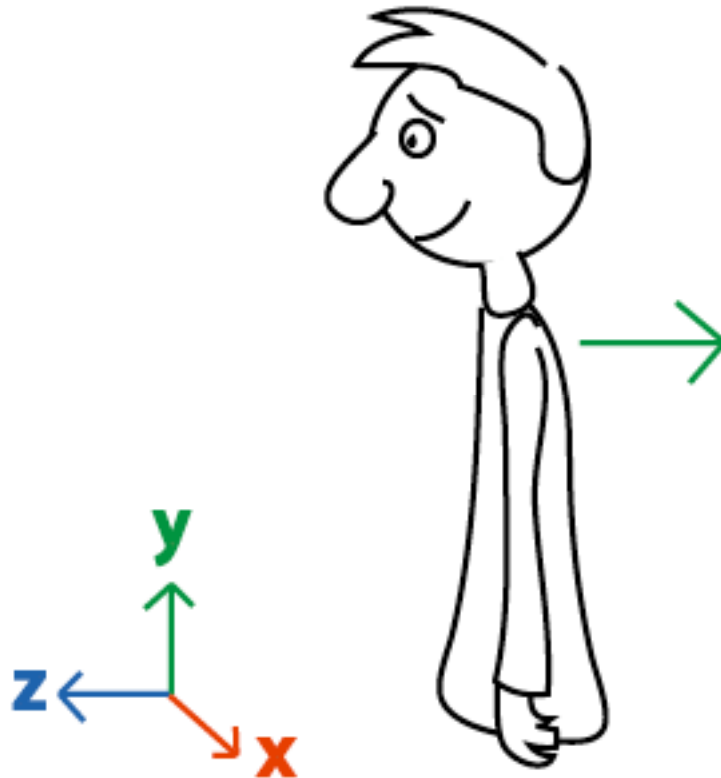


Figure 62 - up and the world axis

Exactly.. *negative* z!

- Set **Up Axis** to **Negative Z**

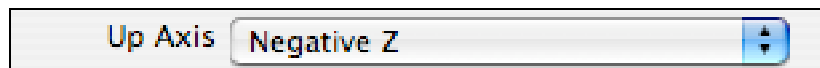


Figure 63 - up axis set to negative z

### 36. Setting the Up Vectors

The final step requires you to tell the joints exactly what their up vector is. Now this seems very confusing as well.. you've got three values for each vector (X, Y, and Z), and you can set them to almost anything you want. The

reason for this is that you may want to specify a vector that's a *combination* of X and Y.. or Z and X. Personally, I never do this because I always try and set up my joints so they aim down an axis which is easy to work with when using splineIK.. and in the case of our back, we can simply do this as well!

The easiest way to see which direction we want to tell the joints to orient is to look at their rotation axis.

- Select **back\_1\_joint**
- Choose **Display > Transform Display > Local Rotation Axis**

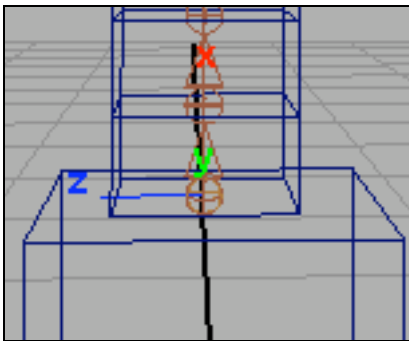


Figure 64 - Local rotation axis turned on for back\_1\_joint

As you can see, the rotation axis displays that the X axis is aiming up the joint, the Z axis is facing the front of the character, and the Y axis is going off to the side.

So if we want to keep the joint at this orientation, then we want to use the *negative z* axis be the one that's used for the up vector!

- Set **Up Vector** to **0 0 -1**
- Set **Up Vector2** to **0 0 -1**
- Select **back\_1\_joint** and go **Display > Transform Display > Local Rotation Axis** to turn off the display of the axis

Up Vector	0.000	0.000	-1.000
Up Vector 2	0.000	0.000	-1.000

*Figure 65 - Up Vector set for the aim axis*

Now when you rotate the controls, you'll notice that the back twists with them! There's only one small problem – we haven't adjusted the rotation orders! If the animator uses this rig, they're going to hit gimbal lock! Let's quickly adjust the rotation orders and make sure they're correct.

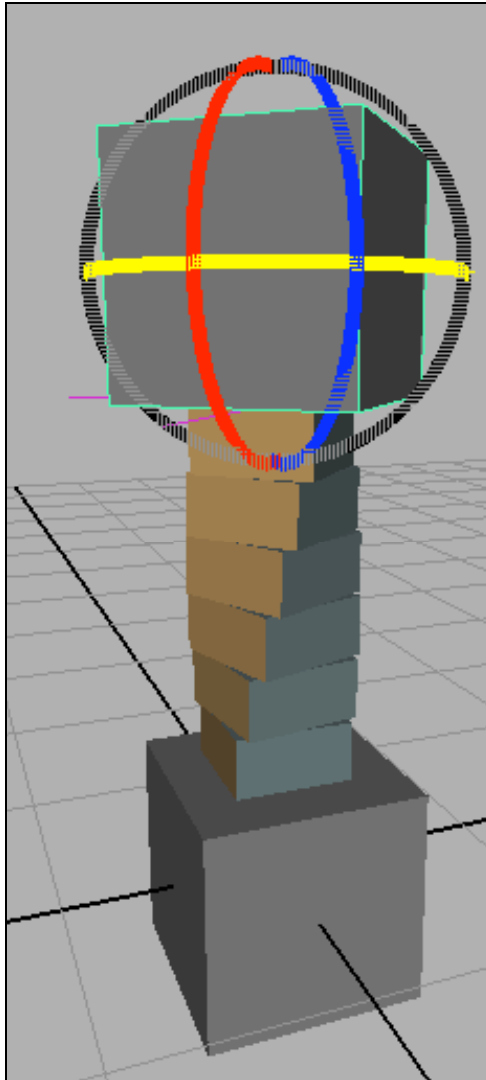


Figure 66 - twist working with the back control

#### 37. Fix the rotation orders on the hips and shoulders

- Select **hip\_anim**
- Open the **Attribute Editor**
- Set the **rotation order** to **zxy**
- Select **shldr\_anim**
- Open the **Attribute Editor**



## Master Classes

# Instructor Notes – Animator Friendly Rigging Part II

Jason Schleifer

- Set the **rotation order** to **zxy**

### Volume Preservation

One thing you may notice as you pull the rig around is that the *width* of the spine never changes as the boxes move further and further apart. This may be exactly what you want to happen, and if so, then that's great! However, I feel that having a little bit of volume deformation will help your characters feel a bit more "grounded" in reality (even though they can stretch their bodies like taffy).

Let's first try a technique similar to the one we used for the bouncing ball. The expression for the volume preservation there looked something like:

```
$scaleX = 1/sqrt($scaleY);  
$scaleZ = 1/sqrt($scaleY);
```

The scale in X and Z is the inverse of the scale in Y.. so as the ball scaled UP in Y, it would scale DOWN in Z and X.

We can easily implement a similar addition to our expression.

### 38. Load the Scale Expression

- Choose **Window > Animation Editors > Expression Editor**
- Choose **Select Filter > By Expression Name**
- Click on **expression1**

Now, expression1 is a really bad name for such an important expression! Let's rename it so we can find it easier later on.

### 39. Rename Expression1

- Click in the Expression Name field, and rename **expression1** to **backScale\_expr**

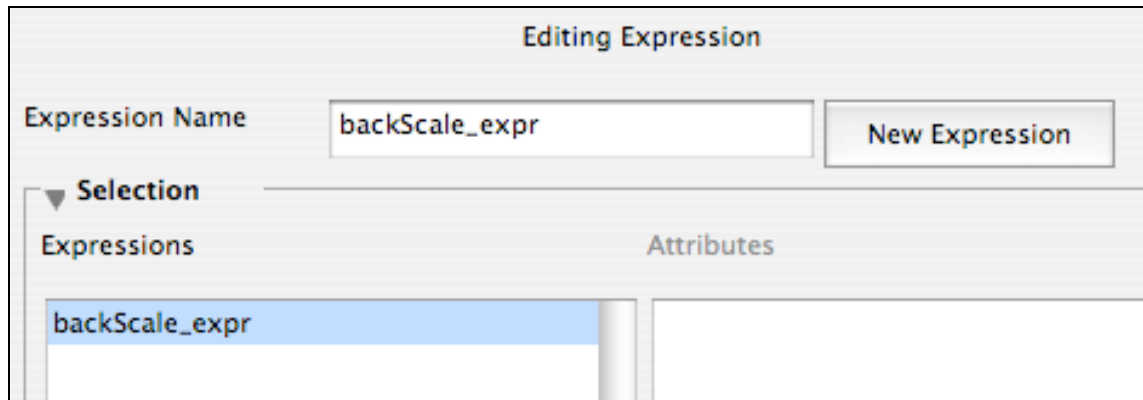


Figure 67 - Back scale expression renamed!

#### 40. Add scaling for Z and Y axis

- Open the Expression Text Area and modify the expression.

```
// Back Scaling Expression
//

// Determine the scale of the back
// The original length of the back is: 1.641501
$scale = back_curveInfo.arcLength / 1.641501;

// get the inverse scale so we can maintain volume
$invScale = 1/sqrt($scale);

// Now apply that scale to the scaleX attributes
// on the joints.  ScaleX is the attribute
// that points down the curve.
back_1_joint.scaleX = $scale;
back_2_joint.scaleX = $scale;
back_3_joint.scaleX = $scale;
back_4_joint.scaleX = $scale;
back_5_joint.scaleX = $scale;
back_6_joint.scaleX = $scale;

// Inverse scale for volume preservation
back_1_joint.scaleY = $invScale;
back_1_joint.scaleZ = $invScale;
back_2_joint.scaleY = $invScale;
```



## Master Classes

### Instructor Notes – Animator Friendly Rigging Part II

Jason Schleifer

```
back_2_joint.scaleZ = $invScale;  
back_3_joint.scaleY = $invScale;  
back_3_joint.scaleZ = $invScale;  
back_4_joint.scaleY = $invScale;  
back_4_joint.scaleZ = $invScale;  
back_5_joint.scaleY = $invScale;  
back_5_joint.scaleZ = $invScale;  
back_6_joint.scaleY = $invScale;  
back_6_joint.scaleZ = $invScale;
```

As you modify the spine, you'll notice it squashing and stretching. This helps preserve volume as the animator works with the rig.



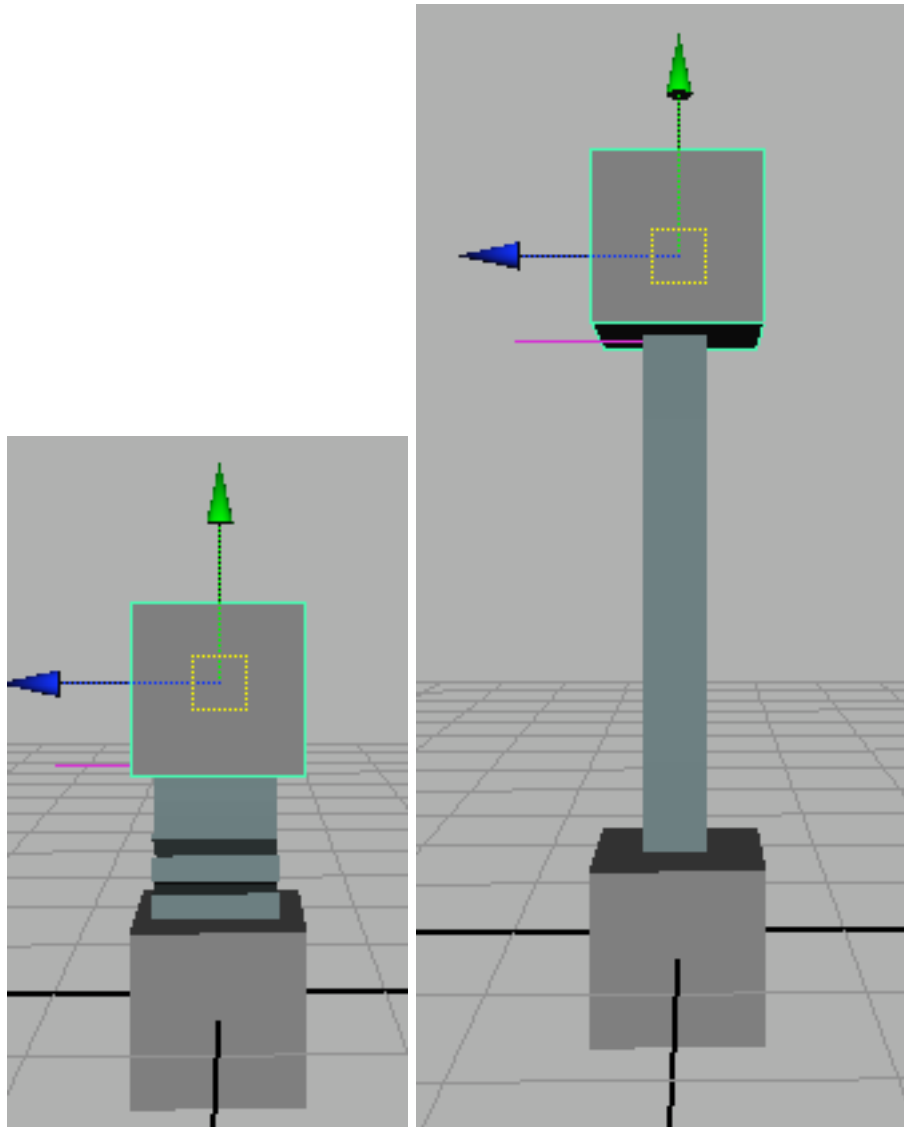


Figure 68 - squashing and stretching

If you look closely, you'll notice that the squashing and stretching is kind of boring. It's all very consistent. Every little part squashes and stretches the same amount. Wouldn't it be nice if we could modify that so the parts in the middle of the spine got thinner as the body stretched apart?

What we want to do is *increase* the amount that those inner parts squash and stretch, more so than the outer parts. If the torso is at it's default state,

Autodesk® Maya® Master Classes – Instructor Notes

SIGGRAPH™ 2006



## Master Classes

# Instructor Notes – Animator Friendly Rigging Part II

Jason Schleifer

everything should be set to 1. But if it's stretched out, we want the middle parts to have a scale of .3, while the other parts have a scale of .5. Not those numbers specifically, but you get the idea.

So how can we increase only some of the values and not others? By using special math skills again! Huzzah!

### Using Pow

We can use Maya's **pow** function to raise the numbers as we need to. For example, if we have a scale of 1 on two objects, and they both are raised to different powers:

$$1^1 \text{ and } 1^2$$

They're both going to be equal to 1. But if the scale increases to 2:

$$2^1 \text{ and } 2^2$$

They're *not* both equal. The first number is equal to **2**, but the second is equal to **4**! We can use this ability to modify our scales so that some get more affect from the scaling than others! For example, we could have something like:

```
back_1_joint.scaleZ = pow($scale,1);  
back_2_joint.scaleZ = pow($scale,1.3);  
back_3_joint.scaleZ = pow($scale,1.5);  
back_4_joint.scaleZ = pow($scale,1.6);  
back_5_joint.scaleZ = pow($scale,1.3);  
back_6_joint.scaleZ = pow($scale,1);
```

See? The joints in the middle will receive *more* of an affect than the joints on the edge. Let's go ahead and implement this into our Expression.

### 41.Add Pow command into the expression

- Choose **Window > Animation Editors > Expression Editor**
- Pick **backScale\_expr**



## Master Classes

### Instructor Notes – Animator Friendly Rigging Part II

Jason Schleifer

- Modify the expression so it looks like the following:

```
// Back Scaling Expression
//

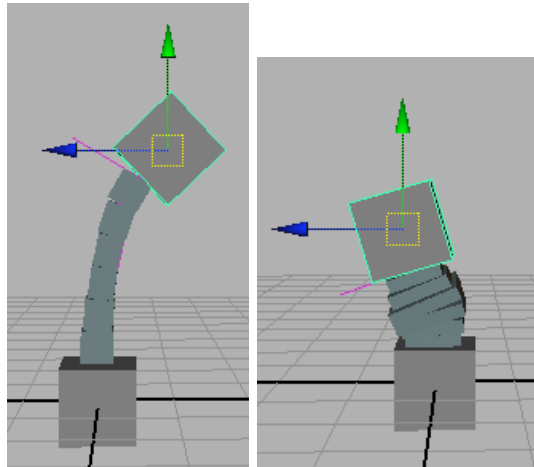
// Determine the scale of the back
// The original length of the back is: 1.641501
$scale = back_curveInfo.arcLength / 1.641501;

// get the inverse scale so we can maintain
// volume
$invScale = 1/sqrt($scale);

// Now apply that scale to the scaleX attributes
// on the joints.  ScaleX is the attribute
// that points down the curve.
back_1_joint.scaleX = $scale;
back_2_joint.scaleX = $scale;
back_3_joint.scaleX = $scale;
back_4_joint.scaleX = $scale;
back_5_joint.scaleX = $scale;
back_6_joint.scaleX = $scale;

// Inverse scale for volume preservation
back_1_joint.scaleY = $invScale;
back_1_joint.scaleZ = $invScale;
back_2_joint.scaleY = pow($invScale,1.5);
back_2_joint.scaleZ = pow($invScale,1.5);
back_3_joint.scaleY = pow($invScale,2);
back_3_joint.scaleZ = pow($invScale,2);
back_4_joint.scaleY = pow($invScale,2.1);
back_4_joint.scaleZ = pow($invScale,2.1);
back_5_joint.scaleY = pow($invScale,1.5);
back_5_joint.scaleZ = pow($invScale,1.5);
back_6_joint.scaleY = $invScale;
back_6_joint.scaleZ = $invScale;
```

Move the shoulders and hips. Notice how the back scales to “keep volume”? Obviously, this isn’t perfect volume preservation, but it does give the effect that we’re going for.

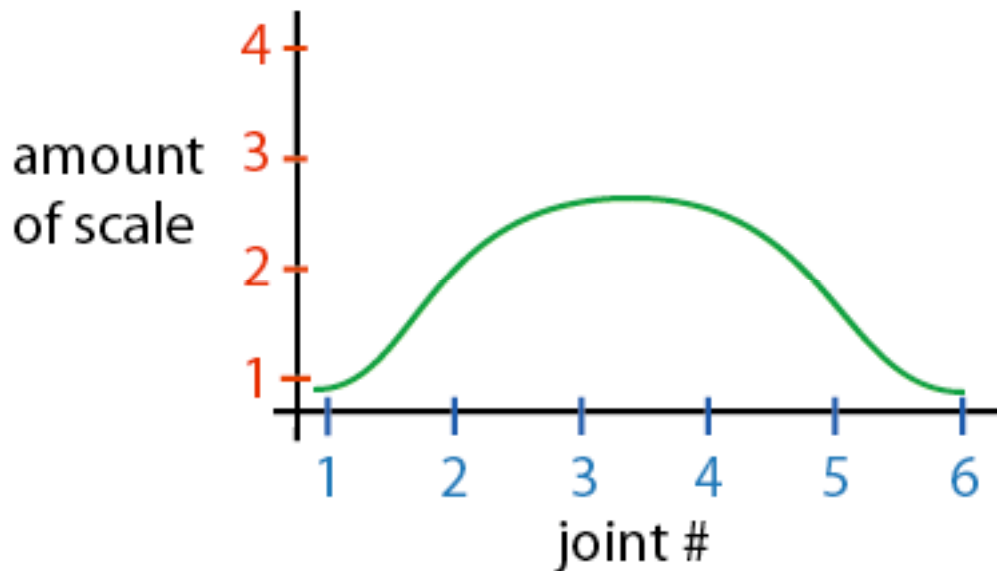


*Figure 69 - volume preservation*

This gives us a quite a bit of power over what our back is doing while we squash and stretch. What if we’re not quite happy with the result? What if we want to tweak the result a bit so we it scales a bit more, or a bit less?

Editing the expression each time can become tedious and frustrating! What would be the ultimate solution to be able to modify something like this?

Personally, I think a graphical way of manipulating the scales for all the objects would be great. Something we could easily select, manipulate, have it change the scales for all the objects, and then be done with it. Since we’re manipulating the shape of the object.. what if we had a curve that we could modify?



*Figure 70 - possible interface for manipulating the amount of scale on the joints*

See how easy that would be to manipulate? Simply modify the curve, and it will modify the joint scale. Luckily, Maya is all about creating animation curves.

#### 42. Add an attribute to the shldr\_anim control for scale

- Select **shldr\_anim**
- Choose **Modify > Add Attribute**
- Name the attribute **backStretch** and make sure it's of type **float**

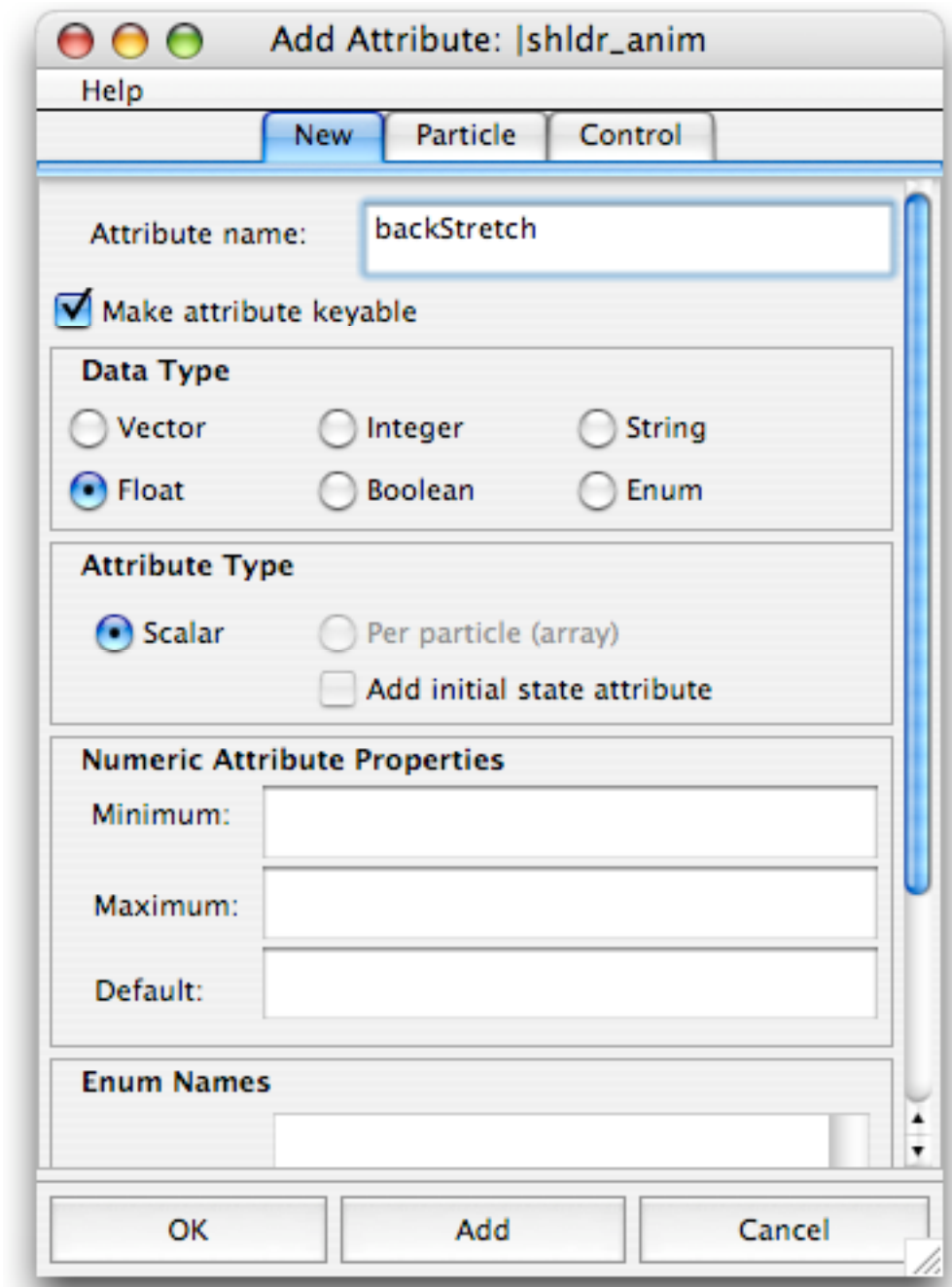


Figure 71 - backStretch attribute

- Click **Ok**.

#### 43. Create the animation curve on backStretch

Now we're going to create an animation curve that will represent the bell curve. We will make the animation on this curve go from 1 to 6 (1 frame for every object we'll be animating).

- Go to frame **1**.
- Save a keyframe for **backStretch** at a value of **1**
- Go to frame **6**
- Save a keyframe for **backStretch** at a value of **1**
- Go to frame **3**
- Save a keyframe for **backStretch** at a value of **2**
- Open the **Graph Editor** (Choose **Window > Animation Editors > Graph Editor**)
- Select all the keys
- Choose **Tangents > Flat**

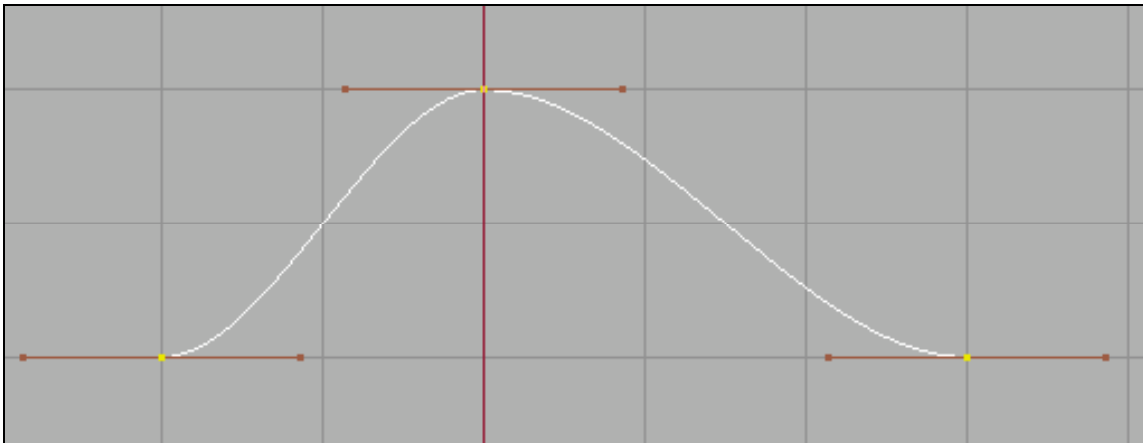


Figure 72 - backStretch with the curve we'll use for modifying the back.

The next step is to grab the value of the curve for each frame. For example, `back_1_joint.sz` should get the value of `shldr_anim.backStretch` at frame 1. `back_3_joint.sz` should be getting the value at frame 3. And so on.



## Master Classes

# Instructor Notes – Animator Friendly Rigging Part II

Jason Schleifer

What's the best way to grab these values? Well, there's a little known node called **frameCache**. Using this node, we can specify exactly what frame we want to grab the value of the curve from!

### 44. Create a frameCache Node and attach it to the curve

- In the script editor, type the following command:

```
createNode frameCache;
```

- Now **rename** the **frameCache** node something which makes more sense, like **back\_1\_cache**;
- Select both **back\_1\_cache** and **shldr\_anim** and graph them in the Hypergraph by choosing **Window > Hypergraph Input and Output Connections**
- Choose **Window > General Editors > Connection Editor**
- Select **back\_1\_cache** and click **Reload Right**
- Select **shldr\_anim** and click **Reload Left**
- On the Left side, choose **backstretch**
- On the Right side, click **Stream**



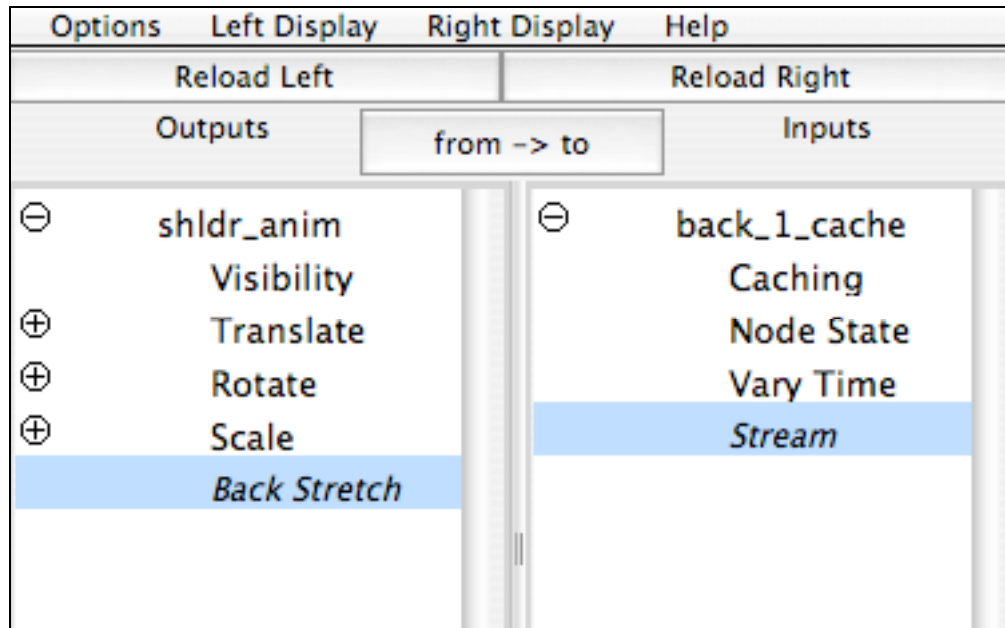


Figure 73 - backStretch connected to the frameCache node

Now that the frameCache node is connected, we need to tell it what frame to connect to. That's done with the **varyTime** attribute.

#### 45. Set the frame for the frameCache

- Select **back\_1\_cache**
- Open the **Attribute Editor**
- Set **Vary Time** to 1.

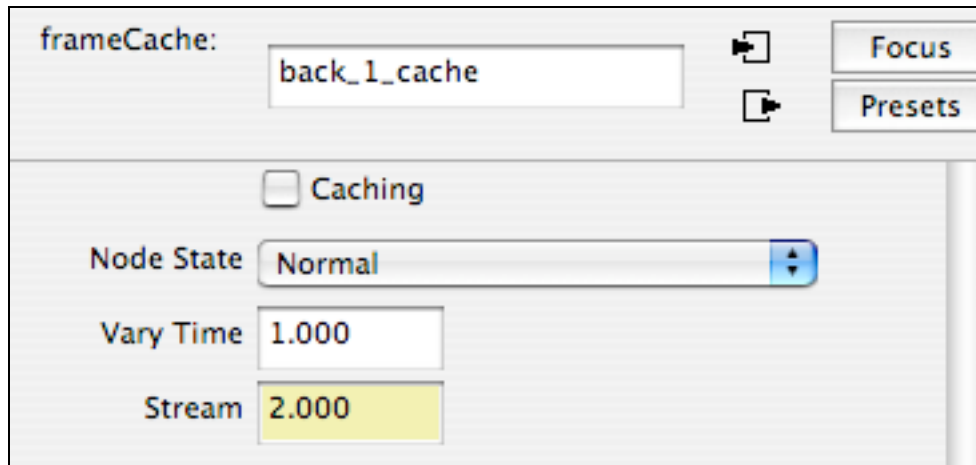


Figure 74 - Setting the frame we'll be grabbing to frame 1.

From now on, if we query the `back_1_cache.varying` attribute, it will tell us exactly what the value of the curve is at frame 1 for the `backStretch` attribute. So let's add that into the expression and see what it does!

#### 46. Adding the `backScale` value into the expression

- Choose **Window > Animation Editors > Expression Editor**
- Choose **`backScale_expr`**
- Modify the following line in the expression from:

```
back_1_joint.scaleY = $invScale;
```

to

```
back_1_joint.scaleY = pow($invScale,back_1_cache.varying);
```

Test this and see what happens. If you drag the shoulder control now, you should see no difference. However, if you leave the shoulder stretched out,



## Master Classes

# Instructor Notes – Animator Friendly Rigging Part II

Jason Schleifer

grab the curve in the graph editor for backStretch and modify *that*, you should see the first joint scaling based on how the curve is modified.

Pretty exciting, isn't it!

We should now add this onto all the joints! Except, adding it can be quite tedious.. which is why you can use the **js\_createCurveControl.mel**<sup>viii</sup> script to do it automatically!

This script will allow you to add an attribute onto an existing object that will drive a whole bunch of objects with the curve. It automatically creates the attributes if they don't already exist, and sets up the curve with a nice bell shaped curve by default.

Let's use it on our back rig. What we're going to do is use the backStretch attribute we've already created, but then have it add a "pow" attribute to each of the joints. That "pow" attribute will be affected by the curve, which will then drive the scale in the expression.

### 47. Remove the previous frameCache node and animation curve

- Select **back\_1\_cache**
- Delete it.
- Select the animation curve for **shldr\_anim.backStretch**
- Delete the curve.

### 48. Use js\_createCurveControl.mel to create the control curve

- Select the back joints (**back\_1\_joint, back\_2\_joint, back\_3\_joint, back\_4\_joint, back\_5\_joint, back\_6\_joint**)
- Type the following command into the script editor:

```
js_createCurveControl shldr_anim backStretch pow;
```

### 49. Modify the expression to use the pow attribute

- Choose **Window > Animation Editors > Expression Editor**

Autodesk® Maya® Master Classes – Instructor Notes

SIGGRAPH™ 2006



## Master Classes

### Instructor Notes – Animator Friendly Rigging Part II

Jason Schleifer

- Choose **backScale\_expr**
- Change the expression so it looks like the following:

```
// Back Scaling Expression
//

// Determine the scale of the back
// The original length of the back is: 1.641501
$scale = back_curveInfo.arcLength / 1.641501;

// get the inverse scale so we can maintain volume
$invScale = 1/sqrt($scale);

// Now apply that scale to the scaleX attributes
// on the joints.  ScaleX is the attribute
// that points down the curve.
back_1_joint.scaleX = $scale;
back_2_joint.scaleX = $scale;
back_3_joint.scaleX = $scale;
back_4_joint.scaleX = $scale;
back_5_joint.scaleX = $scale;
back_6_joint.scaleX = $scale;

// Inverse scale for volume preservation
back_1_joint.scaleY = pow($invScale,back_1_joint.pow);
back_1_joint.scaleZ = pow($invScale,back_1_joint.pow);
back_2_joint.scaleY = pow($invScale,back_2_joint.pow);
back_2_joint.scaleZ = pow($invScale,back_2_joint.pow);
back_3_joint.scaleY = pow($invScale,back_3_joint.pow);
back_3_joint.scaleZ = pow($invScale,back_3_joint.pow);
back_4_joint.scaleY = pow($invScale,back_4_joint.pow);
back_4_joint.scaleZ = pow($invScale,back_4_joint.pow);
back_5_joint.scaleY = pow($invScale,back_5_joint.pow);
back_5_joint.scaleZ = pow($invScale,back_5_joint.pow);
back_6_joint.scaleY = pow($invScale,back_6_joint.pow);
back_6_joint.scaleZ = pow($invScale,back_6_joint.pow);
```

Now play with the back control and then modify the curve. See how you can get the squash and stretch to change how much it is influenced, simply by using the

Autodesk® Maya® Master Classes – Instructor Notes

SIGGRAPH™ 2006



Master Classes

## Instructor Notes – Animator Friendly Rigging Part II

Jason Schleifer

animation curve? On your characters, you can use this to graphically tune them as you wish, and then lock the curves so the animators don't have access to them.

**File:** *exploringBackStretch.ma*

#### Movable Pivot

The last thing we want to explore before working on our rig is the idea of a movable animation pivot.

If you select any object with Maya and hit the **Insert** key (**Home** key on **OS x**), you can move the pivot around.

#### 50. Create a New Scene

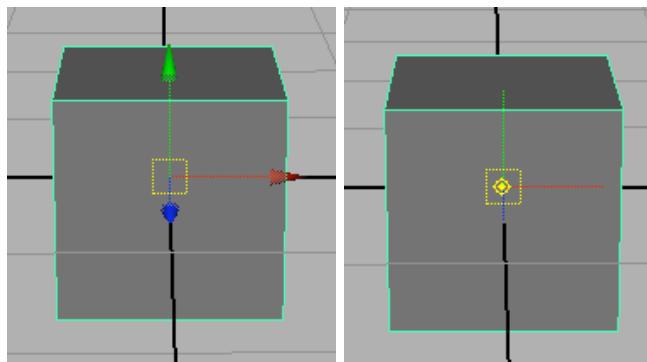
- Choose **File > New**

#### 51. Create a polygon cube

- Choose **Create > Polygon Primitive > Cube**

#### 52. Move the Pivot

- Enter the **Move** tool using the hotkey **w**
- Hit **Insert** (windows/linux/unix) or **Home** (os x)
- Move the pivot to the edge of the box.
- Hit **Insert** or **Home** to go back to the **Move** tool



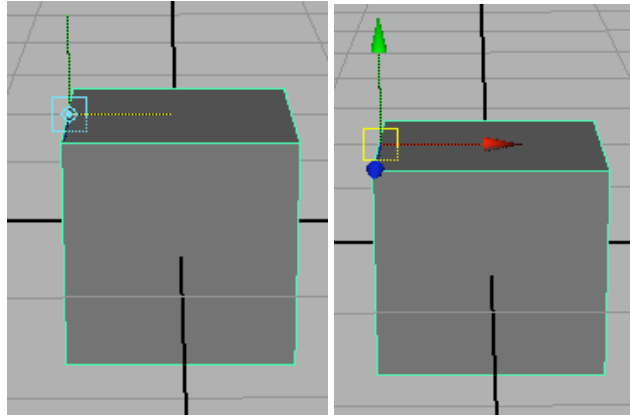


Figure 75 - moving the pivot

So moving the pivot on an object is a pretty useful thing to do. The problem is that this feature is hidden from users who don't have a background in Maya. So if you want to add the ability to do this to any object, it's important to create an icon that the user can select which will allow them to grab the pivot and move it interactively.

#### 53. Create a locator to control the pivot

- Choose **Create > Locator**

#### 54. Attach the locator to the pivot

- Choose **Window > General Editors > Connection Editor**
- Select **Locator1**
- Choose **Reload Left**
- Select **pCube1**
- Choose **Reload Right**
- On the **Left** click **Translate**
- On the **Right** click **RotatePivot**

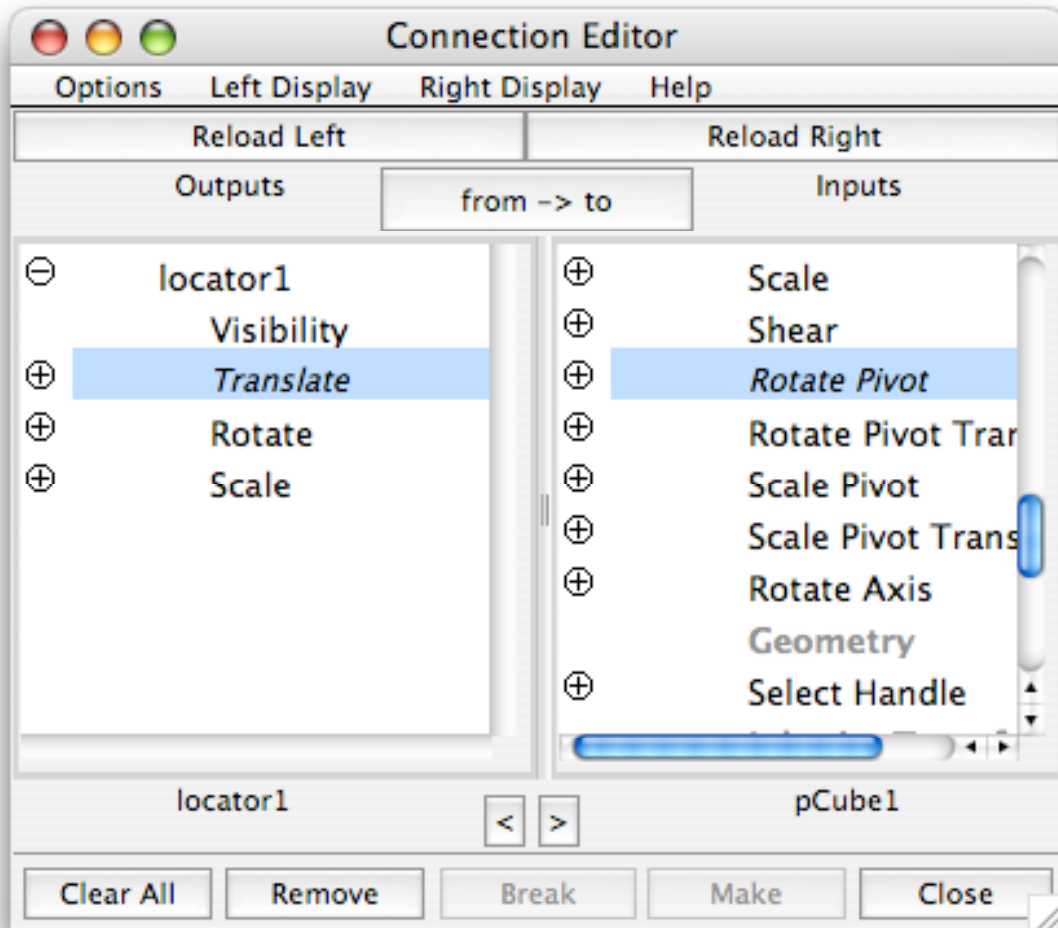


Figure 76 - locator1.translate connected to the pCube1.rotatePivot

#### 55. Parent locator1 to pCube1

- In the Outliner, drag **locator1** onto **pCube1**

#### 56. Rename the objects

- Rename **locator1** to **pCube1\_pivot**



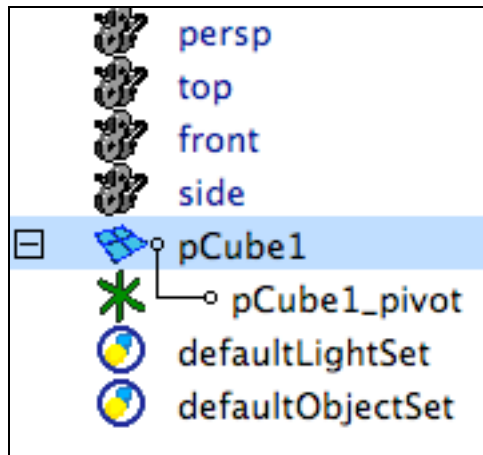


Figure 77 - locator1 renamed to pCube1\_pivot and made a child of pCube1

You'll notice now that when you move **pCube1\_pivot**, and then rotate **pCube1**, it moves the pivot! The great thing is that the animator can animate this pivot as they want to get the character to be able to rotate from wherever necessary.

One artifact of moving the pivot on an object once it's animated is that it will *change the position* of the object. This is because the object's *translation and rotation* are based on the location of the pivot.

#### 57. Find rotation artifact

- Select **pCube1**
- Rotate **pCube1** 30 degrees in **Z**
- Select **pCube1\_pivot**
- Move the object.

Notice that as you translate **pCube1\_pivot** the position of **pCube1** changes? Imagine how this can be for an animator if they're trying to animate a character changing from a standing position to a sitting position. When standing, you want the pivot at the top of the hips. When sitting, you want the pivot at the **base** of the hips, where the buttocks touches the seat.



## Master Classes

# Instructor Notes – Animator Friendly Rigging Part II

Jason Schleifer

The animator can work with this on their own by moving the pivot and then translating the object to try and match it up, but that can be quite difficult. Wouldn't it be easier if there were a simple way to snap the pivot to the new location *without* moving the object?

### Determine the steps

Let's take a look at the steps an animator would go through to move the pivot without making the object jump.

#### 58. Create a “second” axis control for the animator

- The animator can't move the first axis without it modifying the object, so we need to give them a second pivot that they can use to position where they want the new one to be.

#### 59. Save a keyframe for the translation and rotation of the object and pivot on the previous frame.

- Find the pivot locator
- Save a keyframe for the object and pivot on the previous frame.

#### 60. Move the pivot to the new location

- Get the local position of the pivot
- Get the world position of the pivot
- Move the old pivot to the new location
- Move the object to the new world position

#### 61. Clean everything up

- Remove the “second” pivot.
- Save a keyframe for the new location on the current frame

Pretty straight forward, right? Well, not really. Not if an animator actually wants to get work done. That's why we have a series of Mel procedures to make this easier.

#### **Procedure #1: js\_pivot\_create<sup>ix</sup>;**

This script will be something that you use when you set up your characters. I don't recommend you give it to the animators, because you want to specify the controls they can use to animate.

It will work on a number of selected objects, and it returns the names of the new pivot controls.

Usage: Select a number of objects and type:

```
source js_pivot;  
js_pivot_create;
```



#### **Procedure #2: Create a 2<sup>nd</sup> locator for the animator to move**

This is something that your animators will be doing, so you'll need to provide it to them in some way. What they'll be doing is selecting the object and then executing the command:

```
source js_pivot;  
js_pivot_createMov;
```



#### **Procedure #3: Snap the object to the new location**

If the user has the object or the pivot selected, it's now time to snap the pivot to the new location. To do this, the user simply selects an object and executes the command:

```
source js_pivot;  
js_pivot_snap;
```



There are two other commands which come with the **js\_pivot<sup>x</sup>** script which you should also set up for your animators, and they allow them to turn on and off the pivots.

They both take the same arguments, but one works for the pivot, and the other works for the movement pivot (the one the animator can use to move the pivot around).

They are: **js\_pivot\_toggle** and **js\_pivot\_mov\_toggle**.



To toggle the pivot *on* you would select the object and use:

```
js_pivot_toggle 1;
```

To toggle the pivot *off*, try:



## Master Classes

# Instructor Notes – Animator Friendly Rigging Part II

Jason Schleifer

```
js_pivot_toggle 0;
```

And to simply toggle the pivot back and forth, use:

```
js_pivot_toggle -1;
```

We've covered most of the tools for what we'll be using for creating our torso animation rig. Let's review what the requirements are for the rig so we can start creating it.

### Torso Animation Rig Requirements

1. Allows for rotation of hips and shoulders
2. Allows for rotation in all axis – Bend, Side to Side ,and Twist
3. Allows for independent motion of shoulders and hips.
4. Allows for relocation of pivot.

Remember, for each control we have a set of requirements that must be met.

### Rig Requirements – CONTROL

- **Simple controls**
- **Animation should be easily transferable.**
- **Controls should be unique and make immediate sense.**
- **Controls should have the correct rotation orders.**
- **Controls should be named correctly.**
- **Only be able to set keyframes on controls we want animators using.**

And we have a set of requirements that the *entire* rig must have.

### Rig Requirements – ALL

These are general rig requirements that all animation rigs should have. Usually these are done at the end of the rigging process.

- **Only be able to select what the animators can use to animate.**
- **Clean outliner when finished.**
- **Can move the rig to any position and orientation and have it work.**

## Creating the Torso Rig

Now let's move forward and actually create the torso rig for our character. We'll be using the character **JJ** that comes with the DVD. He's a relatively simple character, and should suit our purposes for demonstrating the rigging techniques.

The first step is making sure that we have a good working directory to work with **JJ** for creating our rig.

### 62. Add JJ Model to the Library

- Create a new directory called JJ in **Library > Characters**

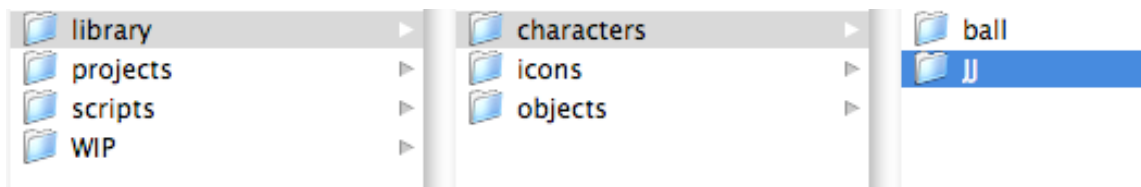


Figure 78 - Creating a JJ directory

- Add a **model** directory inside the **JJ** directory
- Copy **example\_files > jj.ma** to **library > characters > JJ > model**

### 63. Open File *jj.ma*

- Choose **File > Open**
- Navigate to your new **model** directory
- Choose *jj.ma*

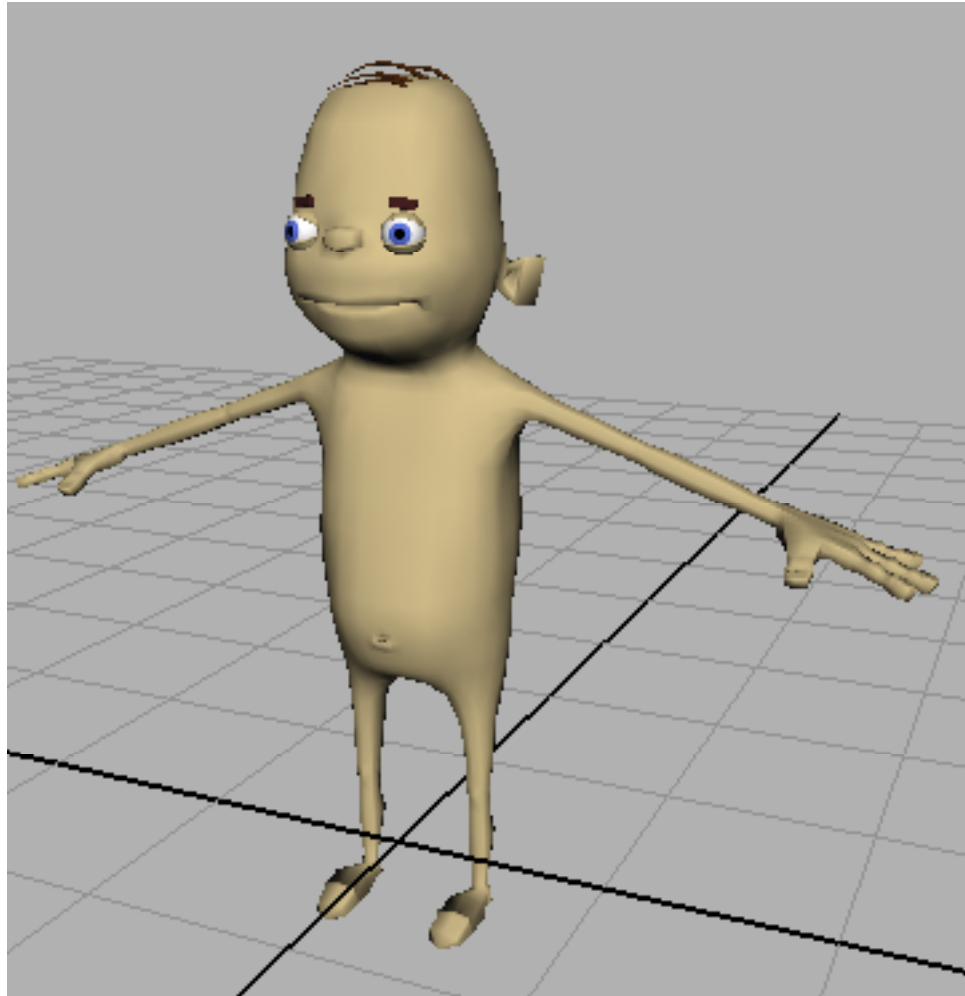


Figure 79 - jj.ma loaded into Maya

One of the first things that we're going to do with our rig is break it apart into smaller pieces. Each of these pieces will be parented to the joints that will make up our animation rig.

This will be our **Low Resolution** model.

Why do we create a **low resolution** version of our character? Because it's *much faster than a skinned model*.



## Master Classes

# Instructor Notes – Animator Friendly Rigging Part II

Jason Schleifer

A fully skinned character can take seconds to update each frame. A model with geometry simply parented to the joints will update in *real time*. The faster the updating, the faster the animator can work, and the better their resulting animation will be.

Remember this equation, because it's one of the most important ones you'll ever learn:

**FASTER = YES!!**

If you ask any animator if they'd like something faster, their answer will be an extremely loud and shocking "YES!!".

We want our animation rig to be as fast as possible, so instead of providing the animator with a final skinned character to animate, we're going to give them a fast, trimmed down character. A low-resolution / fast-animating machine.

To do *that*, we must break apart the model into pieces that we can parent to the joints.

Since we're going to be working on JJ in various stages, we will need to segment him into separate parts. I recommend doing *gross* segmenting (not blood and guts, *large* segments) first. Just break him apart by the torso, arms, legs, head, etc.

Then later we'll refine those segments into smaller bits for us to use as we need them. I also recommend keeping those segments all together in a single file that we update each time we break apart another section. This way, at the very end of the rig you will have a single file you can import with the character broken into segments that you can use very easily.

## Animation Rigging Vs. Skinning

Before we continue on with the creation of the rig, I want to clarify one item. This course is focused on creating an animation rig that your animators like. Its focus is *not* on creating the final skinned character. That topic could easily cover 4 *more* dvds, and is definitely not the focus here. Standard practice in studios does dictate a separation between the two systems. Usually there are two separate rigs.. a skinning rig that deforms the geometry and has a full skin/muscle system driving it, and an animation rig that is controlled by the

animators. There's generally some way that the animation is transferred *from* the animation rig *to* the skinning rig. This can be done by copying the animation from the skeleton on the animation rig to the skeleton on the skinning rig, by constraining the skin rig *to* the animation rig, or a myriad of other ways.

This course *will not cover* that aspect of rigging. It is meant merely to introduce you to interesting and useful techniques of creating an animation rig.

There. Now that that's out of the way, let's continue with our character!

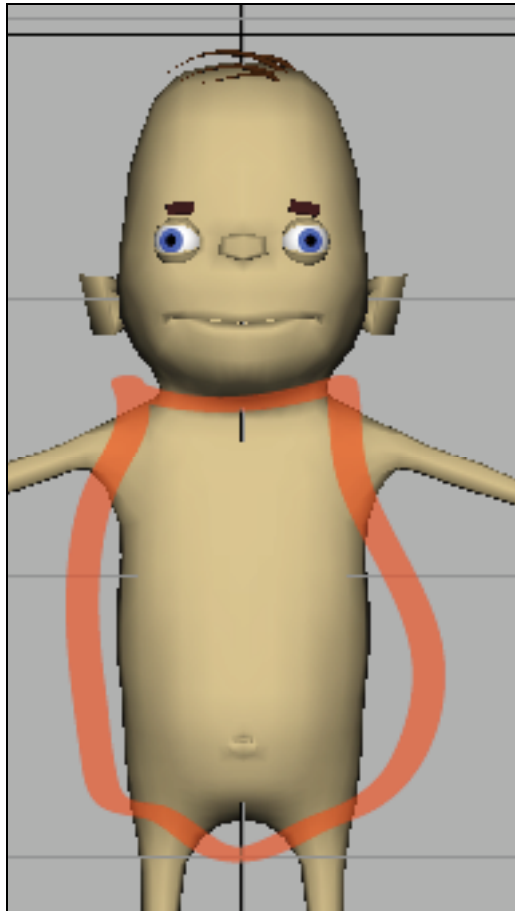


Figure 80 - the area of JJ that we're going to separate

One of the great things about starting with the torso, is that it separates each part of the body. So we can use this as a great starting point to breaking apart the body for each piece of the rig.



#### 64. Select the faces on the torso mesh

- Select **jj\_geo**
- Choose **RMB > Face** so you can select the faces of the polygons.
- Select the faces that make up the torso. You can see them in the images below. Your goal is to pick everything except the neck, head, arms, and legs.

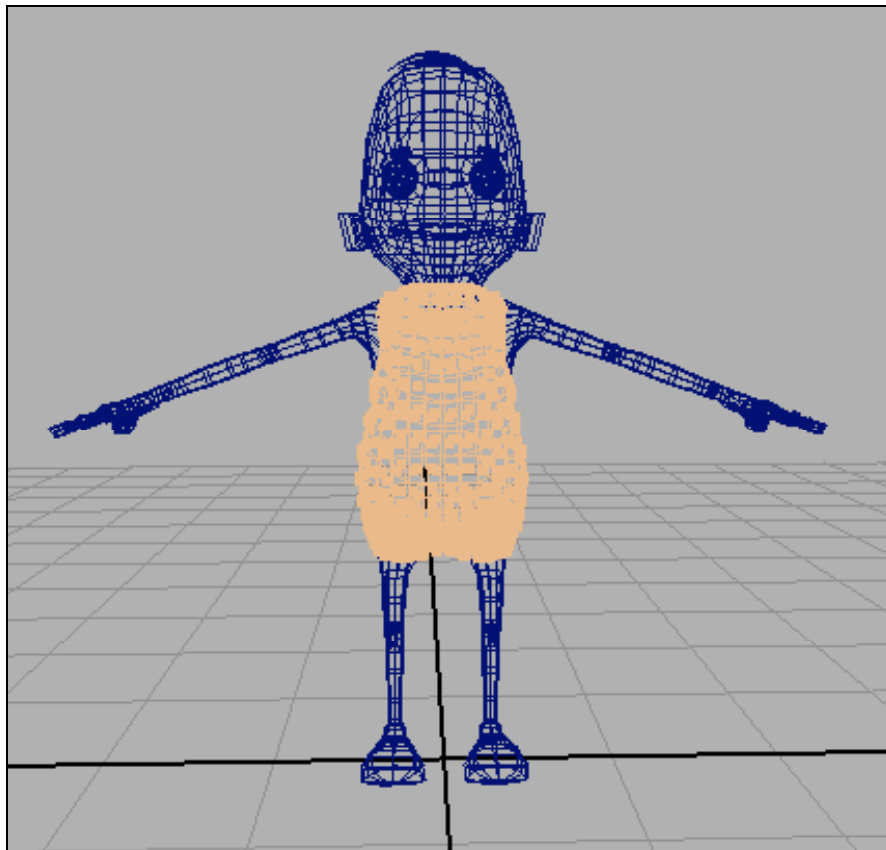


Figure 81 - torso faces

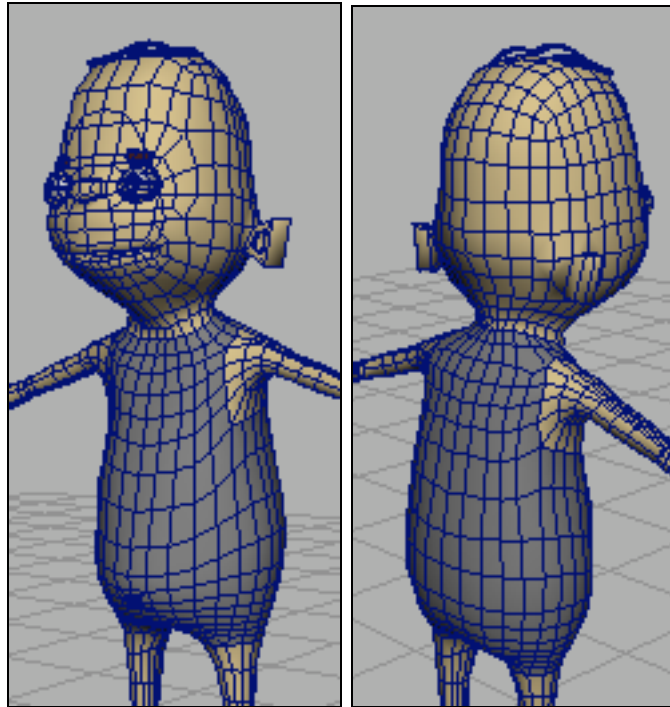


Figure 82 - You can see a close up of various areas of what is selected for the torso faces.

#### 65. Separate those polygons

- Choose **Edit Meshes > Keep Faces Together** and make sure it's checked **ON**. If it's off, when you separate the character each individual face will be it's own polygon. *Nasty!*
- Choose **Meshes > Extract**
- Choose **Meshes > Separate**

You now have separate body parts for JJ, and in fact if you select the torso, you'll see that you just have it selected on it's own.

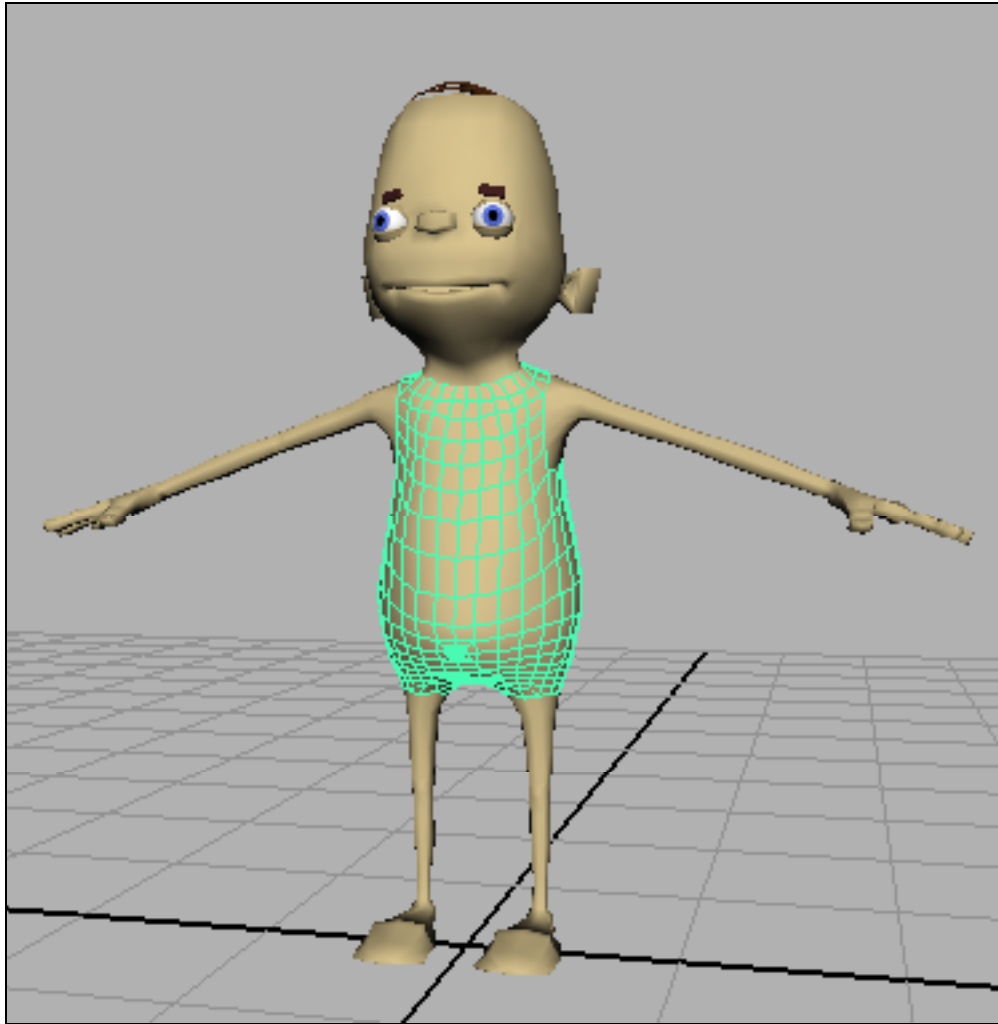




Figure 83 - JJ torso

#### 66. Rename each of the new pieces of geometry

- Open the **Outliner**
- Open **jj\_grp** by clicking on the little 
- Open **jj\_geo** by clicking on the little 
- Notice you now have a list of polySurfaces

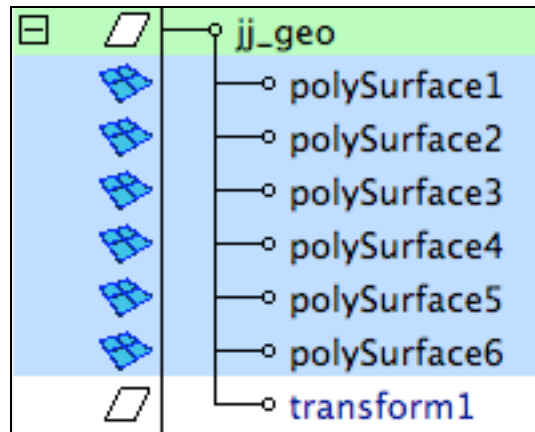


Figure 84 - Resulting Polygonal Surfaces after Separating the geometry

- Name each polySurface based on the body part, so the result looks something like:

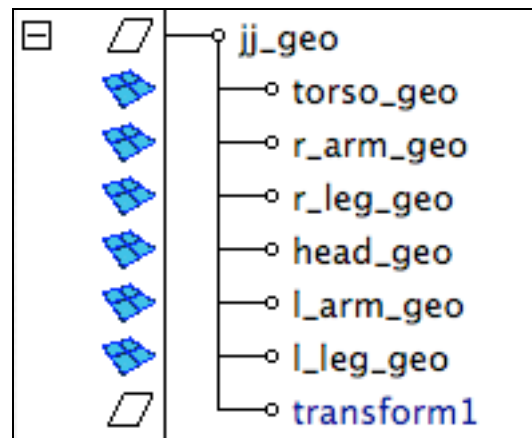


Figure 85 - Surfaces renamed

#### 67. Delete the construction history on the model

- Select the six resulting surfaces (**torso\_geo**, **r\_arm\_geo**, **r\_leg\_geo**, **head\_geo**, **l\_arm\_geo**, **l\_leg\_get**)
- Choose **Edit > Delete By Type > History**
- This will remove that **transform1** object, and remove the construction history on the object.

#### 68. Save the broken apart geometry in a file that we will keep working on



## Master Classes

# Instructor Notes – Animator Friendly Rigging Part II

Jason Schleifer

Since we're going to be working with this character a number of times, and we're not sure exactly how broken up he's going to need to be for the final rig, it's important to begin saving him in his current state in a place we can easily come back to.

I like to create a directory called **model\_lowRes** in the library for my character. This means that we can use these pieces as our low resolution geometry. Note, you don't have to use actual broken pieces of your final model as your low resolution geometry, but it helps the animator get an idea for the shape of the model they should expect to see.

- Create **model\_lowRes** in the **Library > JJ** folder



Figure 86 - creating model\_lowRes folder

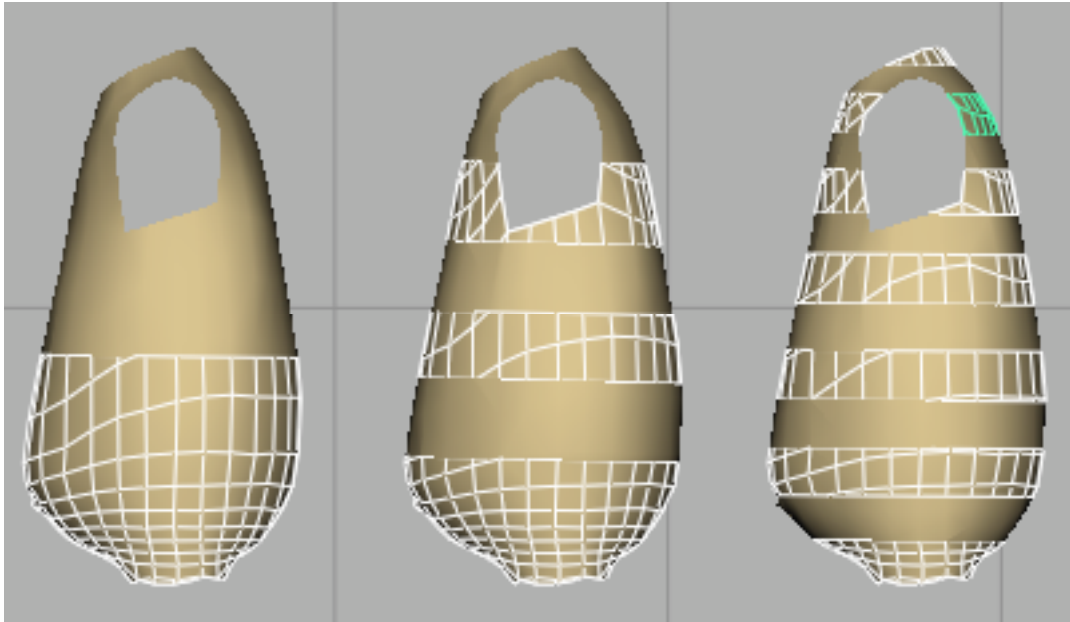
- Save the file as **jj\_lowRes.ma**

We now have a file we can keep breaking apart as we need to while working on our rig. In the end, we'll have a fully segmented model with all the proper naming.

## Segment The Torso

When determining how to break up the torso, it's important to think about what how many segments you need to get the type of motion you want. The more segments, the more flexibility, and the easier it is to control. However, with too many segments, the rig could run too slow.

What is a segment? A segment is one section of geometry. Look at the figure below for an example.



*Figure 87 - the torso broken up into 3 different amounts of segmentation. The torso on the right has 2 segments. These aren't enough to really animate the body. The middle has 6 segments. This can give you a pretty good range of motion. The right one has 13 segments. Probably a few too many unless your character has a reason to be that bendy.*

How many segments should you use? It depends on your character. On average, I'd say you should have at least 5 or 6 segments to work with for your character. This will give you enough joints to manipulate the character with the speed necessary for the animator to love you forever.

#### **How to segment the torso.**

One of our requirements for our torso rig is to keep the hips and shoulders steady. The best way to do that is to make sure that they are 100% controlled by our animation controls. The center part of the back will be the part that is controlled by the splineIK. Thus, you want to segment your character in to even chunks with enough segments in the middle so that the back can bend easily. That means 5 or 6 segments *excluding* the hips and shoulders.

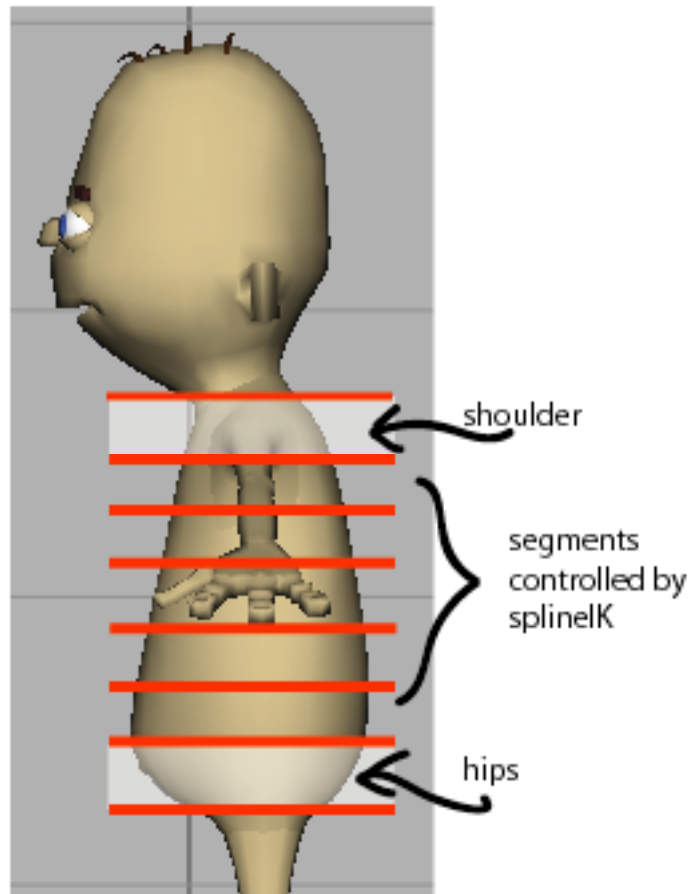



Figure 88 - Segment the body enough for the hips, shoulders, and back sections.

A quick way to segment the torso is to use the **js\_cutPlane.mel<sup>xi</sup>** script which will allow you to take a series of nurbs planes and cut the polygon in half.

#### 69. Segment the shoulder and hip areas

- Select **torso\_geo**
- Click on the **Add Plane for Cutting Polygonal Geometry** button in the Animator Friendly Rigging shelf.
- The cut plane will be created and automatically selected. Move it so that it's at the place where you want the hips to be segmented.



- Click on the Add Plane for Cutting Polygonal Geometry button again to create another plane. *Note: you can have either the plane or the torso selected. If the plane is selected, it will automatically figure out that you want to create a plane for the torso geometry.*
- Move the second plane to the bottom of where the shoulders should be. Check the images bellow for ideas on where they should go.
- Select one of the planes, or the torso.
- Click on the Cut Selected Objects With Planes button on the Animator Friendly Rigging shelf. 
- The torso will automatically be cut and separated wherever the planes were located on the body. The planes will also be deleted when you're finished.

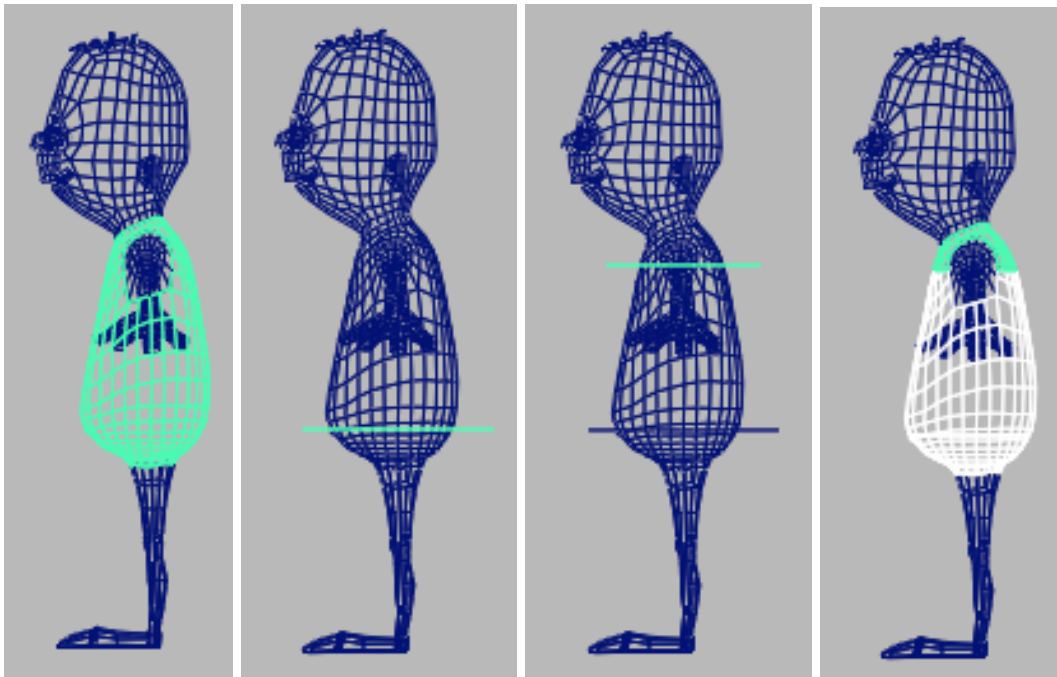


Figure 89 - Selecting the Torso, adding planes for cutting, and finally cutting the body.



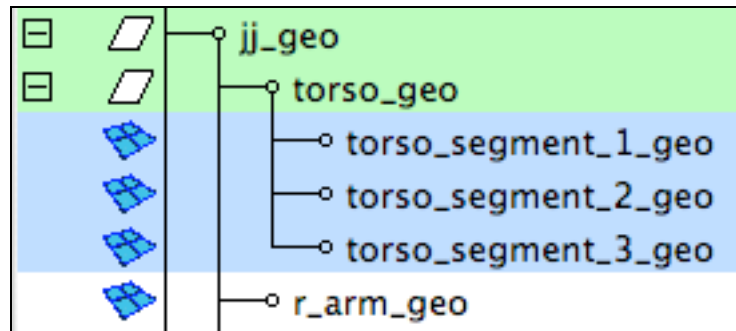


Figure 90 - the result of cutting the torso, you should have three segments.

#### 70. Rename the geometry

- Rename the pieces of geometry so they make more sense. Don't worry about the fact that they're all under **torso\_geo** at this point, we'll worry about that later.

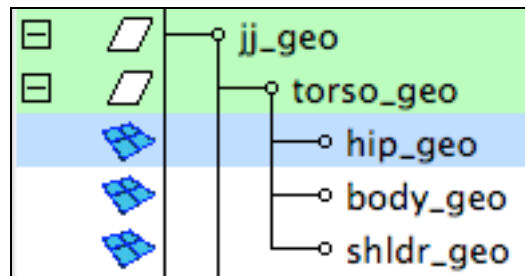


Figure 91 - renaming the geometry

#### 71. Hide all of the geometry except for the torso.

- Hide everything except the torso geometry. This will make the rig easier to create.

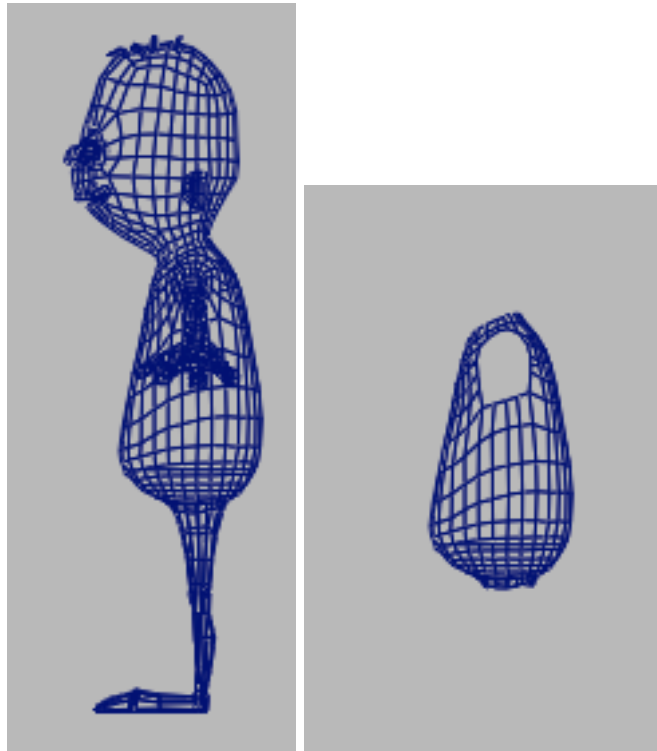


Figure 92 - Geometry hidden except for torso geometry.

#### **Break apart the mid torso for the segments.**

Our next step is to break apart the middle of the torso **body\_geo** into segments. We want to make sure that these are evenly spaced all along the body, and lined up with the skeleton that we're going to use for our **splineIK**.

The best way to do *that* is to first create the skeleton, and then add the nurbs planes at the location of each joint in the skeleton.

The skeleton will go from the **base** of **body\_geo** to the **top** of **body\_geo**. It will make it easier to do this if you temporarily hide **hip\_geo** and **shldr\_geo**.

#### **72. Hide hips and shoulders**

- Select **hip\_geo** and **shldr\_geo**
- Hit **Ctrl+h** to hide them.

#### **73. Create a joint segment going from the base to the top**

- Choose **Skeleton > Joint Tool**
- Click at the base of **body\_geo** in the *middle* of the back horizontally.
- Hold down **SHIFT** and click again at the top of **body\_geo**. *Note: **SHIFT** constrains the next joint horizontally or vertically, depending on which direction you move. It's incredibly handy for making nice straight joints!*
- This will create a segment that travels along the body.
- Hit **Enter** to confirm your joint creation.

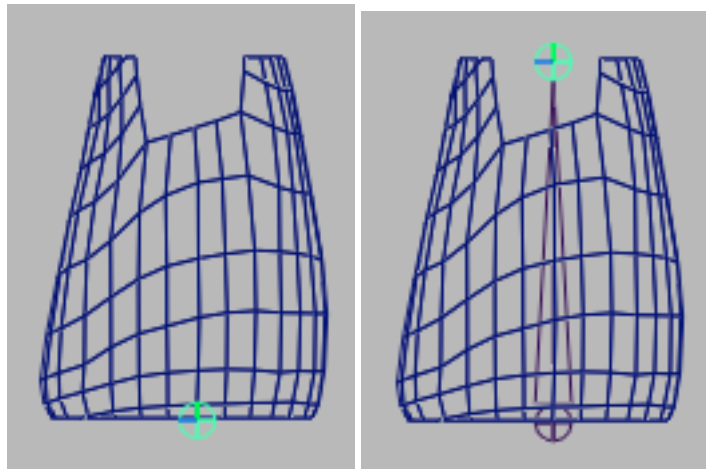


Figure 93 - creating a joint that goes straight up the center of the body

#### 74. Segment the joint

- Select **joint1**
- Click on the **Split Selected Joint** tool in the **Animator Friendly Rigging** shelf.
- Set **Segments** to **6**

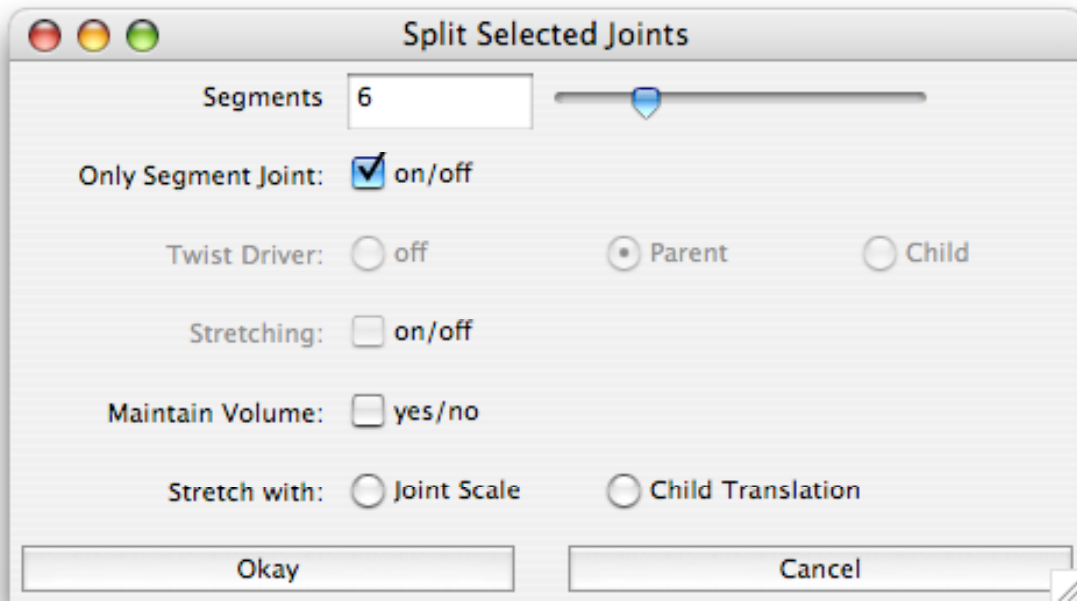


Figure 94 - Split Selected Joints, segments set to 6

- Click **Okay**

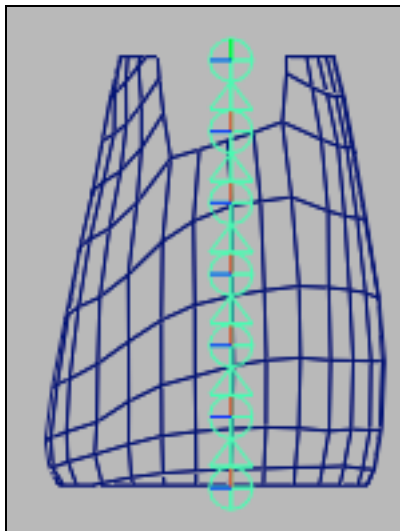



Figure 95 - Skeleton broken apart into 6 segments

#### 75. Create cutting planes for each joint

- Select **body\_geo**
- Click on the Add Plane for Cutting Polygonal Geometry button in the Animator Friendly Rigging shelf. 
- We want to position the plane so it's at the location of the *second* joint in the back. If we place it at the *first* joint, there's no place to cut!

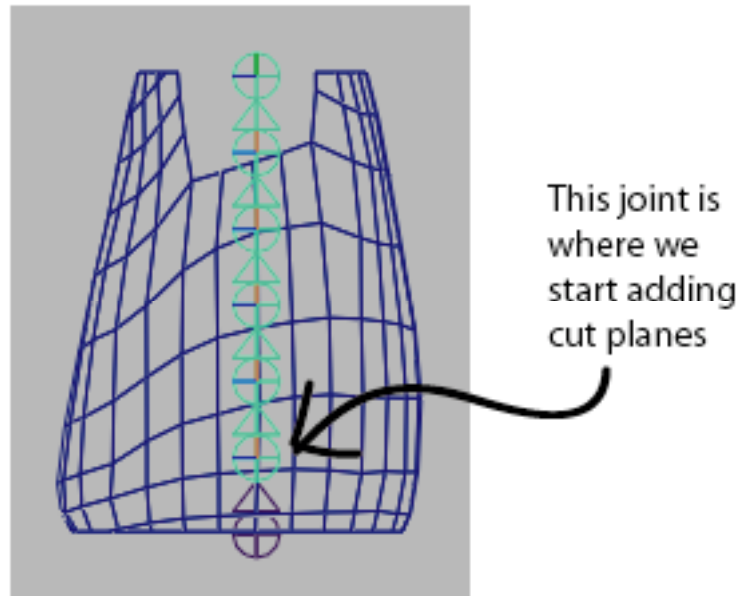
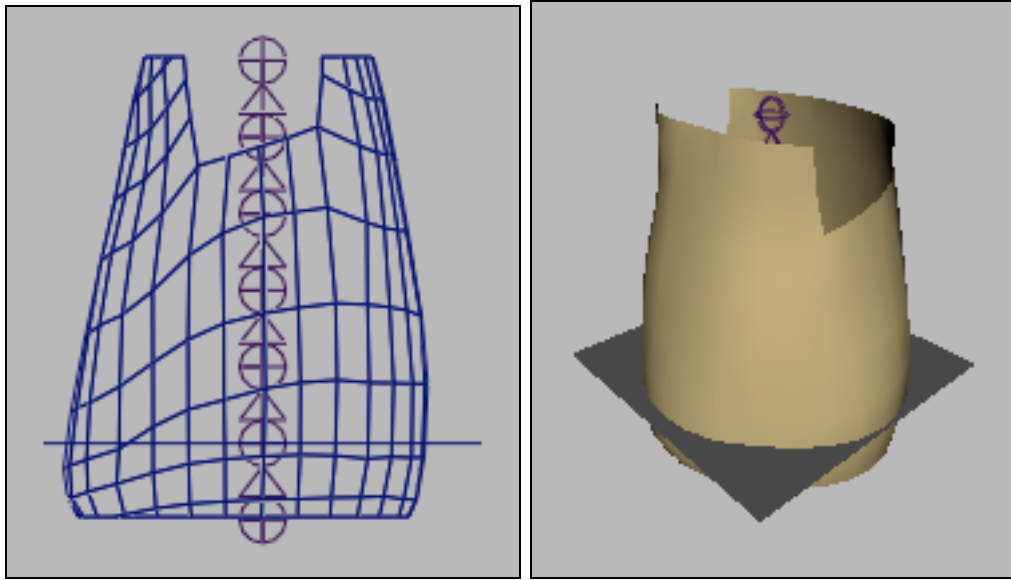


Figure 96 - where to place the first cut plane.

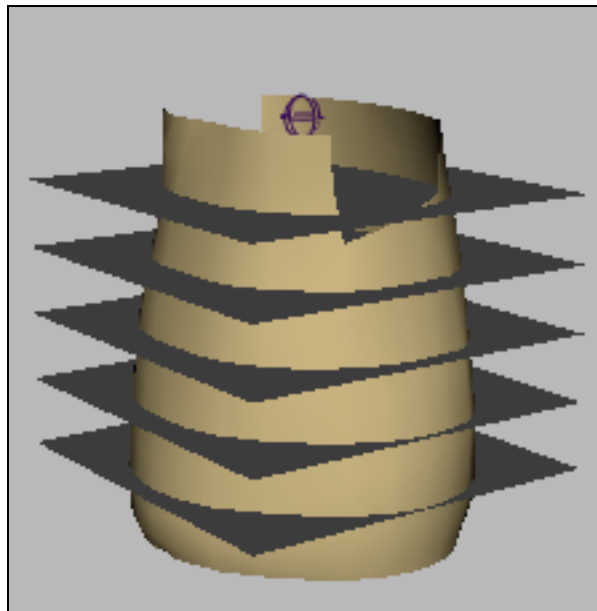
- The easiest way to position the plane is to simply *parent* it to the joint we want. With the **cutPlane** selected, add **joint1\_seg\_1** to your selection.
- Hit the “p” hotkey. This will parent the plane to joint1\_seg\_1.
- In the **Channel Box**, set the **Translation** values to **0 0 0**.
- Set the **rotation** values to **0 90 0**.
- The plane should lay flat, intersecting the torso.

*Note: it doesn't have to completely intersect the torso. The plane is used to specify a position and direction for the cut. It will happen correctly even if the geometry doesn't intersect.*




*Figure 97 - Cut plane position and orientation.*

- Repeat the process until you have a plane for each joint except for the first and last (the bottom and top of the torso).



*Figure 98 - Cut planes for each joint*

#### **76. Cut the Geometry**

- Select the **body\_geo**
- Click on the Cut Selected Objects With Planes button on the Animator Friendly Rigging shelf. 
- The body\_geo will automatically be cut and separated wherever the planes were located on the body. The planes will also be deleted when you're finished.

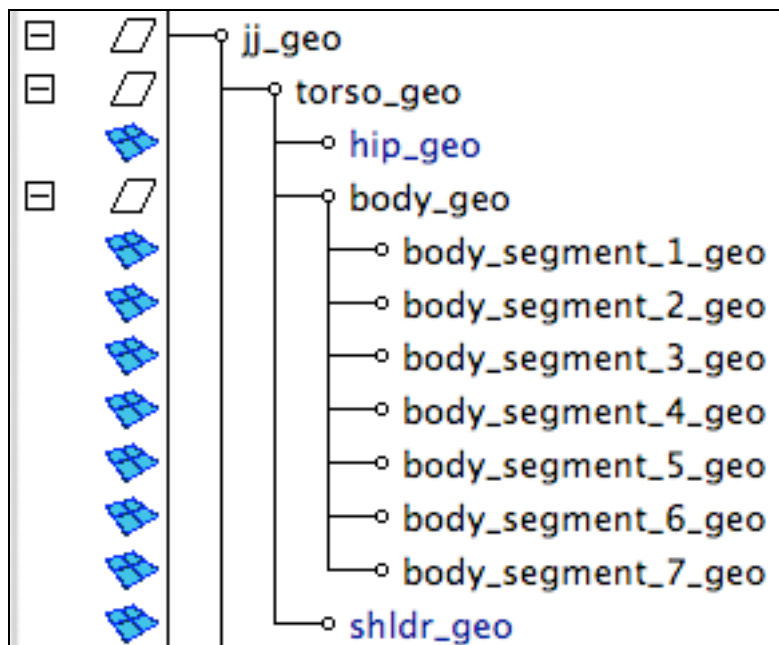


Figure 99 - body broken up into separate segments

Notice that there appear to be more pieces of geometry than there were cut planes. That's because of the very top of the shoulder. The geometry there was cut through the arm holes.



Figure 100 - torso cut through arm holes

We can fix that with the Polygon Unite tool.

#### 77. Unite the top of the shoulders into one piece of geometry

- Select the two pieces of geometry at the top of the torso
- Choose **Mesh > Unite**.
- Choose **Edit > Delete By Type > History**

#### Rename the Geometry

Next we want to name the body pieces. Currently they're a bit over the map, we have **body\_segment\_1\_geo**, **polySurface1**, **torso\_geo**, etc.

I like to choose names for my geometry that will be the same as the joints they will be parented under. This makes it much easier to create Mel scripts to perform repetitive tasks.

#### 78. Name the individual pieces

- Rename the back pieces to **torso\_#\_geo**.



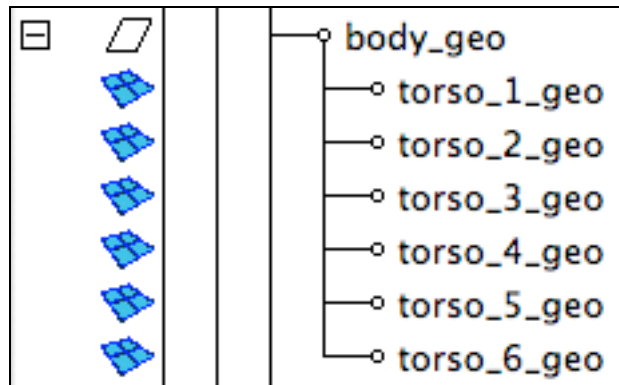


Figure 101 - renaming the torso pieces

- Place all the pieces under torso\_geo

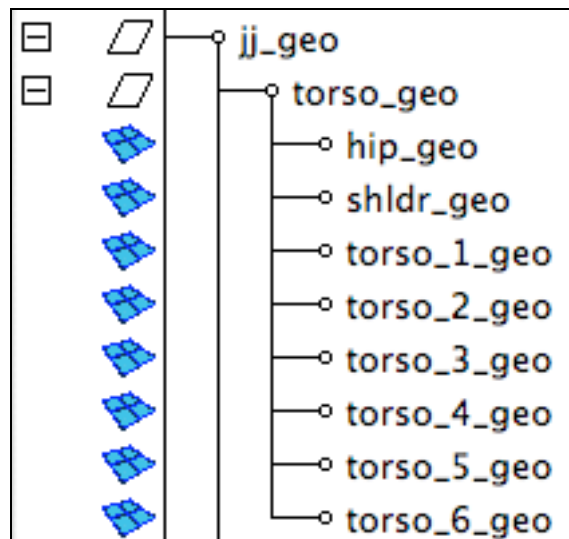


Figure 102 - all the individual pieces parented under torso\_geo

#### Save the latest version of the broken apart geometry

Remember, as we break apart the geometry, we want to keep saving it to our lowRes folder for use at a later date.

#### 79. Show the other geometry

- Select all the bits of geometry we hid earlier, and make them visible.

#### 80. Export jj\_grp

- Select **jj\_grp**
- Choose **File > Export Selected**
- Navigate to your **model\_lowRes** folder
- Save the file as **jj\_lowRes.ma**. *Note: You can save it as another filename if you like, in order to keep previous versions around. For example, save it as **jj\_lowRes\_v01.ma**.*

Delete everything except for **torso\_geo**. We're just going to be working on the back, so we want to make sure we've got a clean scene to work with.

Now it's time to start creating the back controls!

#### Parent the Geo to the Joints

Since we've already used a series of joints to cut our geometry, it makes sense to use them again as the basis for our rig.

#### 81. Rename the joints

- Since we've already chosen names for our geometry, let's name the joints in a similar fashion.
- Name the joints **torso\_#\_joint**

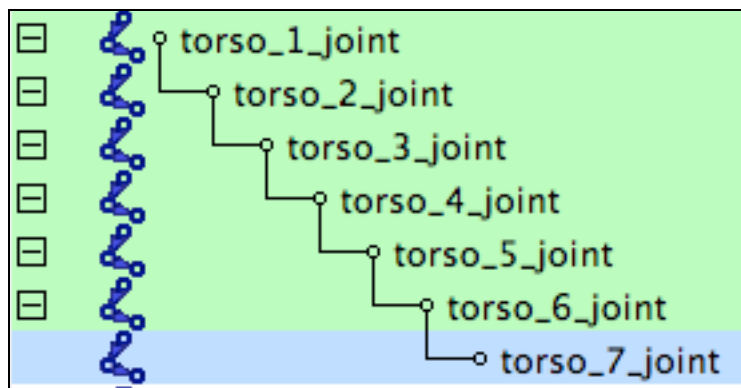


Figure 103 - renaming the torso joints

#### 82. Parent the geometry to the appropriate joint



## Master Classes

# Instructor Notes – Animator Friendly Rigging Part II

Jason Schleifer

Normally I would not recommend parenting geometry to a joint, mainly because of the rule that it should be easy to swap out new geometry for old geometry. However, if we constrain each of these pieces of geometry to the joints *instead* of parenting them, it will slow down our scene *considerably*.

And by considerably, I mean a *lot*.

I tried a quick experiment. I parented each piece of geometry to the corresponding joint and played back some animation as fast as possible. Maya clocked it playing back at around 100fps.

I then kept the same animation and used a *parent constraint* instead of parenting.

Maya played back at 85 to 90 fps. You can see that this is a really large decrease in speed, especially considering it's a small scene to begin with.

So how do we solve our problem of making sure we can swap out geometry quickly and easily? By doing what we just did in step 1: naming the joints and geometry correctly. If we do this, we can easily write a quick script to delete all the old geometry (delete “\*\_geo”), import the new geometry, and parent it to the appropriate joint.

However, we don't have to worry about that at this point. All we need to do *now*, is simply parent the geometry to the appropriate joint.

- Parent **torso\_1\_geo** to **torso\_1\_joint**.
- Parent **torso\_2\_geo** to **torso\_2\_joint**
- Parent **torso\_3\_geo** to **torso\_3\_joint**
- Continue until all the torso\_#\_geo polys are parented to the appropriate joints.

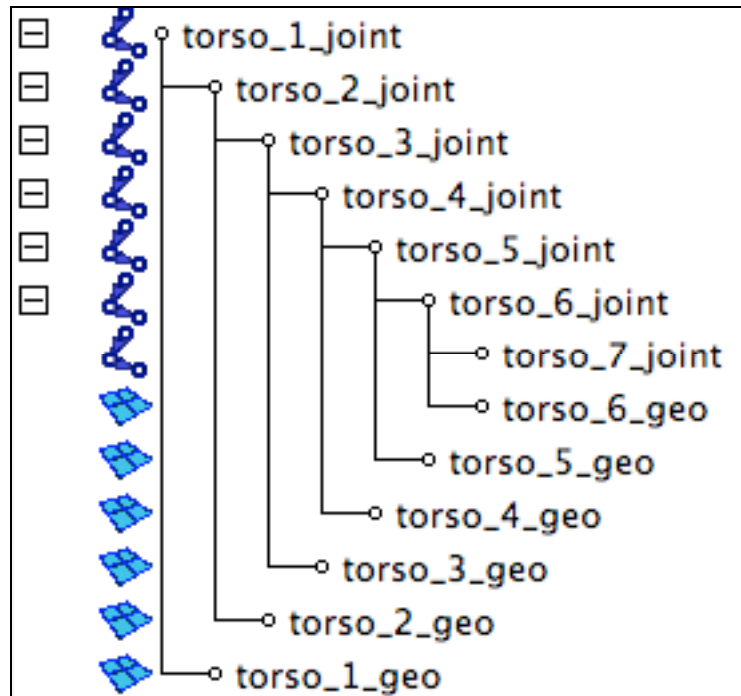


Figure 104 - parenting the geometry to the joints.

Notice that **hip\_geo** and **shldr\_geo** aren't parented to anything yet. That is because we haven't created joints for them yet. Remember, we're not including them in the splineIK system, because we want their motion to be *completely* derived from the position of the hip and shoulder controls.

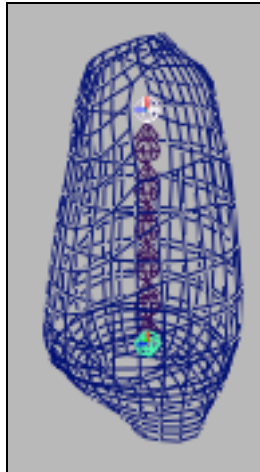
#### 83. Create a hip\_joint

- Select **torso\_1\_joint**
- Duplicate it by choosing **Edit > Duplicate**
- Delete all of it's children
- Rename it **hip\_joint**

You now have a hip joint in the same location as torso\_1\_joint.

#### 84. Create shldr\_joint

- Select **hip\_joint**
- Duplicate it by choosing **Edit > Duplicate**
- Choose the **Move** tool.
- Hold down the “v” hotkey to enable **Point Snap** mode.
- With the **Middle Mouse Button**, Click and drag near **torso\_7\_joint** in the perspective view. This will snap the new joint to the position of **torso\_7\_joint**.
- Rename the new joint **shldr\_joint**



*Figure 105 - hip and shldr joints positioned at the top and bottom of the spine*

#### 85. Parent the hip and shoulder geometry

- Parent **hip\_geo** to **hip\_joint**
- Parent **shldr\_geo** to **shldr\_joint**

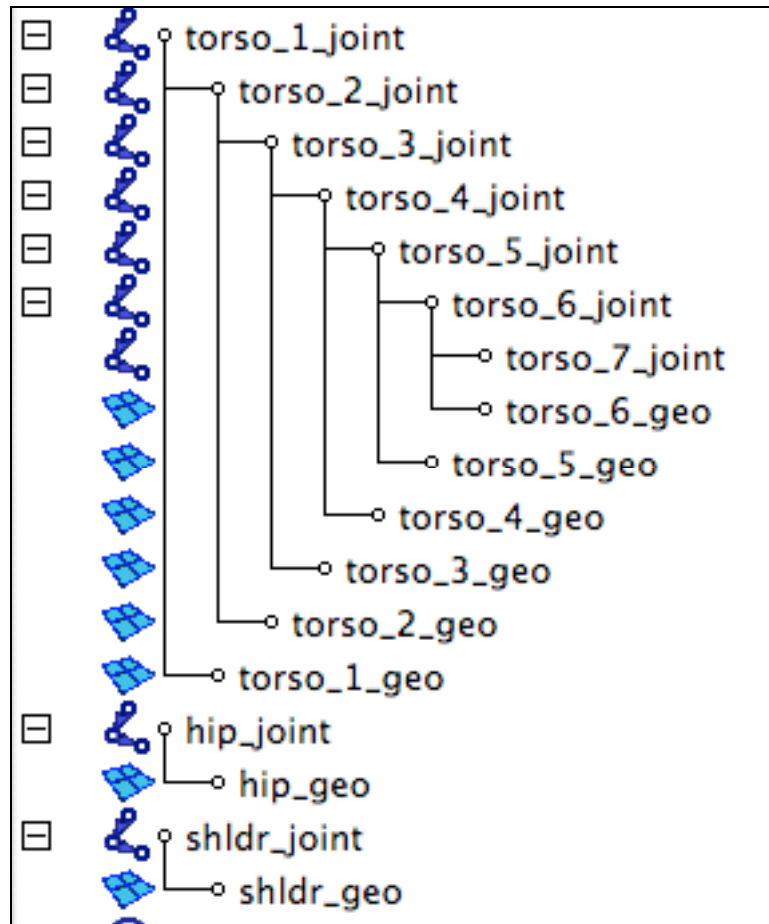
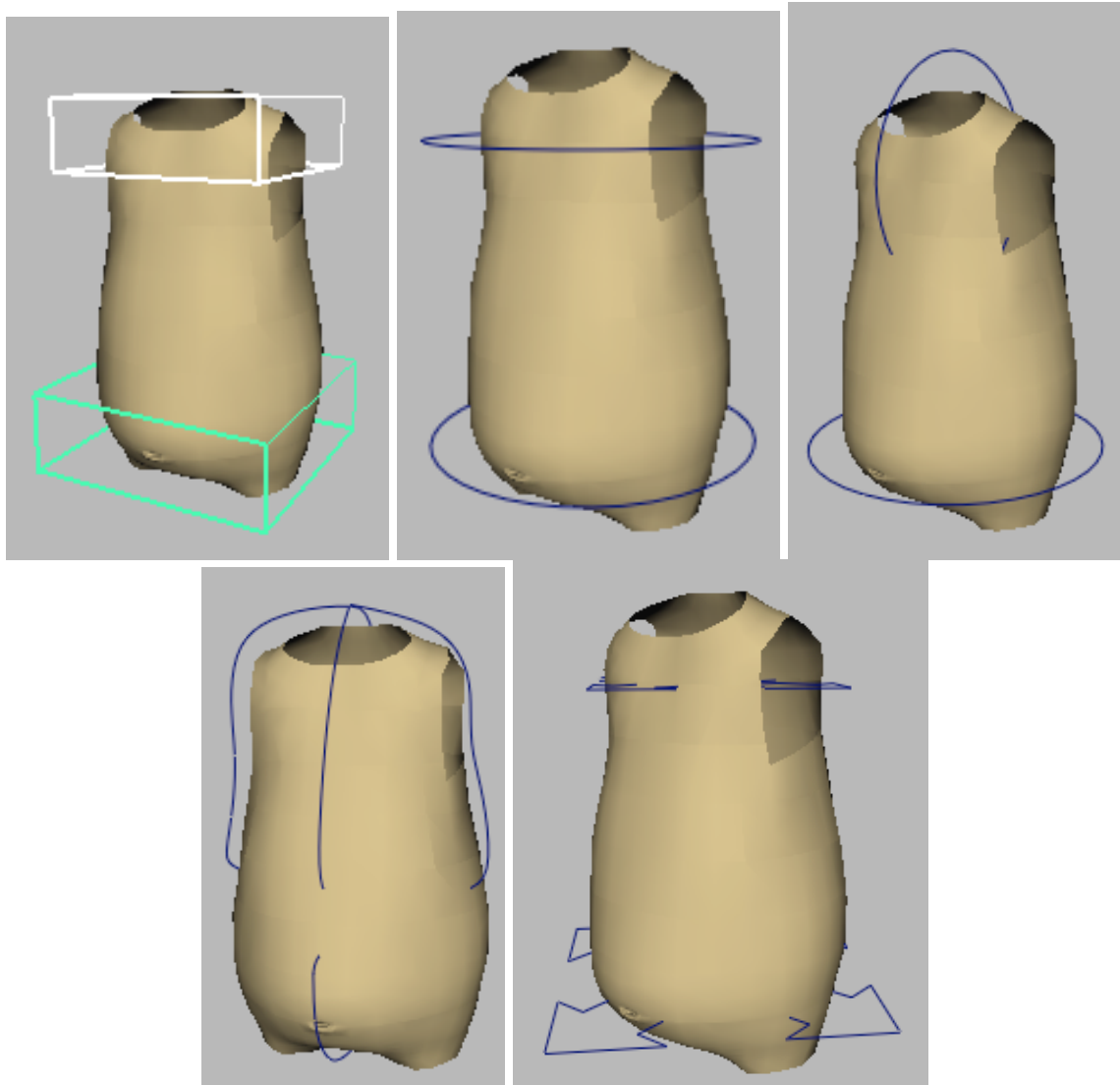


Figure 106- torso geometry parented to the appropriate joints.

#### Create the Hip and Shoulder Control

The first step in getting the joints to move the way we want is to create our control structure. This means we need to create two iconic control curves that will control the hips and shoulder.

We want something that will tell the animator “this is the shoulder” and “this is the hip”. Since these are the main controls the animator will be using to manipulate the torso, it’s important that they are clean, easy to use, and do not clutter up the interface too much.

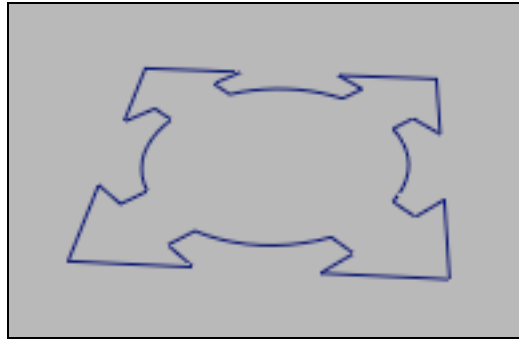


*Figure 107 - various control types which are available*

As you can see, there are limitless types of controls you can choose for the shoulders and hips. Personally, I prefer a more minimalist approach, one that doesn't detract from the performance, is easy to select, and isn't offending to the eye.

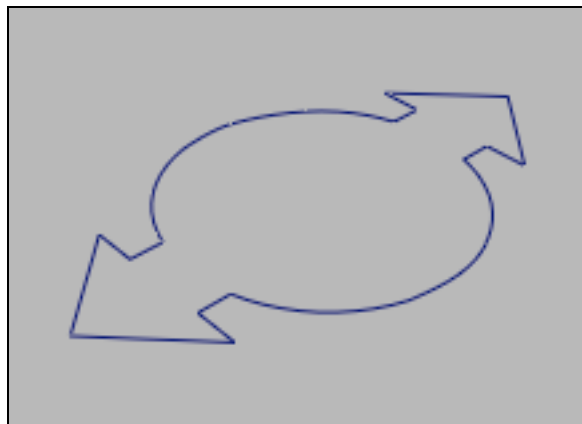
Do any of these meet those requirements? Not really. So which do we choose? Well, there's something I really like about the circle surrounding the body. It seems calm, and less "harsh" than the cubes. However, we can't tell what

direction the control is facing simply by looking at the circle. What if we combined a couple of controls and made our own circle/arrow?



*Figure 108 - a combination circle and arrow control*

We could also use maybe a similar one, but with only two arrows for the torso (remember, we have arms there and we'll need controls for them, so we don't want these arrows to interfere).



*Figure 109 - circle and only two arrows.*

Both of these files are available in the *example\_files/icons* directory.

#### 86. Import Controls

- Choose **File > Import**
- Navigate to **example\_files > icons**
- Import *circleArrow.ma*



- Import *circleArrowTwo.ma*

#### 87. Rename the controls

- Name the circleArrow control that was imported to **hip\_anim**
- Name the circleArrowTwo control to **shldr\_anim**

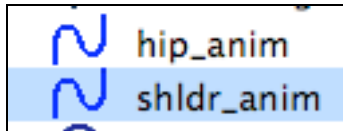


Figure 110- controls renamed

#### 88. Move the hip\_anim control

- Select **hip\_anim**
- Choose the **move** tool
- While holding down **v**, click and drag with the **Middle Mouse Button** and move it to the location of the **hip\_joint**



*Figure 111 - hip control moved to the hip joint*

#### 89. Move the shldr\_anim control

- Select **shldr\_anim**
- Choose the **move** tool.
- While holding down the **v** hotkey, click and drag with the **Middle Mouse Button** and move it to the location of **shldr\_joint**.
- Rotate the control 90 degrees in Z, and scale it down a bit to fit around the body.



*Figure 112 - shldr\_anim moved and rotated and scaled into position.*

Before continuing on, it's important to make sure that each of these controls are created in a way that's easy for the animator to reset if they want to set the character back to it's "default" position. If you look at the translation, rotation, and scale values for the controls, you can see that they're set at very odd values.

What we want to do is freeze the transformations of these objects, so the translation and rotation are at 0, and the scale is at 1.

#### **90. Freeze Transformations on hip\_anim and shldr\_anim**

- Select **hip\_anim** and **shldr\_anim**
- Choose **Modify > Freeze Transformations > Option Box**
- Make sure the transformation freezing is set for Translation, Rotation, and Scale

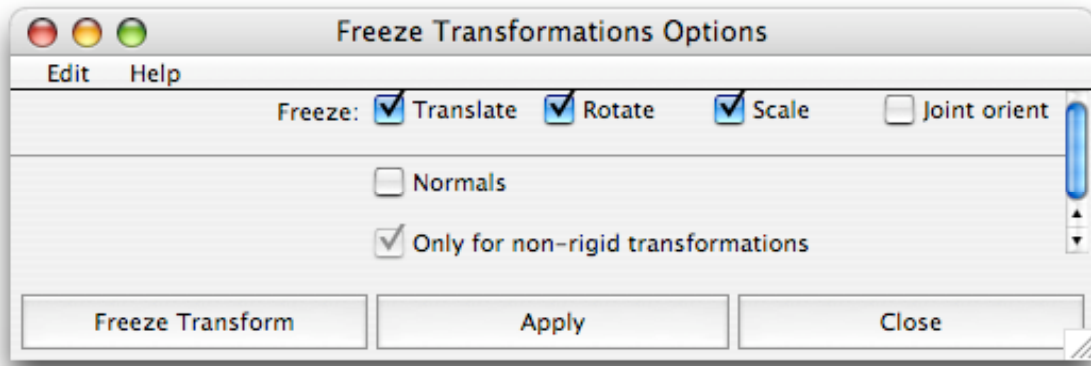


Figure 113 - Freeze Transformations

- Click **Freeze Transform**

#### 91. Constrain hip\_joint and shldr\_joint to the appropriate anim controls

- Select **hip\_anim** and **hip\_joint**
- Choose **Constrain > Parent**
- Select **shldr\_anim** and **shldr\_joint**
- Choose **Constrain > Parent**

#### 92. Set the correct Rotation Orders

- Select **hip\_anim**, **shldr\_anim**, **hip\_joint**, and **shldr\_joint**
- Choose the **Change Rotate Order On Selected Objects** button on the **Animator Friendly Rigging** shelf. This brings up the **js\_rotationOrderWin.mel<sup>xiii</sup>** script



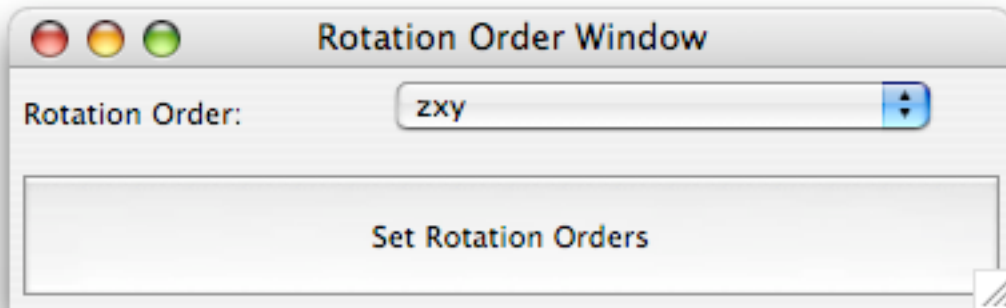


Figure 114 - Rotation Order Window - allows you to set the rotation order on multiple objects.

- Set the **Rotation Order** to **zxy**
- Click **Set Rotation Orders**

#### 93. Remove non-animated attributes

- Select **shldr\_anim** and **hip\_anim**
- In the **Channel Box** highlight **scaleX**, **scaleY**, **scaleZ** and **visibility**.
- Choose **RMB > Lock and Hide Selected**



*Figure 115 - hip and shldr controls ready to use.*

Now we have our two hip and shoulder animation controls ready to use. However, the back doesn't move with them! It's time to set up the spline IK control on the torso\_#\_joints. Remember the steps we used from our testing? We're going to do the exact same things here. This is why it's so handy to test before we actually create the rig. We know precisely what steps to take in order to get our rig to work how we want.

#### Add the Spline IK Control

##### 94. Create Spline IK

- Choose **Skeleton > IK Spline Handle Tool**
- In the **Outliner**, click on **torso\_1\_joint** and then hold down **Ctrl** (⌘ on Mac) and click on **torso\_7\_joint**. This will create the splineIK Control.

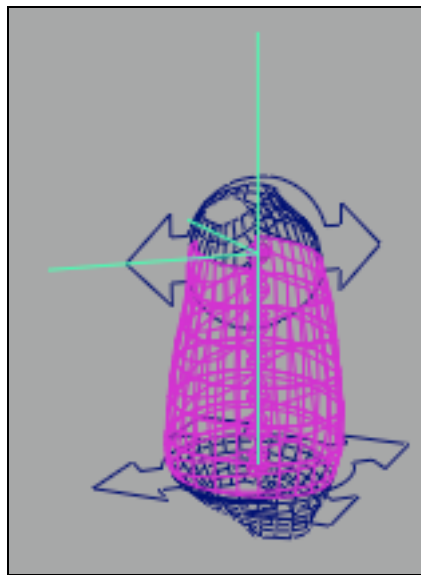


Figure 116- splineIK from torso\_1\_joint to torso\_7\_joint

##### 95. Rename the parts

- Rename **ikHandle1** to **back\_ikHandle**
- Rename **curve1** to **back\_curve**
- Rename **effector1** to **back\_effector**

##### 96. Attach the curve to the hip and shoulders

- Select **hip\_joint**, **shldr\_joint**, and **back\_curve**
- Choose **Skin > Bind Skin > Smooth Bind**

Notice that now when you move the hip and shoulder controls around, the back will deform a bit and try and line up.

But it doesn't stretch, so we have to add the stretching in.

#### 97. Add Squash and Stretch

- Select **back\_curve**
- Click on the **Create Stretchy Spline** button in the **Animator Rigging Shelf**. This will execute the **js\_createStretchSpline.mel<sup>xiii</sup>** script that will automatically create a stretchy spline *with* the control curve for managing the volume preservation.
- Move the hips and shoulders around to see what type of deformation you get.

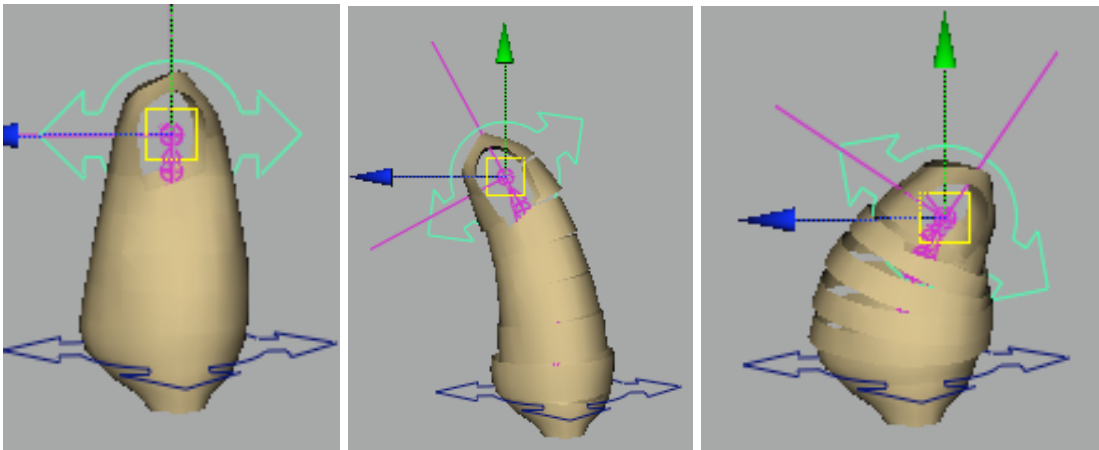


Figure 117 - deformation of torso

#### 98. Modify Squash/Stretch Curve

- Stretch the back out so it's very long (set the **shldr\_anim.t** to **0 2 0**)
- Select **back\_curve** – This is where the curve is that represents the back squash and stretch control.
- Open the **Graph Editor**
- Tweak the curve until you get the type of deformation that you like.



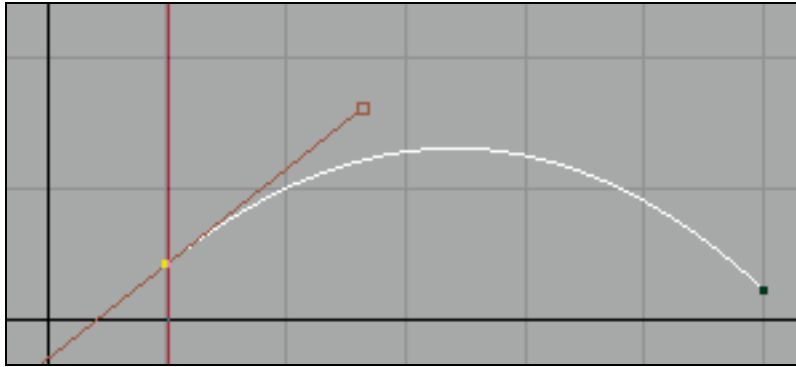


Figure 118 - Back curve tweaked to perfection

#### 99. Reset the shldr\_anim translation

- Set **shldr\_anim.t** to **0 0 0** to put it back into it's default position.

#### Add Rotation

The next step in our process of creating the back rig is to add the twisting into the back, based off of the hips and shoulder rotation.

#### 100. Open the Advanced Twist Options

- Select **back\_ikHandle**
- Open the **Attribute Editor**
- Open **Ik Solver Attributes**
- Scroll down and open **Advanced Twist Controls**
- Turn on **Enable Twist Controls**
- Set **World Up Type** to **Object Rotation Up**
- Set **Up-Axis** to **Negative Z**
- Set **Up Vector** to **0 0 -1**
- Set **Up Vector 2** to **0 0 -1**
- Set **World Up Object** to **hip\_anim**
- Set **World Up Object 2** to **shldr\_anim**

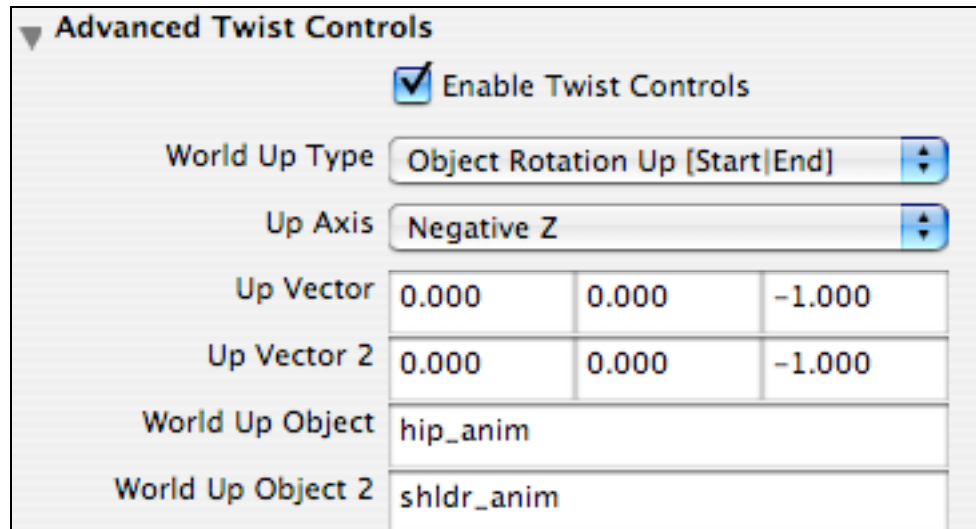


Figure 119 - Advanced Twist Controls Set

The IK back should now move around exactly how we want. Cool!

Let's double-check our animation control requirements and make sure that the hip\_anim and shldr\_anim meet all the requirements.

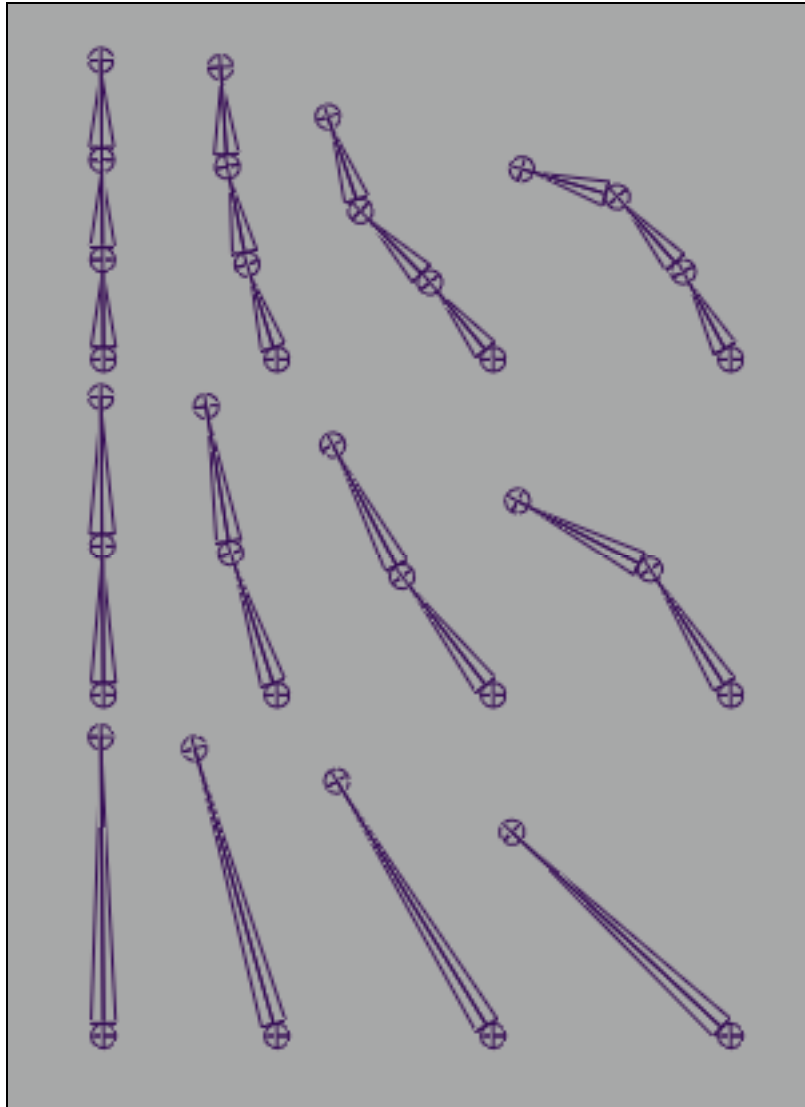
## Rig Requirements – CONTROL

- **Simple controls**
  - Yep, 2 controls, and they both move the way we want. Simple indeed!
- **Animation should be easily transferable.**
  - Animation can be transferred without difficulty.
- **Controls should be unique and make immediate sense.**
  - The controls are unique, and *do* make sense.
- **Controls should have the correct rotation orders.**
  - Rotation orders are correct!
- **Controls should be named correctly.**
  - We took care of this early on, and they're named correctly.
- **Only be able to set keyframes on controls we want animators using.**
  - Again, solid, true, and 100% working.

## Adding FK Back Control

Now we have to add our FK control for the back. For this I definitely recommend using as few joints as possible. You can usually get away with 3

joint segments. That will allow you to do things like bending the character over & still provide overlap.



*Figure 120 - notice how the top set of joints can add a lot more interesting overlap and follow-through to the body than the bottom two sets of joints.*

#### 101. Create the back structure

- Choose **Skeleton > Joint Tool**
- While holding down the **v** key for **point snap**, Click and drag near the **hip\_joint** to snap the joint at the hip.
- While holding down the **v** key for **point snap**, click and drag near the **shldr\_joint** to snap to the joint at the shoulder.
- Hit **enter** to complete your segment. This should create a joint segment going from the hip to the shoulder.

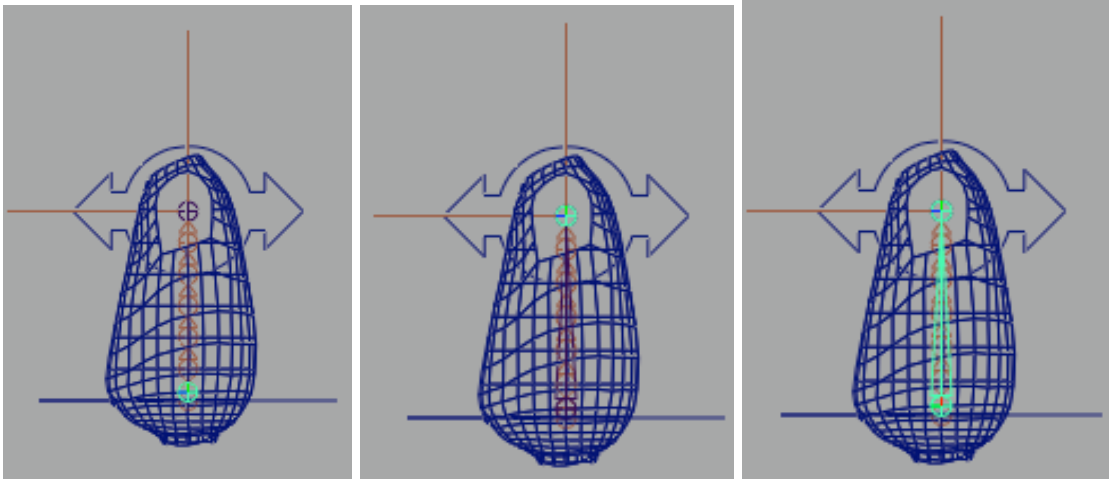


Figure 121 - Creating the joints for the FK skeleton

#### Orient the Joint correctly

Before we segment out the joint into it's various sections, it's *uber-important* to get the orientation of the joint correct. This isn't the **rotation order**, this is the **joint orient**. It's the thing that controls which *direction* the object rotates when you change a rotation value.

For example, look at the two joints in the following image:

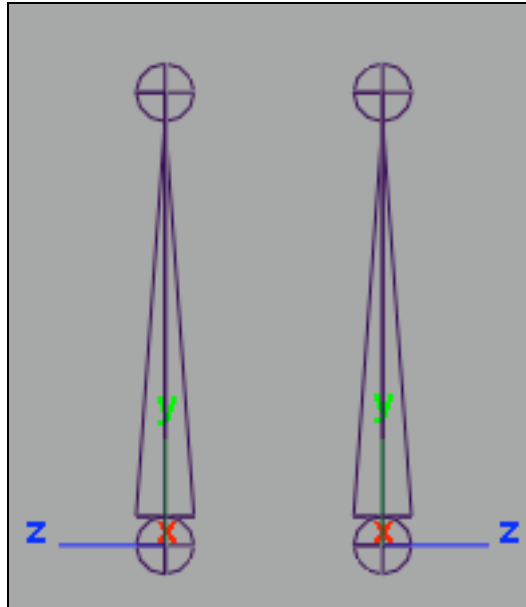


Figure 122 - two joints, both have rotation values of 0 0 0, but their joint orients are quite different!

If we were to rotate the joints in X, they would move in *opposite* directions.

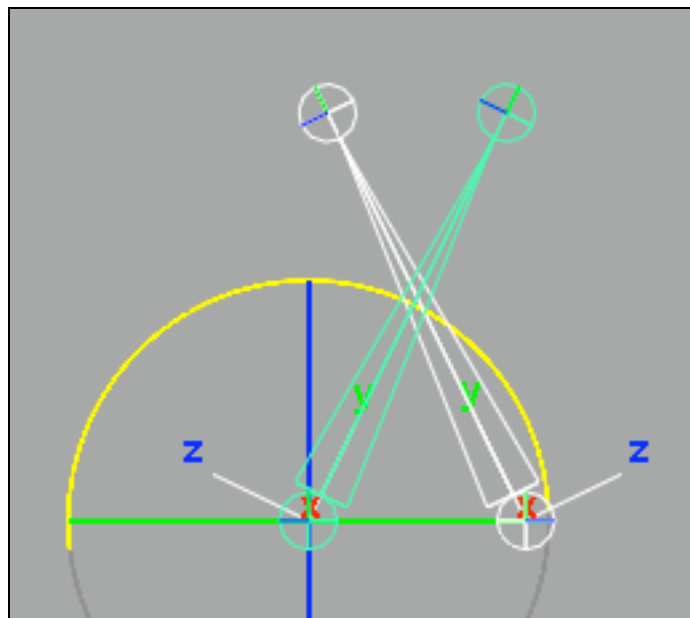


Figure 123 - both joints rotated 25 in X.

Let's look at the joint in perspective. Take a look at the rotation axis. Notice how X points in two different directions? The joints can *look* the same, but *behave* completely different, depending on their orientation.

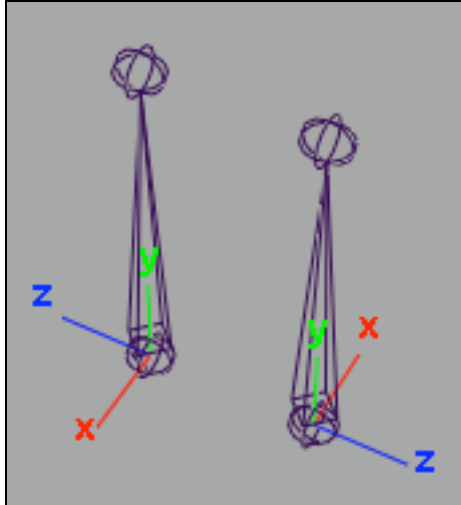


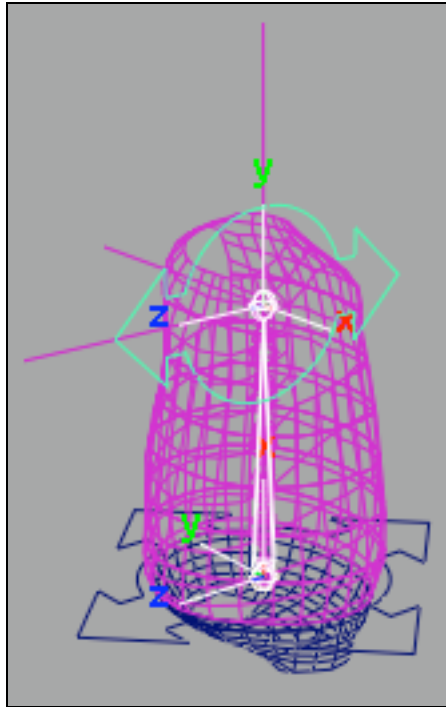
Figure 124 - The rotation order in 3d. Notice how X and Z both point in opposite directions!

Understanding the joint orientation is extremely important, because it can really help an animator when they're tweaking their animation. Imagine that rotating in X with a positive value bends a character *forward* with **shldr\_anim**, but bends the character *backwards* with the fk controls.

That just gives them extra things to think about, which they *should not have to do*. So we have to make sure our *FK* controls have the same orientation as our *IK* controls.

#### 102. Turn on the display of the rotation axis

- Select **shldr\_anim** and **joint1**
- Choose **Display > Transform Display > Local Rotation Axis**



*Figure 125 - Local Rotation Axis display for joint1 and shldr\_anim*

As you can see, the rotation axis do NOT line up for the two controls.  
This is baaaaaad!

#### 103. Orient the joint

- Select **joint1**
- Choose **Skeleton > Orient Joint > Option Box**

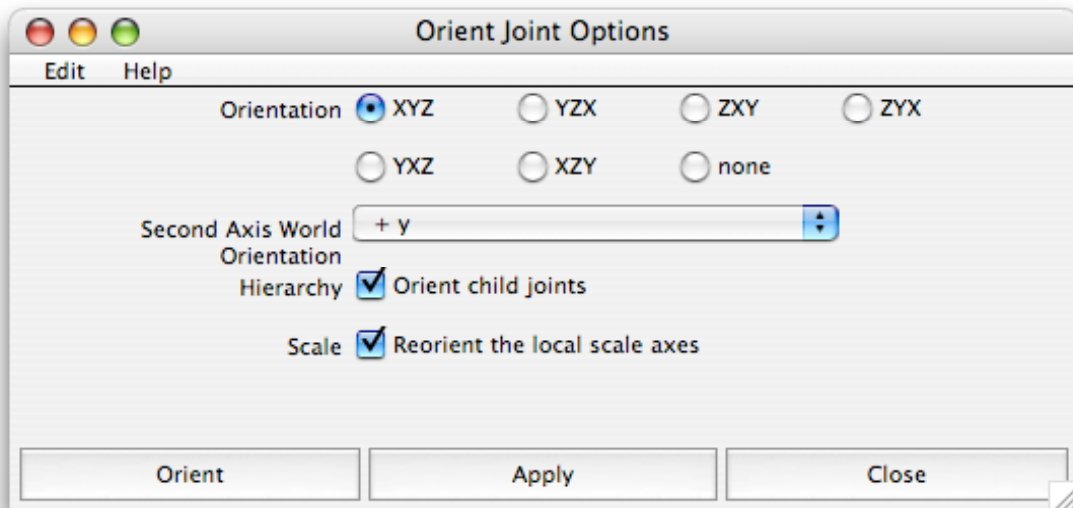


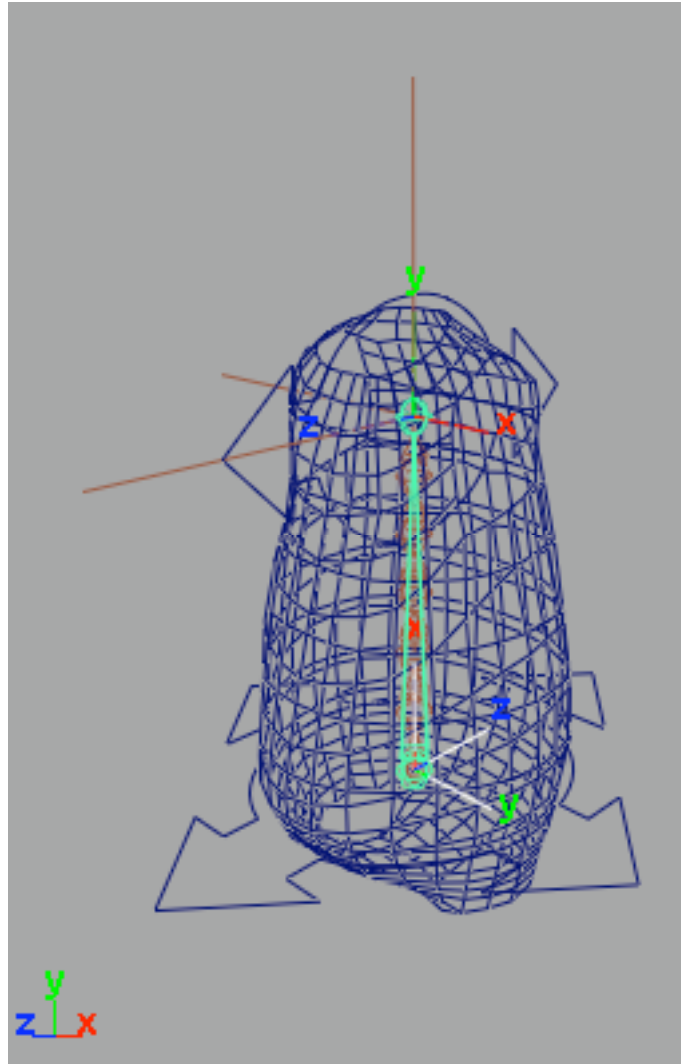
Figure 126- Orient Joint Option Box

So how does this work? First look at the **orientation** setting. The first axis (“x” if you have XYZ picked) is the axis that aims down the joint. The second axis (“y” in the previous example) orients *with* the world axis specified in **Second Axis World Orientation**. So with the default settings of XYZ and +Y, our joint aims the X axis down the joint, then tries to align the Y axis with the +Y in world. But it can’t do that, because the X axis is already facing up, so it picks another axis, which may or may not give us what we want.

Try switching the Second Axis World Orientation to +X. Hit Apply. Notice now that the Y axis on the joint orients with +X in world space.

But this still isn’t what we want for our joint. Let’s look at what the shldr\_anim axis is and what the world space axis is:

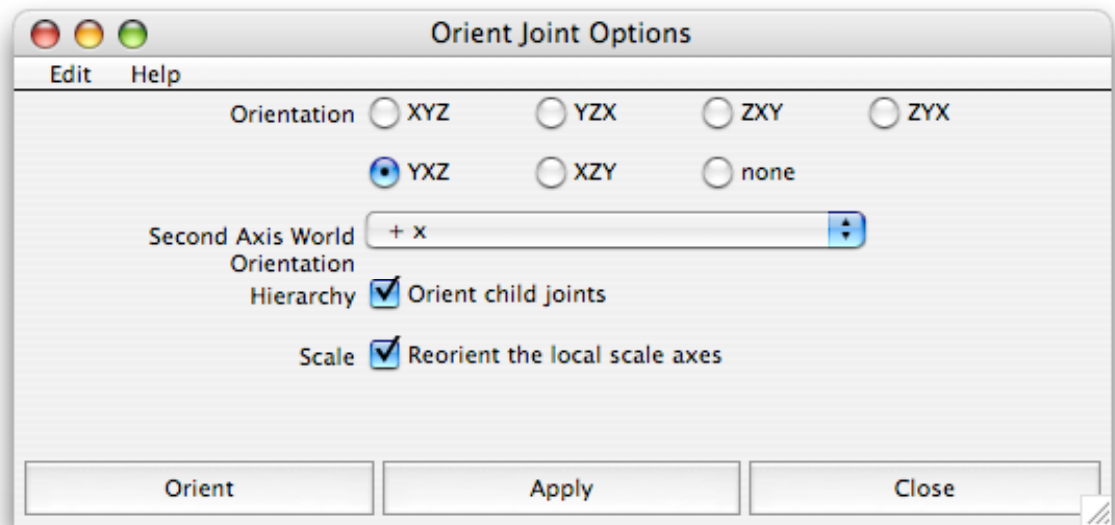




*Figure 127 - world space axis, shdlr\_anim axis, and joint1 axis*

We want to get the Y axis to aim up, the Z axis to point positiveZ, and the X axis to point positiveX.

To achieve this, there are two possible options for us.



OR

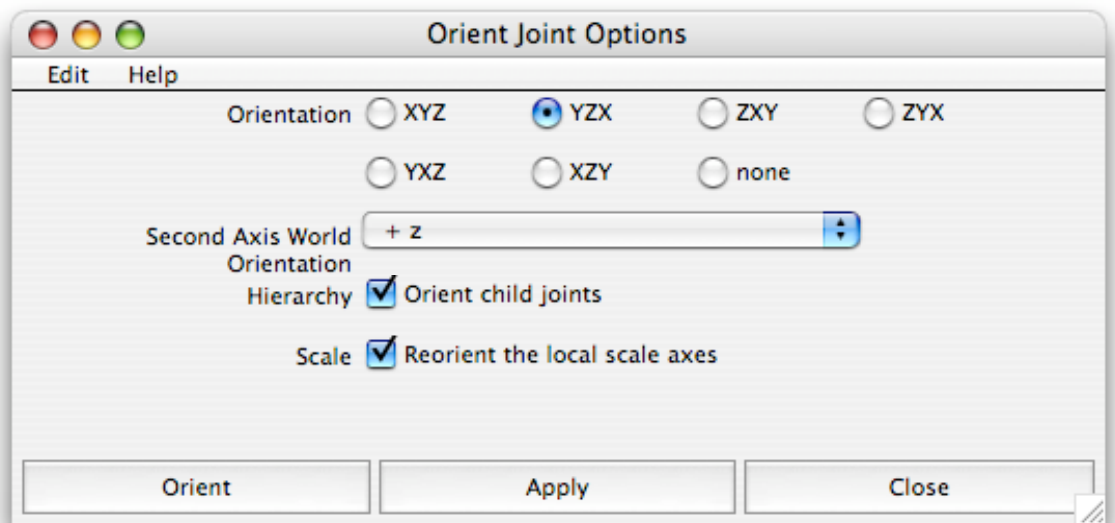


Figure 128 - orient joint axis options.

Either of these will work for what we want.. the Y axis will aim down the joint, and the X or Z axis will align with positive X or Z.

- Pick either **Orientation YZX and Second Axis World Orientation of +Z**, OR **Orientation YXZ and Second Axis World Orientation of +X**.
- Click **Orient**

- The rotation axis should now line up

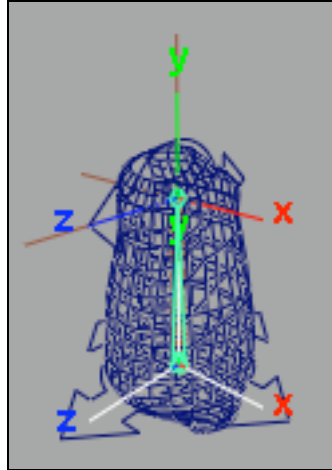


Figure 129 - Rotation Axis Lining Up

#### 104. Turn off the display or the rotation axis

- Select **shldr\_anim** and **joint1**
- Choose **Display > Transform Display > Local Rotation Axis**

#### 105. Change the rotation order

- The final step is to change the rotation order for the joint. Now we don't want the rotation order to be exactly the same as the one for the main controls. With Joints, you want the joint that is aiming down the bone to be the *last* rotation order available, so it twists.
- Select **joint1**
- Change the rotation order to **YZX**

#### Create the rest of the fk joints

#### 106. Segment the Joint

- Select **joint1**
- Click on the **Split Selected Joint** button on the **Animator Friendly Rigging** shelf.
- Set **Segments** to 3

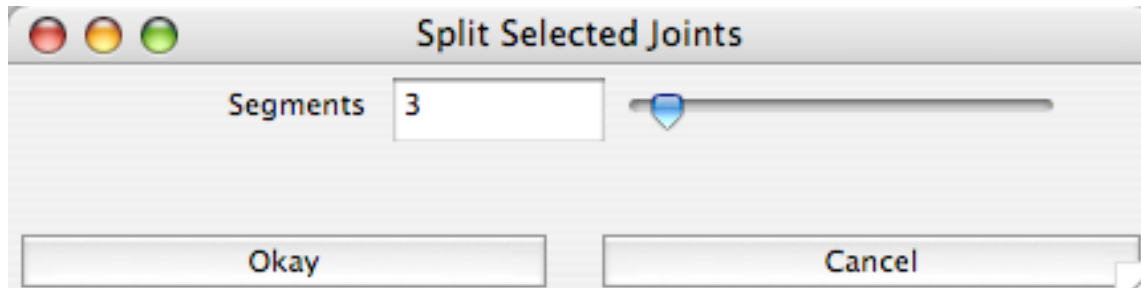


Figure 130- Split Selected Joints

#### 107. Rename each of the joints

- Rename **joint1** to **torso\_base\_joint**
- Rename **joint1\_seg\_1** to **torso\_1\_anim**
- Rename **joint1\_seg\_2** to **torso\_2\_anim**
- Rename **joint2** to **torso\_end\_joint**

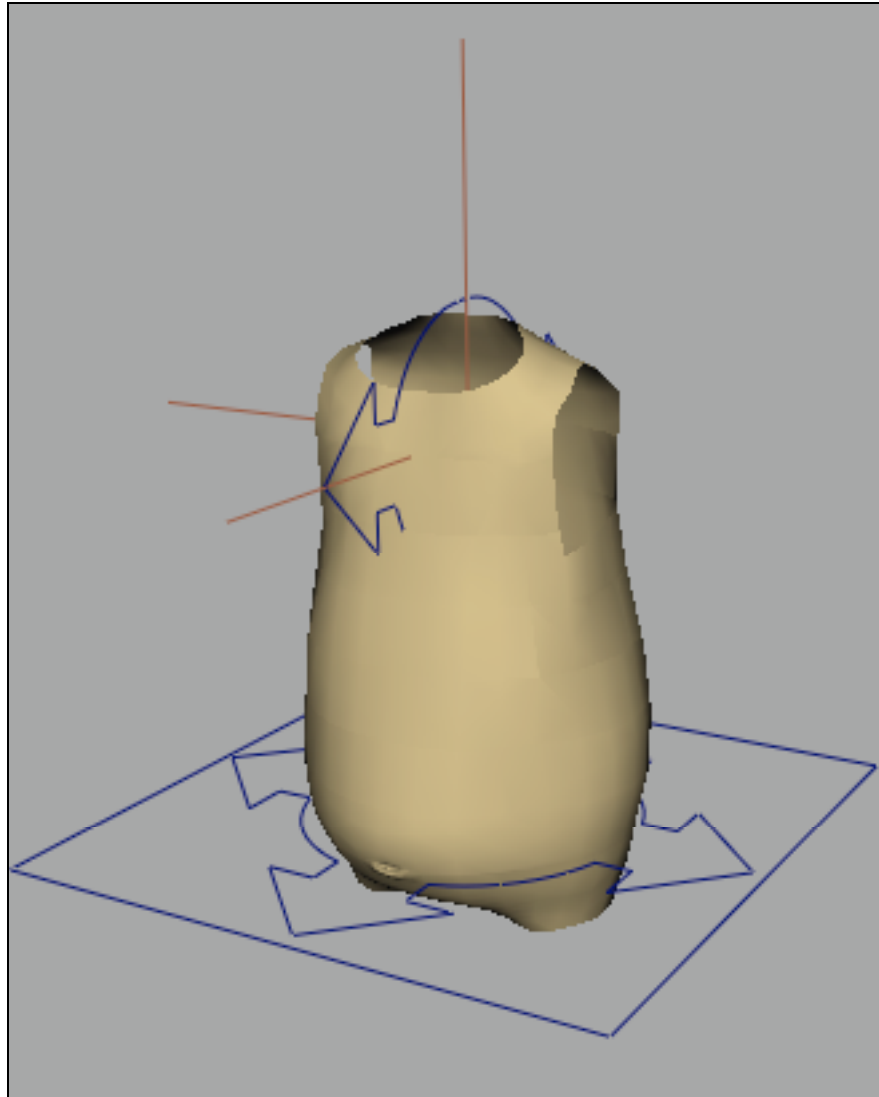
Why did we not name **joint1** to **torso\_1\_anim**? This is because we're going to be creating a *better* control to handle the base rotation of the back. We want one which will carry the entire back around with it, and will *look* like a full back control. This joint will be just extra baggage that the animator shouldn't have to animate. You can certainly allow animation on it if you wish, but I've found that by allowing it, you actually have two controls that do similar things (rotate the spine from the base), but work in different ways. We want our base spine control to rotate the same as our hips and shoulders with Y being the most important axis to be used for orientation, not the least important & used for twist.

#### Create a main body control

Just as we did with the hip and shoulder controls, we need a good main body control. This will be used to move both the FK and IK back around, and it's also where our pivot will be located.

Again, we want something relatively simple looking, something that won't distract the animator, but something that gives an idea as to what the control does.

One option is to use a square around the base of the control:



*Figure 131 - using a square as the main body control*

The nice thing about using this square is that it will fit neatly around the hip control, but both are easily selected.

#### **108. Import squareCurve.ma**

- Choose **File > Import**

- Navigate to **example\_files/icons/**
- Choose *squareCurve.ma*

#### 109. Rename square\_anim

- Rename **square\_anim** to **body\_anim**

#### 110. Move body\_anim to the base of the spine

- Select **body\_anim**
- Choose the **Move** tool
- Holding down the **v** hotkey for point snap, click and drag with the **Middle Mouse Button** near **hip\_joint** to snap **body\_anim** to the position around the hip.

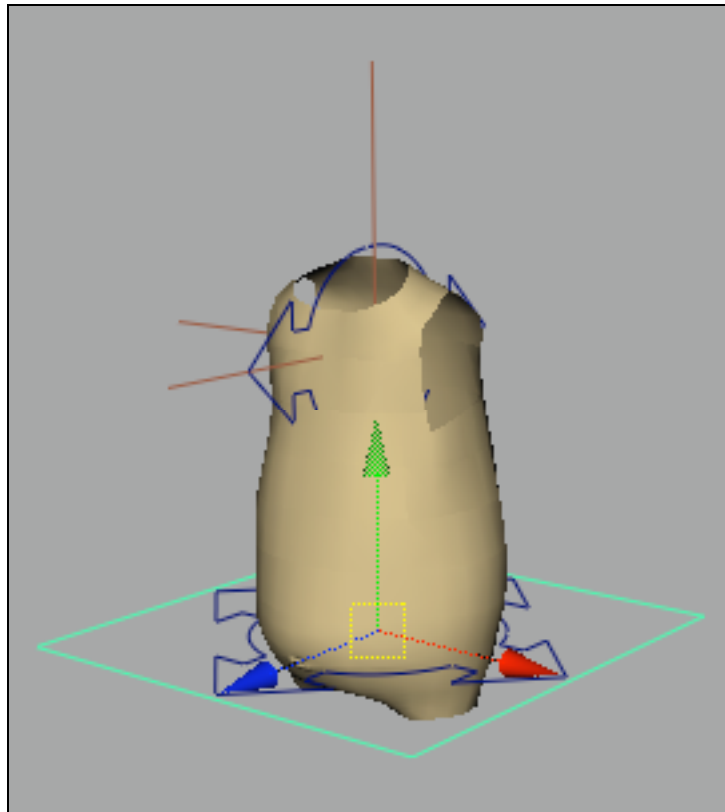
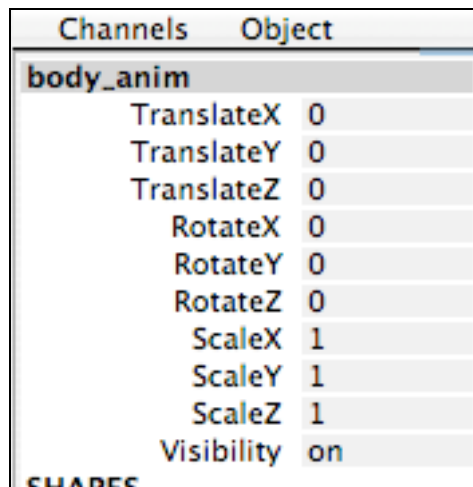


Figure 132 - moving the body position

#### 111. Freeze Transformations on body\_anim

- Choose **Modify > Freeze Transformations**
- Make sure that the translations and rotations are 0 0 0, and the scale is 1 1 1



Channels	Object
<b>body_anim</b>	
TranslateX	0
TranslateY	0
TranslateZ	0
RotateX	0
RotateY	0
RotateZ	0
ScaleX	1
ScaleY	1
ScaleZ	1
Visibility	on

Figure 133 - body\_anim after Freeze Transformations is applied

#### 112. Parent torso\_base\_joint to body\_anim

- Select **torso\_base\_joint**
- Shift + Select **body\_anim**
- Hit the **p** hotkey to parent **torso\_base\_joint** to **body\_anim**

#### 113. Parent hip\_anim to torso\_base\_joint

- Select **hip\_anim**
- Hit **CTRL+G** to group it to itself. *Note: This is necessary to ensure that hip\_anim's translations and rotations stay at 0 0 0 when it's parented.*
- Rename **group1** to **hip\_grp**
- Select **hip\_grp** and then **torso\_base\_joint**
- Hit the **p** hotkey to parent **hip\_grp** to **torso\_base\_joint**

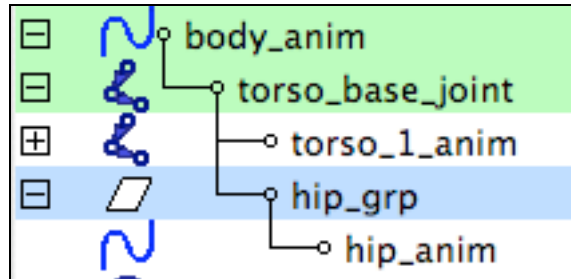


Figure 134 - hip\_grp parented to torso\_base\_joint

#### 114. Parent shldr\_anim to torso\_end\_joint

- Select **shldr\_anim**
- Hit **CTRL+G** to group it to itself
- Rename **group1** to **shldr\_grp**
- Select **shldr\_grp** and then **torso\_end\_joint**
- Hit the **p** hotkey to parent **shldr\_grp** to **torso\_end\_joint**

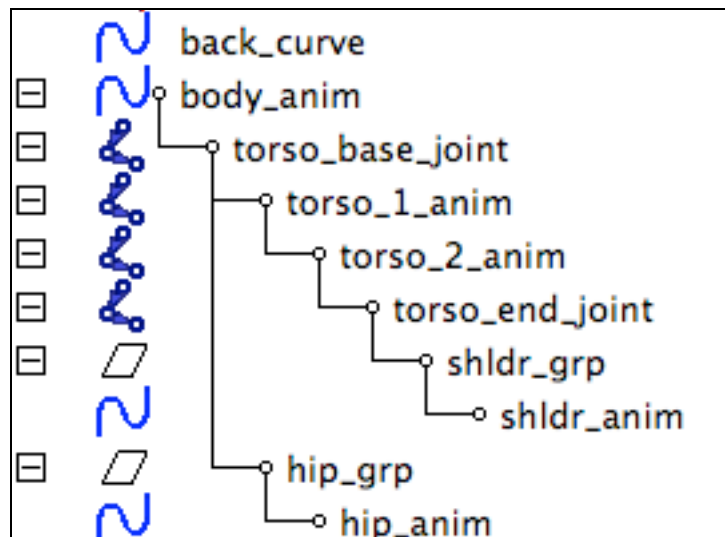


Figure 135 - shldr\_grp parented to torso\_end\_joint

#### 115. Change the rotation order of body\_anim

- Select **body\_anim**
- Click **Change Rotation Order on Selected Objects** Shelf Button.



- Change the rotation order to **zxy** (so it matches the hip\_anim and shldr\_anim controls).

#### 116. Add a pivot for body\_anim

- Select **body\_anim**
- Hit the **Pivot Create** button in the **Animator Friendly Rigging** shelf
- Hide the pivot by now clicking the **Pivot: Toggle Pivot Visibility** button on the **Animator Friendly Rigging** shelf



#### Set The Attributes That Are Allowed Animatable.

Follow the following chart for setting attributes non-keyable and locked

Objects	Attributes to hide and lock
body_anim	scaleX, scaleY, scaleZ, visibility
torso_1_anim, torso_2_anim	translateX, translateY, translateZ, scaleX, scaleY, scaleZ, visibility
torso_base_joint, torso_end_joint	translateX, translateY, translateZ, rotateX, rotateY, rotateZ, scaleX, scaleY, scaleZ, visibility
hip_grp, shldr_grp	translateX, translateY, translateZ, rotateX, rotateY, rotateZ, scaleX, scaleY, scaleZ, visibility

#### Rig Requirements – ALL

Let's take a look at the rig requirements now and finish up our torso rig.

- **Only be able to select what the animators can use to animate.**
- **Clean outliner when finished.**
- **Can move the rig to any position and orientation and have it work.**



## Master Classes

# Instructor Notes – Animator Friendly Rigging Part II

Jason Schleifer

**Only allow selection on what the animator should be able to select**

### 117. Hide anything that you don't want selected

- Select **back\_ikHandle** and **back\_curve**
- Hide them by hitting **ctrl+h**

### 118. Create Display layers for the controls and the things you don't want the animator to select.

- Create a Display Layer called **mid\_anim\_layer**
- Create a Display Layer called **doNotTouch\_layer**

### 119. Add some items to DoNotTouch

- Select **torso\_1\_joint**, **hip\_joint**, **shldr\_joint**, **back\_ikHandle**, **back\_curve**
- Click with the **Right Mouse Button** over **doNotTouch\_layer** and choose **Add Selected Objects**

### 120. Load the Display Layer Relationship Panel

- Choose **Layers > Membership**

### 121. Assign Some objects to DoNotTouch

- Select **doNotTouch\_layer**
- Click on **torso\_base\_joint** and **torso\_end\_joint**

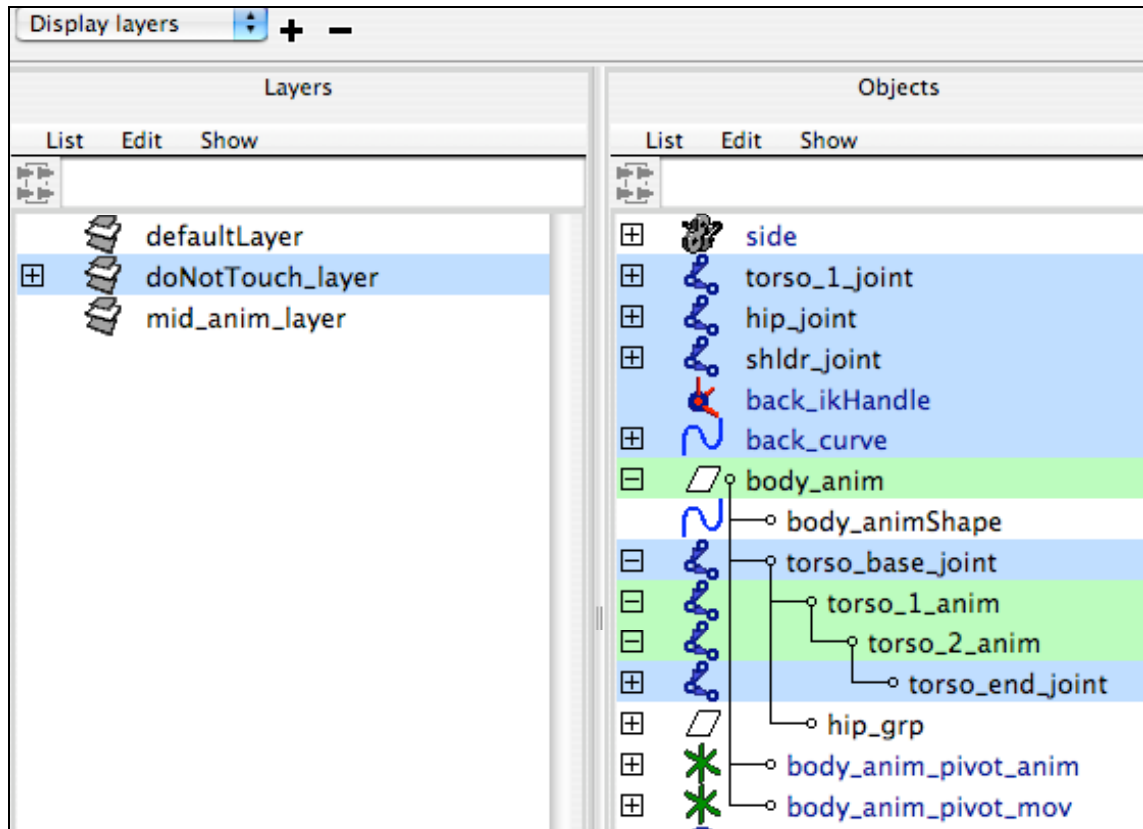


Figure 136- All the assigned objects in doNotTouch\_layer

#### 122. Assign the animation controls to mid\_anim\_layer

- Click on **mid\_anim\_layer**
- Select **body\_anim**, **torso\_1\_anim**, **torso\_2\_anim**, **shldr\_anim**, **hip\_anim**, **body\_anim\_pivot\_anim**, and **body\_anim\_pivot\_mov**

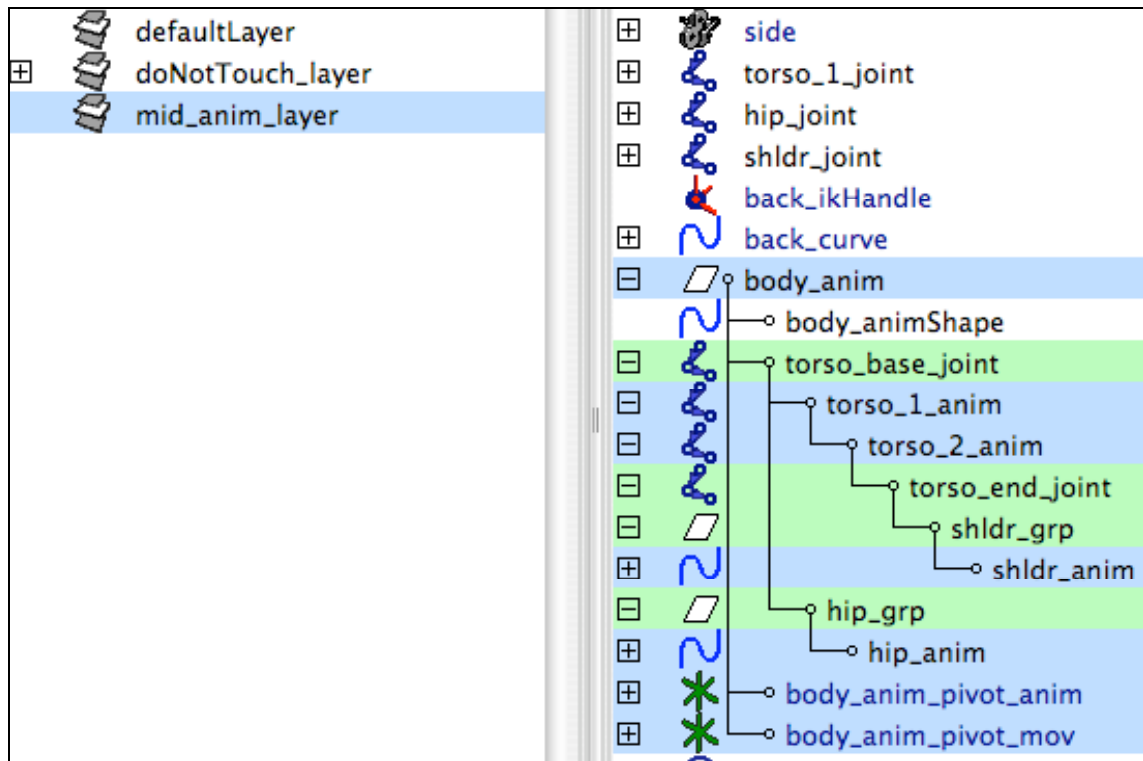


Figure 137 - All the animation controls in mid\_anim\_layer

#### 123. Set doNotTouch\_layer to Reference

- Click on the empty box next to the  in the Layer Editor until it looks like an R

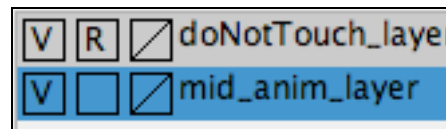


Figure 138 - doNotTouch\_layer set to Reference mode

Now the animator shouldn't be able to select anything that you don't want them to touch.

#### Clean Outliner

Now we have to clean the outliner and make sure it's fit for use within a shot. This means that everything should be clearly labeled.

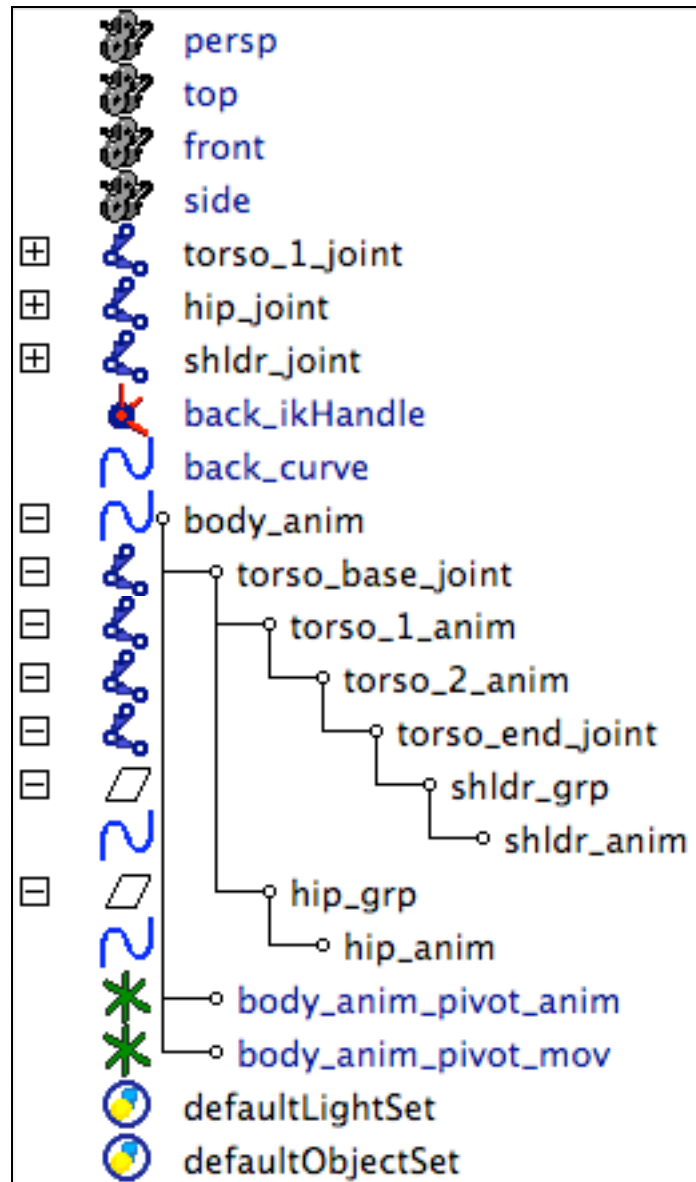


Figure 139 - Outliner Pre Cleaning

#### 124. Group body\_anim to itself to form a group called anim\_grp

- Select **body\_anim**
- Use the **CTRL+g** hotkey to group **body\_anim** to itself.
- Rename **group1** to **anim\_grp**

#### 125. Group all the other nodes together

- Select **torso\_1\_joint**, **hip\_joint**, **shldr\_joint**, **back\_ikHandle**, **back\_curve**
- Use the **CTRL+g** hotkey to group them all together.
- Rename **group1** to **doNotTouch\_grp**

#### 126. Lock the attributes on doNotTouch\_grp

- In the Channel Box, select all the attributes.
- RMB > Lock Selected

#### 127. Group everything under an all\_anim

- In the outliner, select **body\_anim** and **doNotTouch\_grp**
- Use the **CTRL+g** hotkey to group them together.
- Rename **group1** to **all\_anim**

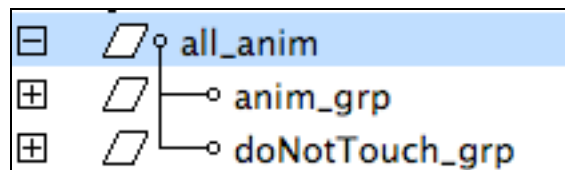


Figure 140- all\_anim

Now notice what happens if you try and move all\_anim.

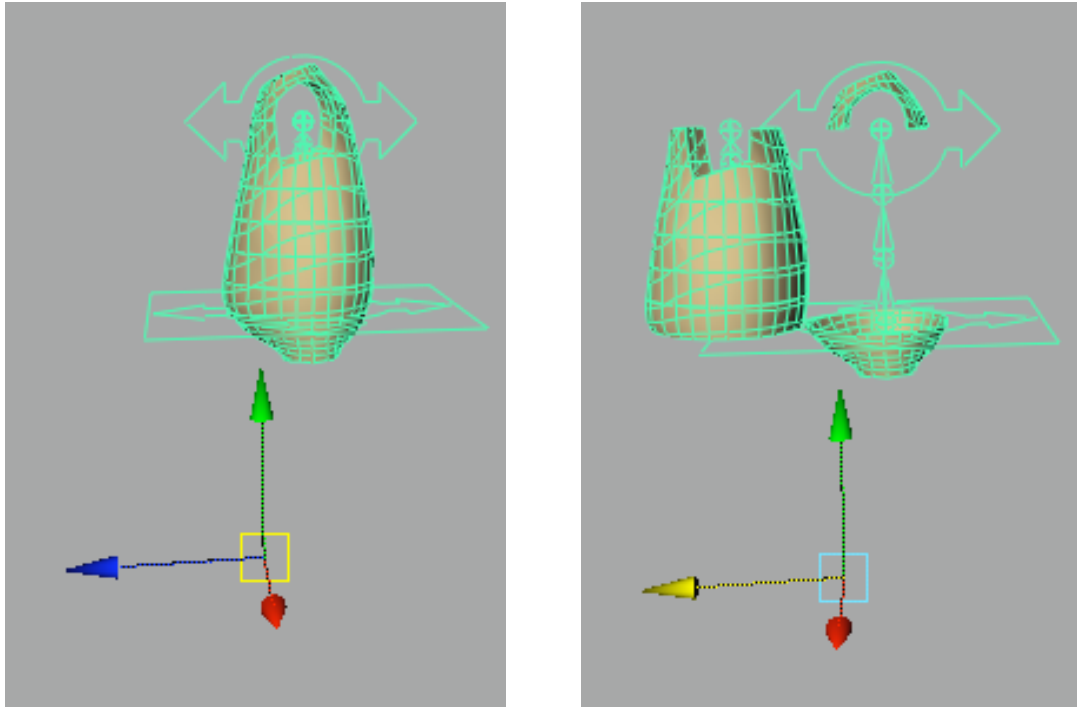


Figure 141 - Body "exploding" when moving anim\_all

This is because there's something called "double-transformation" happening. The **back\_curve** is being controlled by the smooth skinning that we did earlier. But it's also a child of **all\_anim**. When **all\_anim** moves, it moves the joints which control **back\_curve**, but then it *also* moves **back\_curve**.. giving it an appearance of moving *twice*. Hence the name "double-transformation".

We can remove this effect, simply by turning off the ability for the **back\_curve** to inherit any transformation from its parent joint.

#### 128. Turn off inherits transform on back\_curve

- Select **back\_curve**
- Open the **Attribute Editor**
- At the bottom of the **Transform Attributes** section, turn *off* **Inherits Transform**

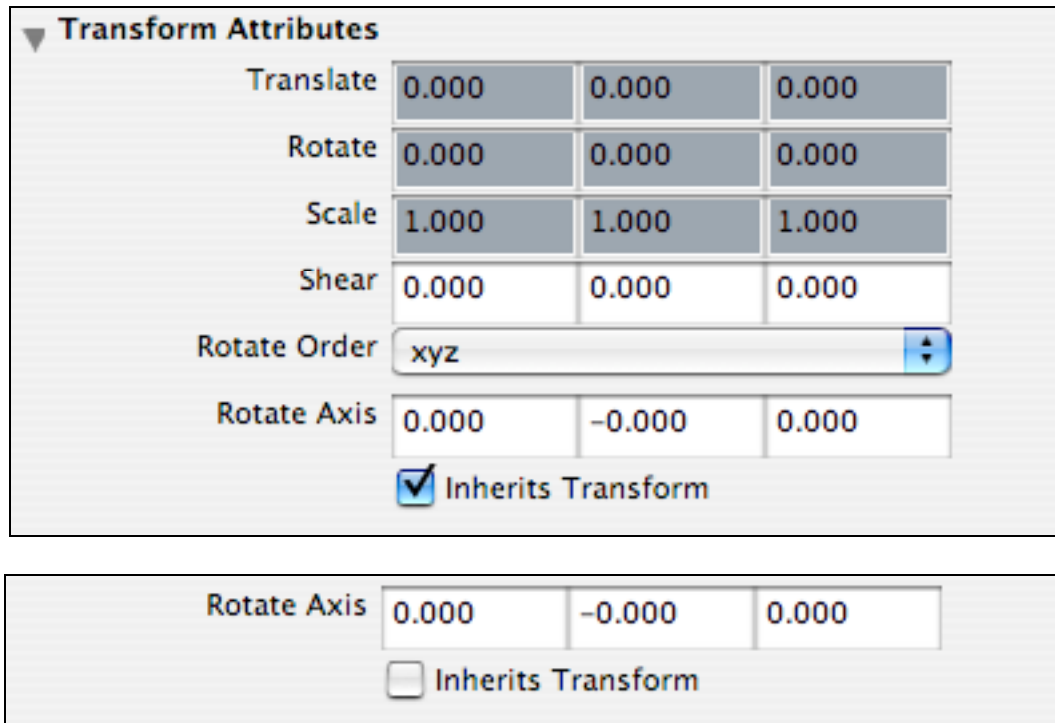


Figure 142 - Inherits Transform being turned off

#### Enable Scaling of the Character

Quite frequently it will be important to scale the character just a *little* bit to help integrate it into a shot. Let's test our all\_anim control and see how it handles scale.

#### 129. Scale all\_anim

- Select **all\_anim**
- Hit **s** to go to the scale tool.
- Scale the character uniformly.



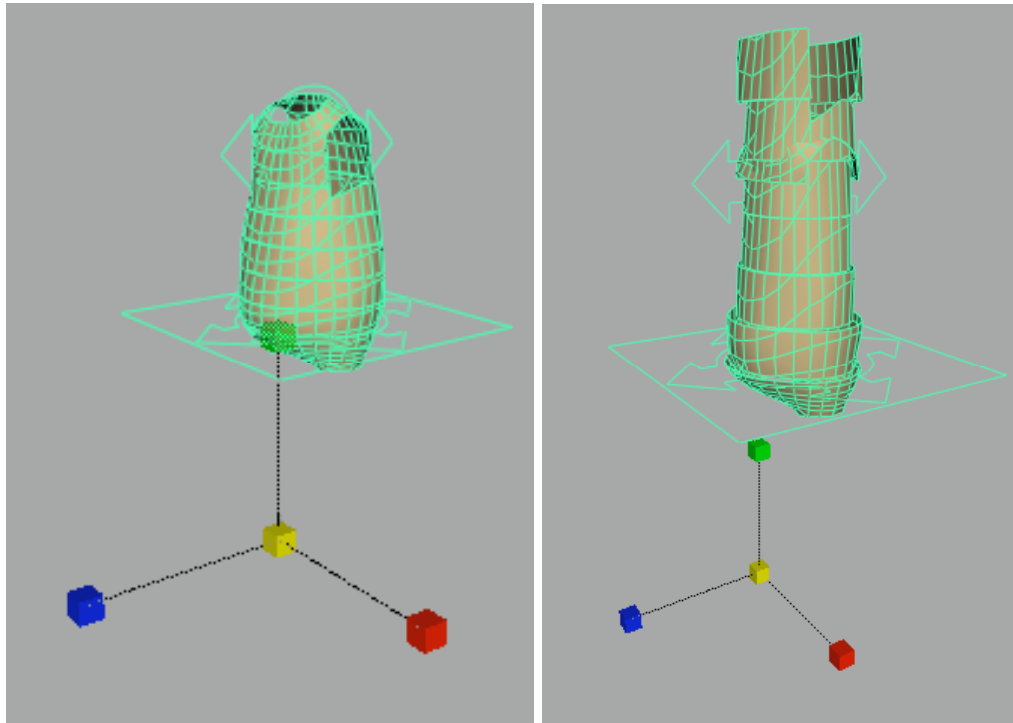


Figure 143 - body scaled uniformly. Notice the bad artifacts of the scale!

You will notice that the back stretches longer than the hips and shoulders are moved. This is because we have an expression driving the length of the back to match the distance of the hips and shoulders. Unfortunately, it's stretching too far simply because the distance of the curve is increasing by the amount the rig is being scaled, but we're not taking that into account in the expression.

#### 130. Fix the back scale expression

- Open up the expression editor choosing **Window > Animation Editors > Expression Editor**
- Choose **Select Filter > by Expression Name**
- Choose **back\_curve\_expr**
- At the top of the expression, you will see a line that looks like this:

```
$scale = curveInfo1.normalizedScale;
```

- Replace it with a line that looks like:

```
$scale = curveInfo1.normalizedScale/all_anim.scaleY;
```

Autodesk® Maya® Master Classes – Instructor Notes

SIGGRAPH™ 2006



## Master Classes

# Instructor Notes – Animator Friendly Rigging Part II

Jason Schleifer

Notice that the scale is based off of the scaleY attribute. We want the character to be scaled uniformly, so we have to connect the scaleY attribute to scaleX and scaleZ.

### 131. Connect scaleX and scaleZ to scaleY

- Select **all\_anim**
- Open the **Connection Editor** by choosing **Window > General Editors > Connection Editor**
- Click **Reload Left** and **Reload Right** to load **all\_anim** into both sides.
- Make sure **show non-keyable** is off on both sides.
- On the **Left side**, click **scaleY**
- On the **Right side** click **scaleX** and **scaleZ**

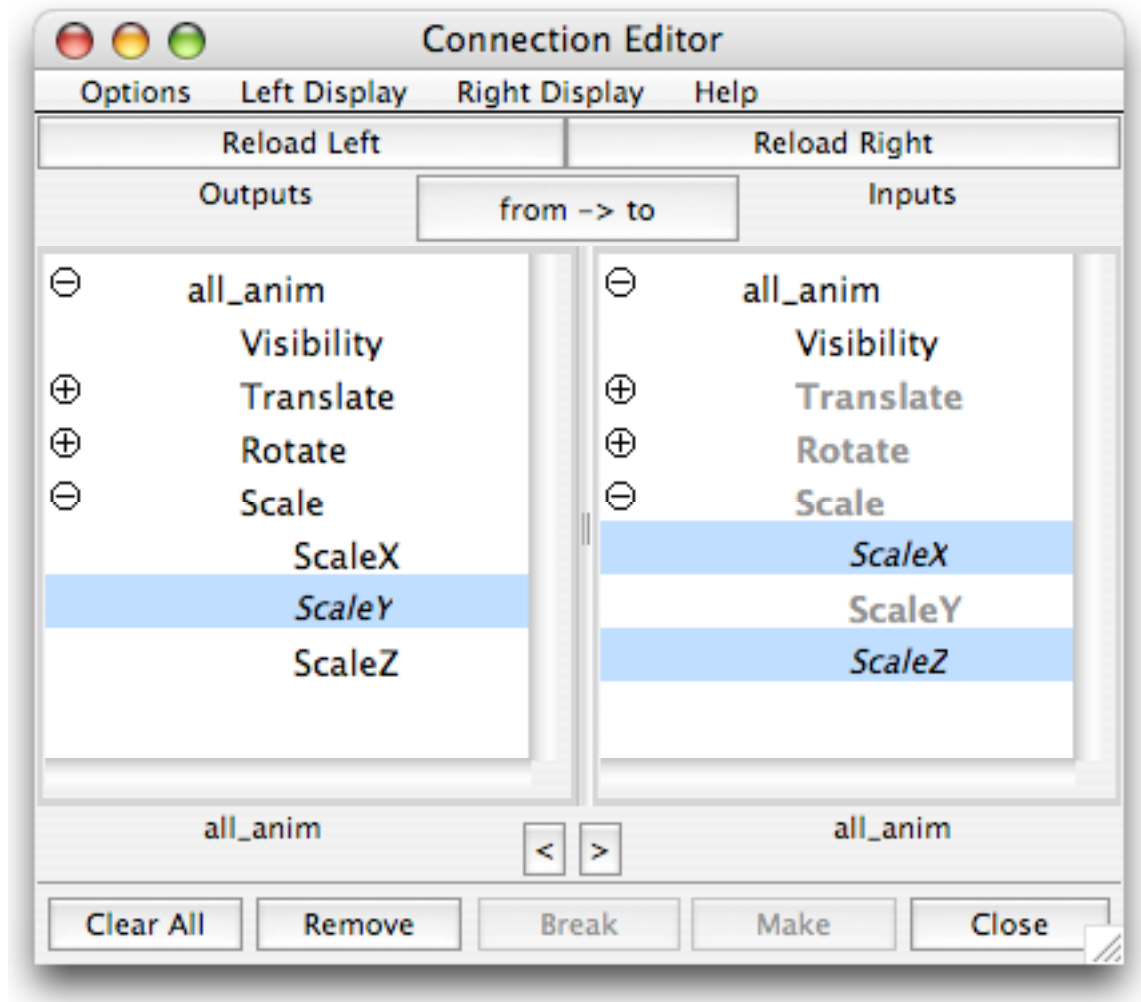


Figure 144 - connecting scaleY to scaleX and scaleZ

#### 132. Hide scaleX and scaleZ in the channel box

- Select **all\_anim**
- In the **Channel Box**, select **ScaleX** and **ScaleZ**
- Use **RMB > Lock and Hide Selected**

#### 133. Rename scaleY to something that makes sense to the animator

- Open the **script editor**. We're going to use the **aliasAttr** command to make an alias for scaleY

Autodesk® Maya® Master Classes – Instructor Notes

SIGGRAPH™ 2006

- Enter the following command in the script editor:

```
aliasAttr globalScale all_anim.scaleY;
```

Channels	Object
all_anim	
TranslateX	0
TranslateY	0
TranslateZ	0
RotateX	0
RotateY	0
RotateZ	0
globalScale	1
Visibility	on
OUTPUTS	
back_curve_expr	

*Figure 145 - scaleY is now aliased as globalScale. This makes much more sense to the animator, so when they see globalScale, they understand exactly what it means.*

Our rig is now clean, can be moved anywhere, and the animator can only select what we want them to. However, it's still not as clean and tidy looking as it could be.

Notice all the joints that are going up and down the back.

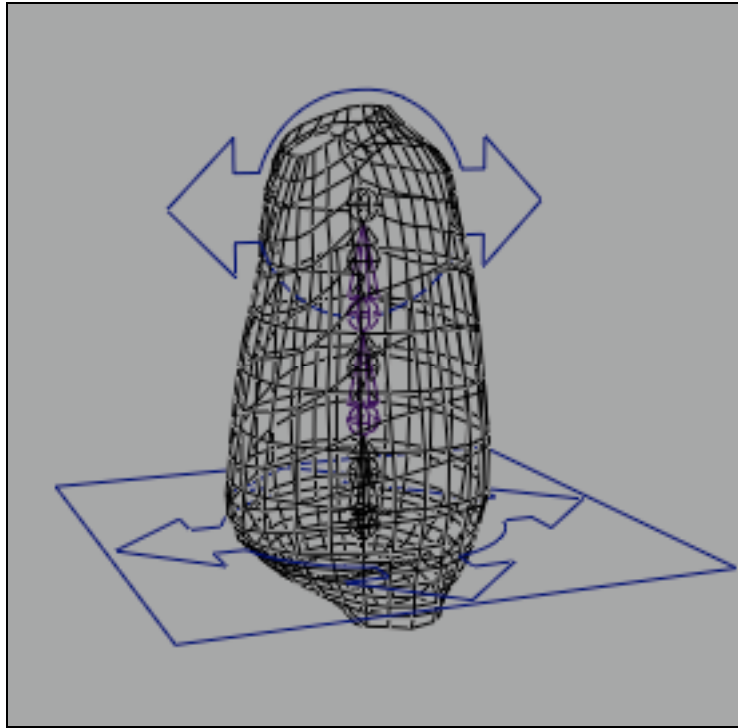


Figure 146- visible joints

With a clean rig, we shouldn't be able to see them, since they can be distracting, and really aren't the most beautiful things to look at.

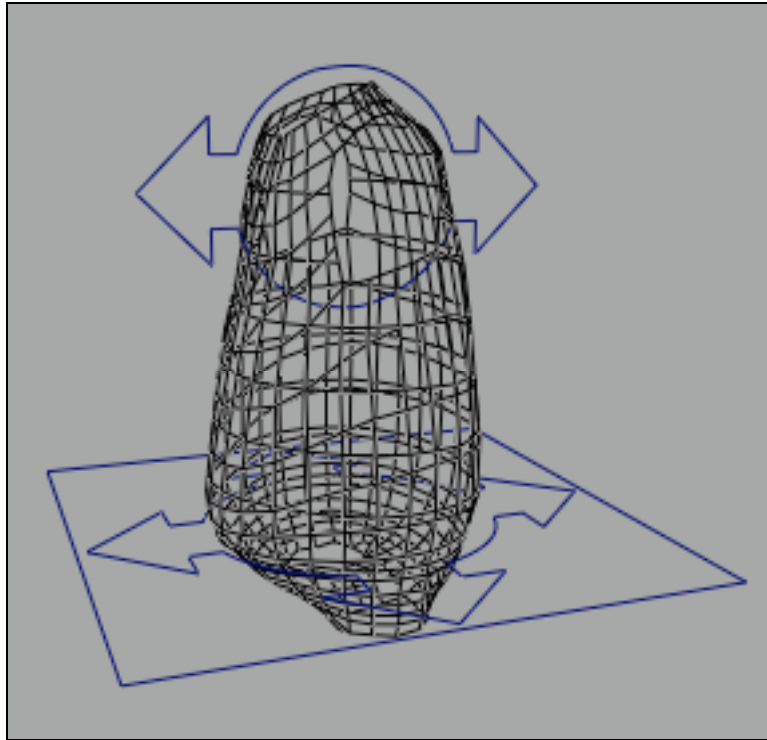
Unfortunately, we can't just hide them because the geometry is parented to the joints (this is one reason why sometimes it's nice to constrain things instead of parent).

We *can*, on the other hand, turn off the display in the panel.

#### 134. Turn off joint display in the panel

- Choose **Show > Joints** to turn off display of all joints.

Unfortunately, this will not hide joints for every panel, nor will it ensure that your animators turn off the display of joints.



*Figure 147 - Joint Display turned off*

If you can convince your animators to turn joints off, however, they will find playback much faster, and the scene will appear cleaner.

Unfortunately, with our joints turned off, we can't select our FK joints for manipulating! So the next step is to create some fk controls that we can use to manipulate the joints, while still keeping them invisible.

#### **Add FK control curves for the torso\_1\_anim and torso\_2\_anim joints**

##### **135. Create a nurbsCircle**

- Choose **Create > Nurbs Primitives > Circle**

##### **136. Parent nurbsCircleShape1 to torso\_1\_anim**

- Enter the following command in the script editor:

```
parent -add -shape nurbsCircleShape1 torso_1_anim;
```

This will add an instance of the shape of the circle to the joint.. basically *giving the joint a nurbsCircle as a display option!*

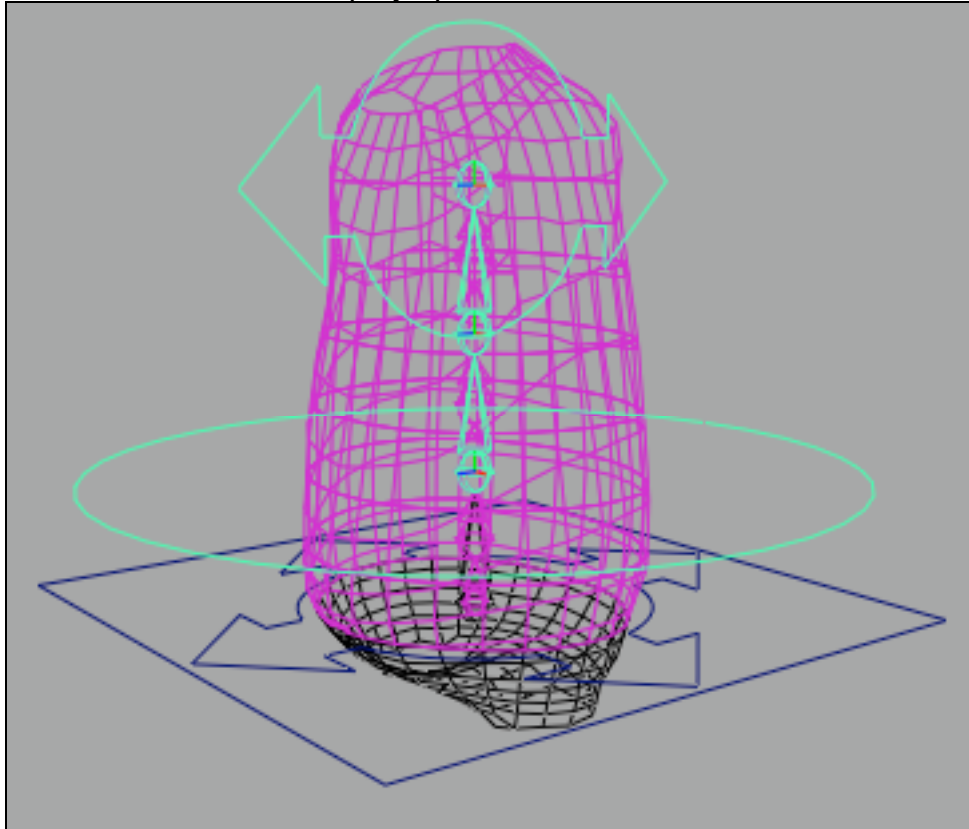


Figure 148 - nurbsCircle added as a display to the joint.

#### 137. Scale the circle

- Select **torso\_1\_anim**
- Hit **F8** to switch to component mode
- Select the cvs on the circle
- Scale them until the circle fits snugly around the torso

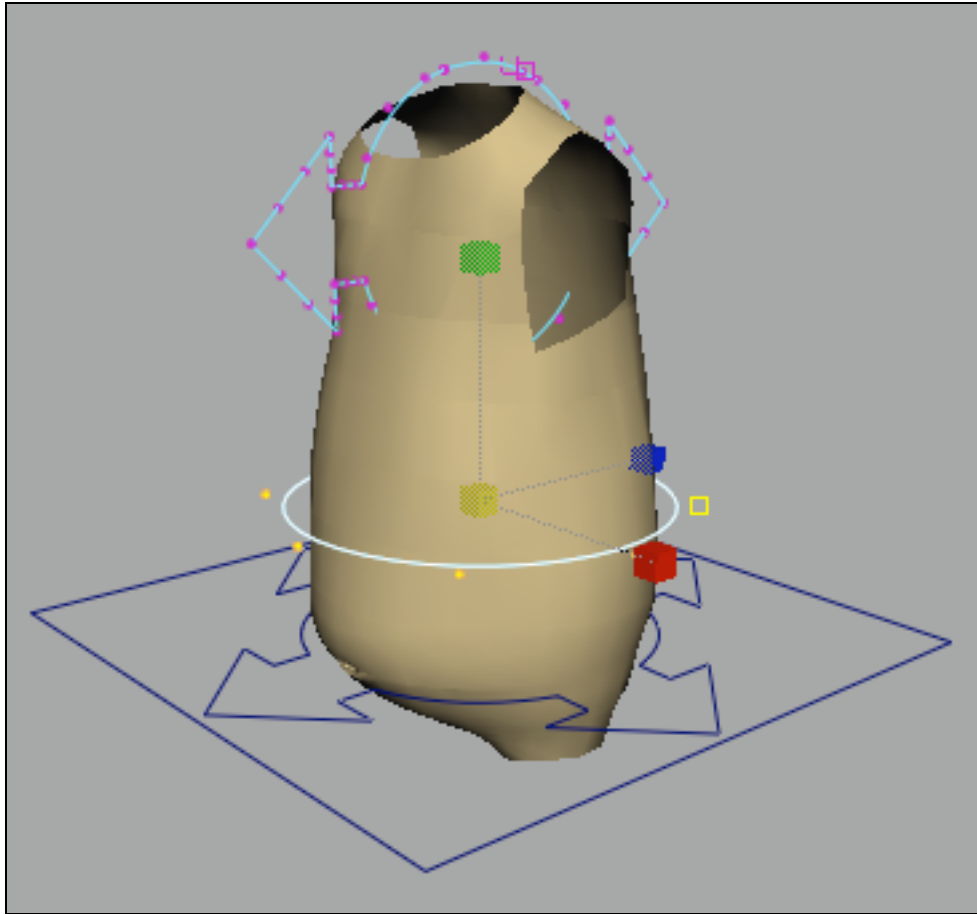


Figure 149 - scaling the circle

**138. Delete the original circle**

- Delete **nurbsCircle1**

**139. Rename the shape**

- In the channel box, rename **nurbsCircleShape1** to **torso\_1\_animShape**



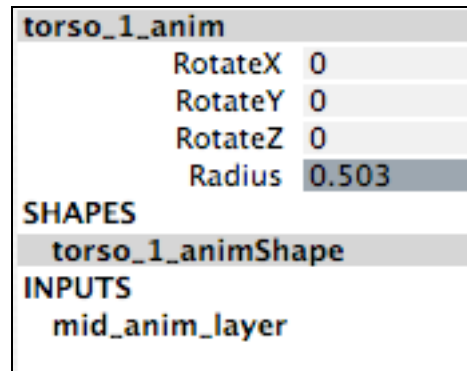


Figure 150- renaming the circle shape

#### 140. Create another nurbsCircle

- Choose **Create > Nurbs Primitives > Circle**

#### 141. Parent the shape to torso\_2\_anim

- Enter the following command in the script editor

```
parent -add -shape nurbsCircleShape1 torso_2_anim;
```

#### 142. Scale the Cvs so it's smaller

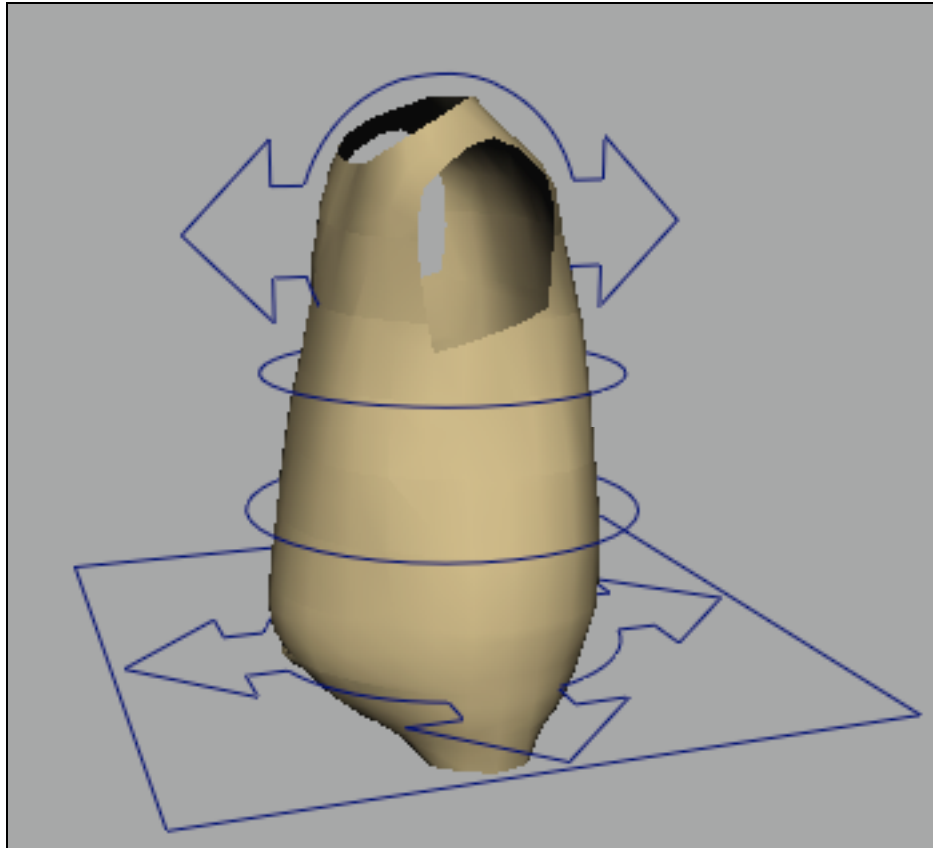
- Select **torso\_2\_anim**
- Hit **F8**
- Scale the CVS.

#### 143. Delete the original circle

- Delete **nurbsCircle1**

#### 144. Rename the shape

- In the channel box, rename **nurbsCircleShape1** to **torso\_2\_animShape**



*Figure 151 - Curve for controlling the FK, but no joint display!*

#### **Finishing Up The Torso**

We're almost finished, just a few more minor things to do! We've accomplished all of our animation and rigging requirements, but before handing something off to the animator, it's a good idea to make sure that they have a way of toggling the visibility of the controls they want on and off.

Currently they can use the layer editor to turn off ALL the controls on the character, but it's not set up to handle specific character controls, for example FK controls only. Also, the FK and IK controls all look exactly the same. This can become visually distracting for the animator. Therefore, I recommend creating different display layers for the various sections that we want the animator to be able to control.

#### 145. Create a display layer called **fk\_torso\_layer**

- Click on the Create Layer button
- Rename the new layer **fk\_torso\_layer**

#### 146. Add the fk controls to the **fk\_torso\_layer**

- Select **fk\_torso\_layer**
- Choose **Layers > Memberships**
- Select **fk\_torso\_layer** in the Relationship Editor
- Now select **torso\_1\_anim** and **torso\_2\_anim** in the Relationship Editor

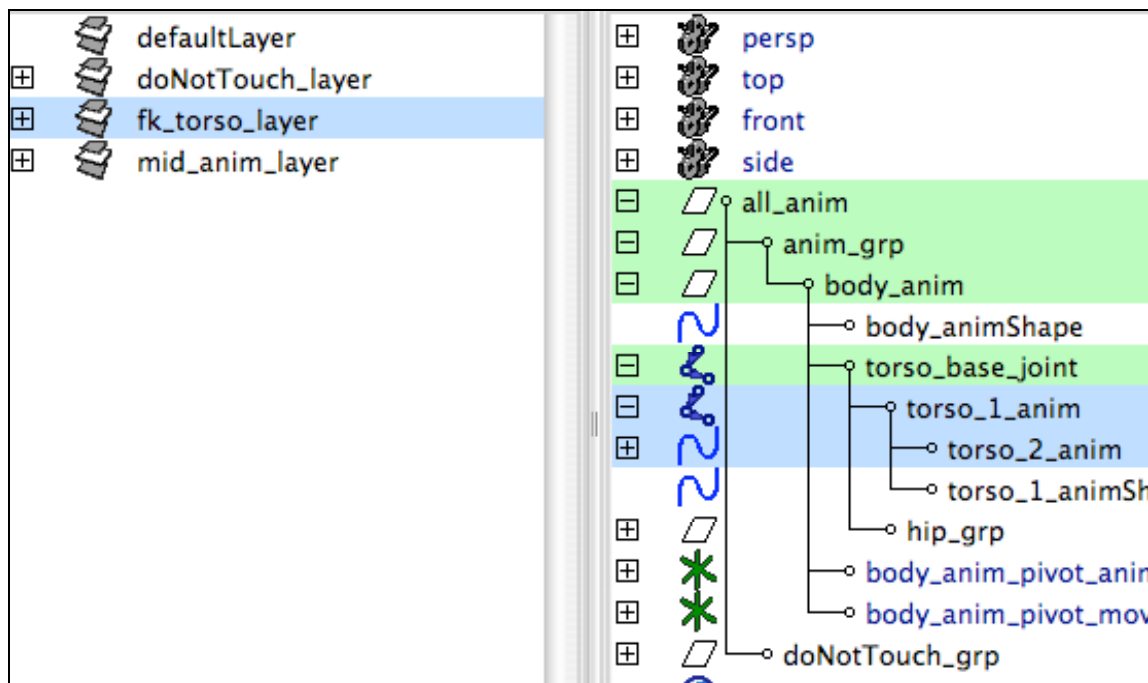


Figure 152 - adding torso\_1\_anim and torso\_2\_anim to the fk\_torso\_layer

#### 147. Change the colors of the layers

- Double-click on **fk\_torso\_layer**

- Change the color of the layer to something distinctive.

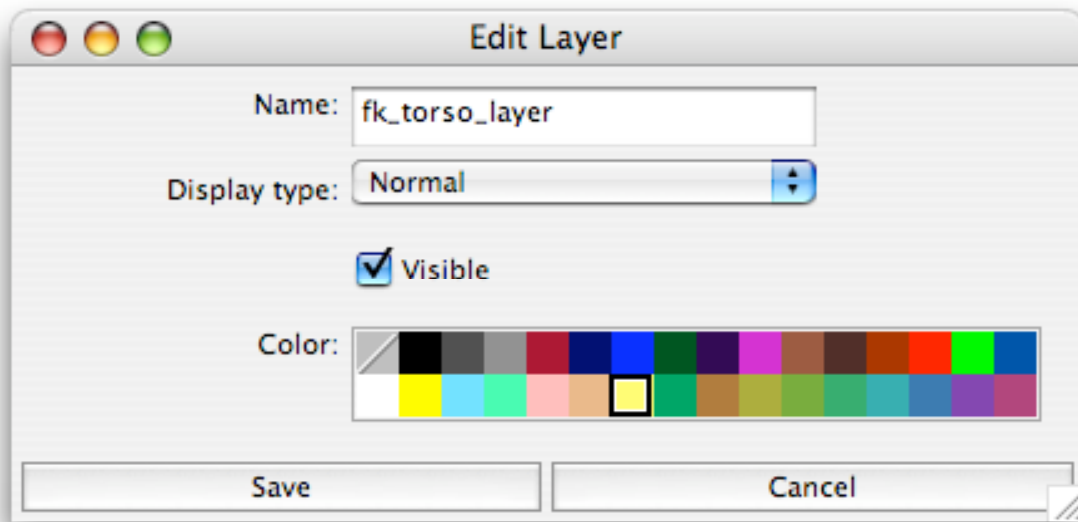
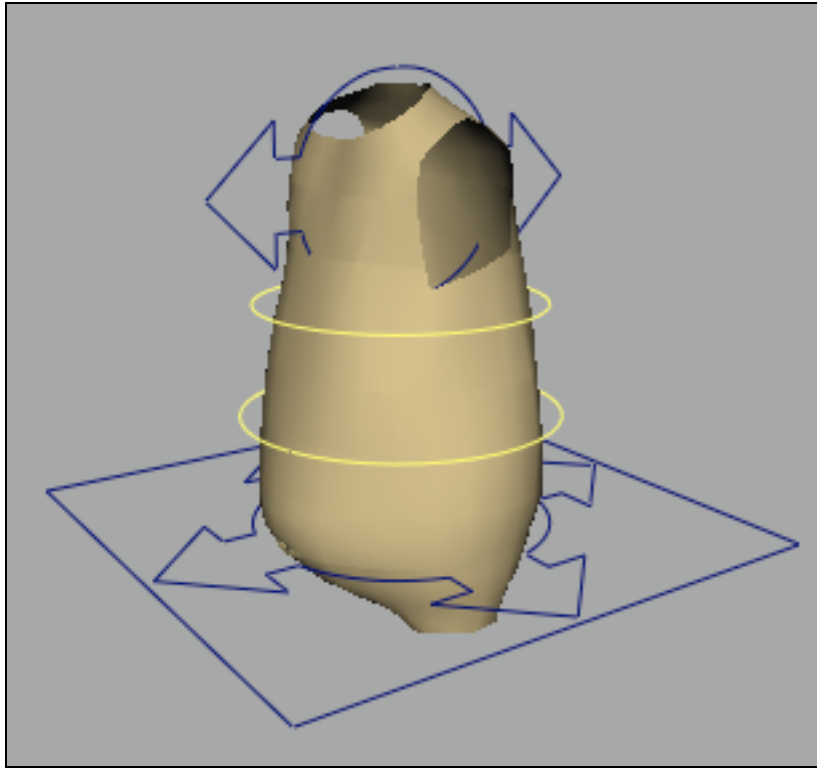
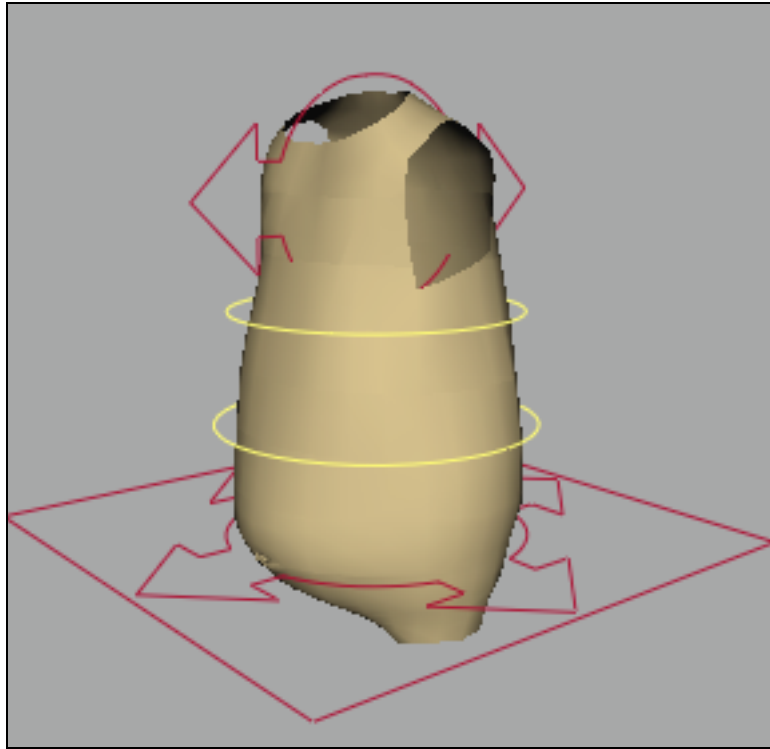


Figure 153 - Changing the color of the `fk_torso_layer`



*Figure 154 - the result of changing the color of the fk controls.*

- Do the same thing with **mid\_anim\_layer** and pick a color that will stand out against the background.



*Figure 155 - mid\_anim\_layer modified for better color*

#### Checking the Visibility

The final step is to check and make sure the visibility does what we want it to. First, turn off **mid\_anim\_layer** by clicking on the **V** in the layer. The entire torso should be hidden.

Next, turn **mid\_anim\_layer** back on, and try turning off **fk\_torso\_layer**.

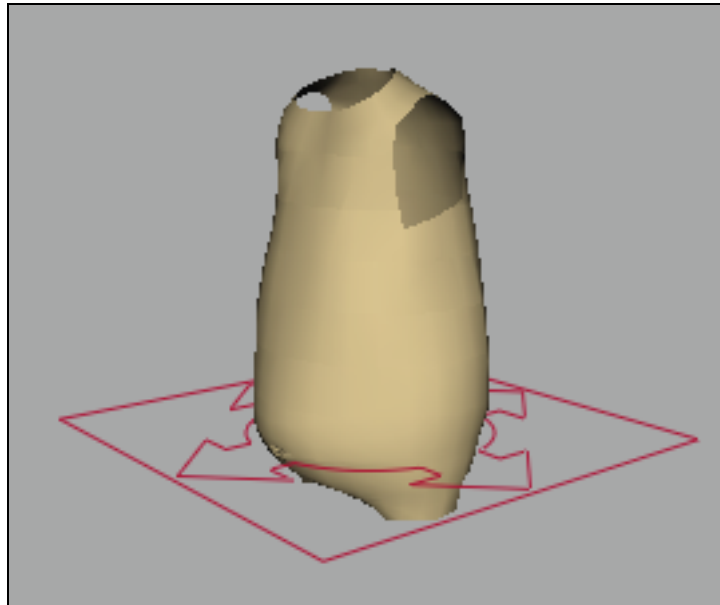


Figure 156- turning off the visibility of `fk_torso_layer`

Notice anything wrong? *The `shldr_anim` control is missing!* This is an example of why it's a good idea to *test* your rigs before handing them to the animators, you never know what might break!

So how do we fix this problem? Well, the reason that `shldr_anim` is disappearing is because it's a child of the `fk` control. If we make it NOT a child of the `fk` control, then it won't disappear anymore.

How do we do that? By using constraints.

If you take a look at the hierarchy of the `shldr_anim` node, you'll remember that we placed a `shldr_grp` node above it.

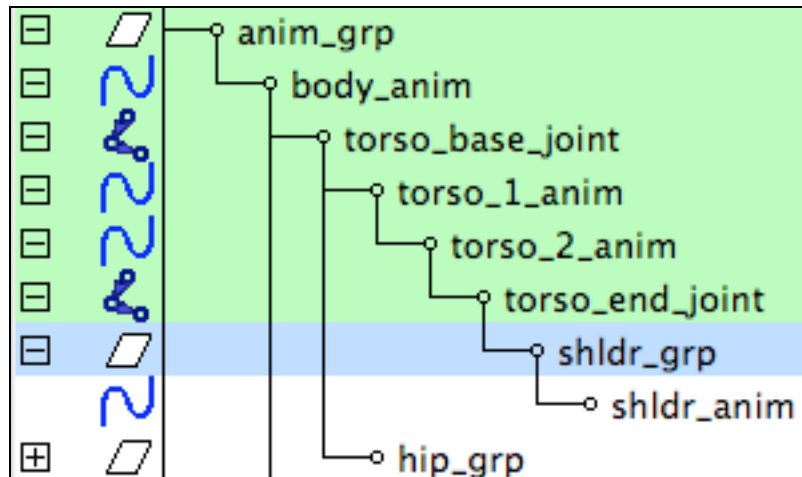


Figure 157 - hierarchy of the animation controls

We can constrain the shldr\_grp node to another node in the same spot! Then, we can put the shldr\_grp node at the same level of the hierarchy that hip\_grp is, meaning shldr\_grp won't be hidden, but it will *move with* torso\_end\_joint!

#### 148. Unlock shldr\_grp

- Select **shldr\_grp**
- In the channel box, click with the **Right Mouse button** and bring up **Channel Control**
- Switch to the **Locked** tab and select all the attributes on the left.
- Move them over to the right by clicking **Move >>**



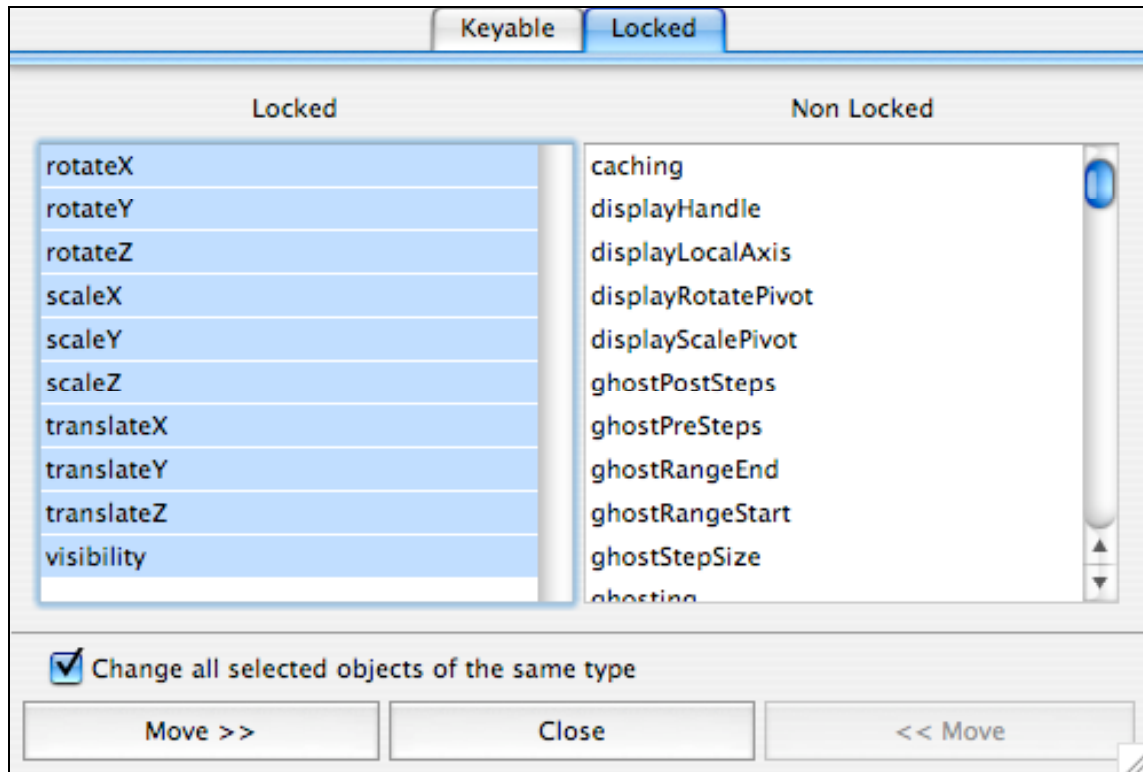


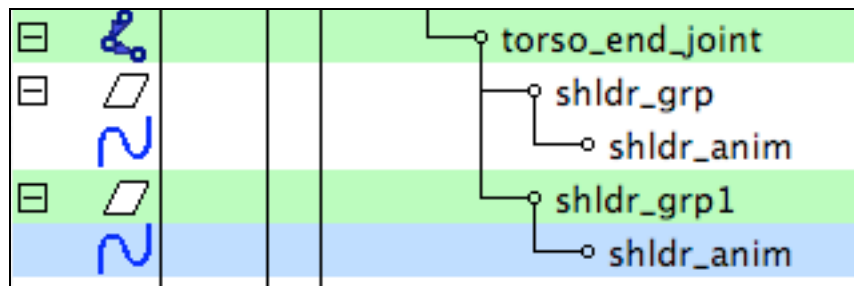
Figure 158 - attributes that need to be unlocked

#### 149. Duplicate shldr\_grp

- With **shldr\_grp** selected, hit **ctrl+d** to duplicate it

#### 150. Delete the child of shldr\_grp1

- Select the child of **shldr\_grp1** and delete it



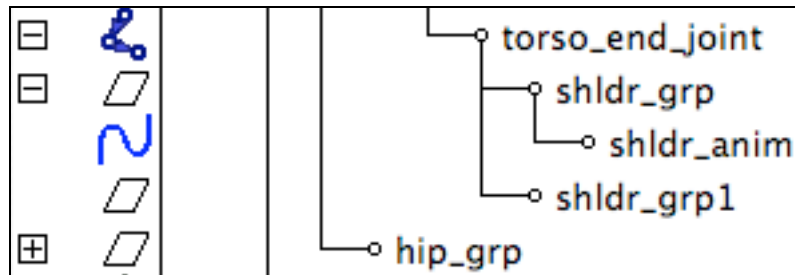


Figure 159 - Deleting the child of shldr\_grp1

151. Rename shldr\_grp1 to shldr\_grp\_pos

152. Constrain shldr\_grp to shldr\_grp\_pos

- Select **shldr\_grp\_pos** and then **shldr\_grp**
- Choose **Constrain > Parent**

153. Parent shldr\_grp under torso\_base\_joint

- Select **shldr\_grp** and then **torso\_base\_joint**
- Hit the **p** hotkey to parent it

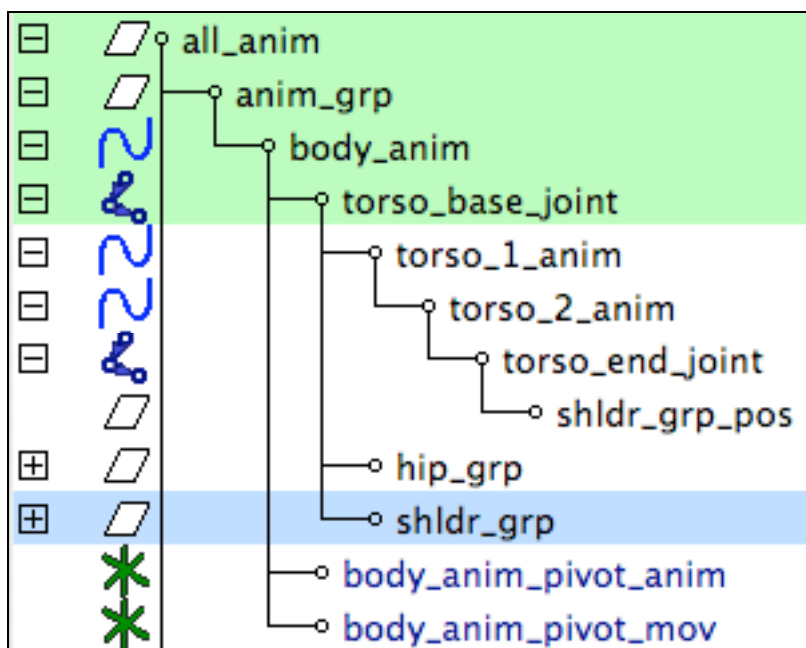


Figure 160- shldr\_grp reparented and constrained to shldr\_grp\_pos



## Master Classes

### Instructor Notes – Animator Friendly Rigging Part II

Jason Schleifer

At this point, your rig should be pretty solid. We've covered all our bases and made sure that all the controls work the way the animator will expect.

Save the file in your WIP directory as *backRig\_v1.ma*, so you know that this is the most recent version of your file.

It's time to move on to the next control we'll be looking at.. the head!

**Finished File: `jj_backRig.ma`**

#### The Head and Neck



Of course our next step is to add the head and neck controls to our character. While it's good fun to animate decapitated people running around like chickens with.. well.. you know what I mean, eventually you will want to have your characters be able to look around.

So, how do we determine what we need the head to be able to do? Well, just like the torso, we want to take a look at reference material to determine what is physically possible.



*Figure 161- analyze reference material to see what the head needs to do.*

Included on the dvd you will find for movies showing head specific reference in the movies/head/ directory. Please watch the movie files and analyze the way the head moves in relation to the torso. And remember the follow rule:

It is the *relationship between* the **head** and **shoulders** that defines the character's attitude.

#### **General head motion requirements**



## Master Classes

# Instructor Notes – Animator Friendly Rigging Part II

Jason Schleifer

- Head needs to be able to orbit side to side
- Head needs to be able to look up and down
- Head needs to be able to lean side to side
- Head should be able to move forward and back
- Head can compress and extend
- Head should be able to move side to side.

### Additional head motion considerations

- Head should be able to orient *independent* of shoulders and body (i.e. twist shoulders, head should stay at same orientation), or *with* the shoulders and body (twist shoulders & it orients the head).
- Head translation should be able to be done in shoulder space, body space, or all space.

So it looks like we need to be able to orient and translate the head, both in body and in shoulder “space”.

What does it mean to move something within another “space”? That simply means that it’s translations and rotations are going to be *relative* to the other object. For example, when I say “The head must translate in *shoulder space*”, I’m implying that the head will act as if it is a child of the shoulder. If it’s in *body space*, then it will act as if it’s a child of the body.

The fun thing about thinking about how different spaces can affect the various parts of the body, is that it starts to open up a whole set of approaches to rigging. What if the head were working in *hand space*? That would mean that when I translated the head, it would be moving in relation to however the hand was oriented. Pretty handy, if the head is resting in the character’s hand!

### Neck Motion Requirements

I’m going to tell you a little secret about the neck when animating characters..  
**most of the time, the animator really doesn’t care about a character’s neck.**  
What they care about is that the head is in the correct place in relation to the shoulders, and they *expect* the neck to just “do the right thing”.



## Master Classes

# Instructor Notes – Animator Friendly Rigging Part II

Jason Schleifer

It's true! We spend most of our time just making sure that the head is moving and acting correctly, and we really only worry about the neck if it looks wrong or we want to do something very specific with it. So nine times out of 10 (or more like 9.5 times out of 10), we never really pay attention to the neck, we really just care about the head.

That being said, the neck *is* important for the rigger. In fact, it's almost *more* important than the head. Since the animator doesn't want to have to touch the neck, it's vital that you get it to look correct with minimal involvement from the animator.

Another special note: If you have a neck that is longer than usual, or has special needs based on it's design, it's very important to provide enough control for the animator to get what they want. The technique I show for this class should give you that control, but you may need to make special considerations depending on the character design.

## Head/Neck Toolkit

### Constraints

Since we want to allow the head to move differently based on various options, we're going to have to create constraints to handle that motion. By constraining the *position* of the head to the neck, but not the *orientation*, we can keep the head's rotation still while we rotate the neck, giving the animator the ability to limit counter-animation.

### pointConstraint

Let's take a look at an example of **pointConstraints**. We used a pointConstraint in our bouncy ball example where we had the squash node follow the ball's translation, but not the rotation. But what if we want to give the animator different choices? What if we want the head of our character to follow the position on the neck *or* a global position so the body's motion doesn't affect it at all?

To do this, we'll need to provide the animator options with multiple pointConstraints.

#### 1. Create a new scene

- Choose **File > New**

#### 1. Create 3 Polygon Cubes

- Choose **Create > Polygon Primitives > Cube**
- Hit the **g** hotkey twice to repeat the command.

#### 2. Move pCube1 and pCube2

- Select and move pCube1 and pCube2 so they're offset from pCube3

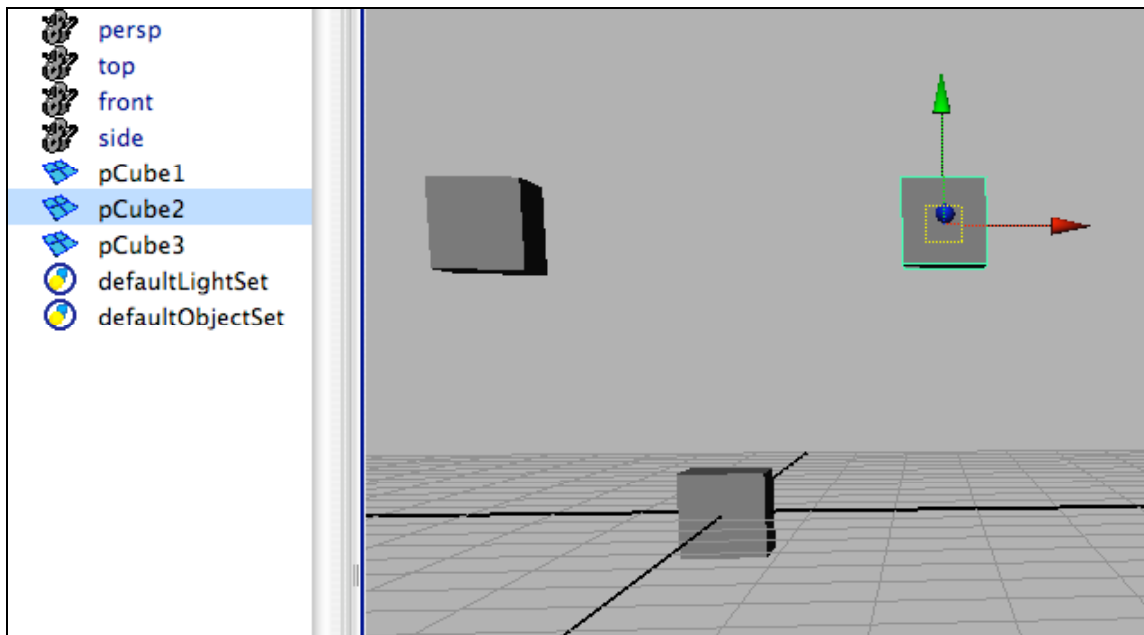


Figure 162 - pCube1 and pCube2 moved

#### 3. Constrain pCube3 to pCube1 and pCube2

- Select **pCube1** and then **pCube2** and finally **pCube3**
- Choose **Constrain > Point > Option Box**
- Make sure the options look like the attached image:



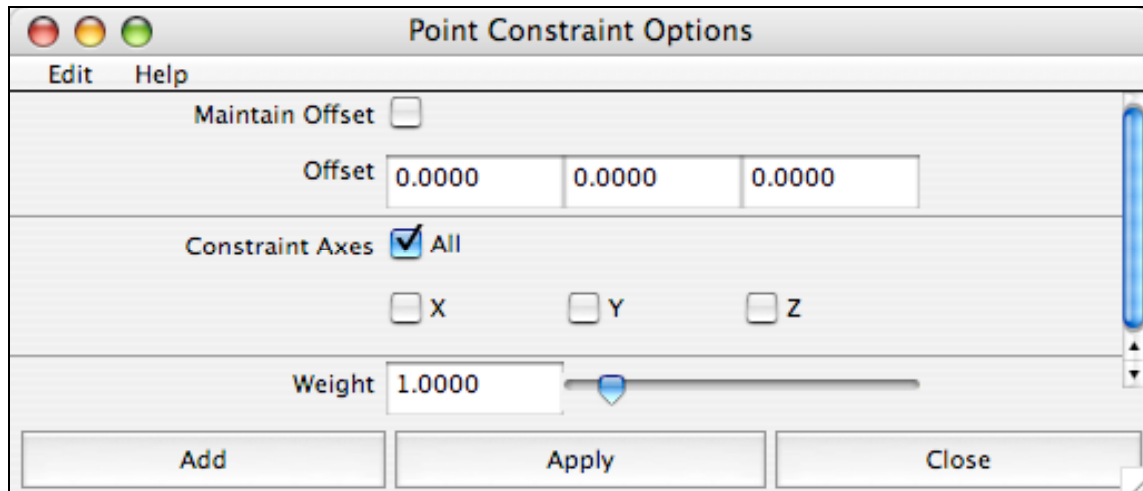


Figure 163 - pointConstraint options

- Click **Add**

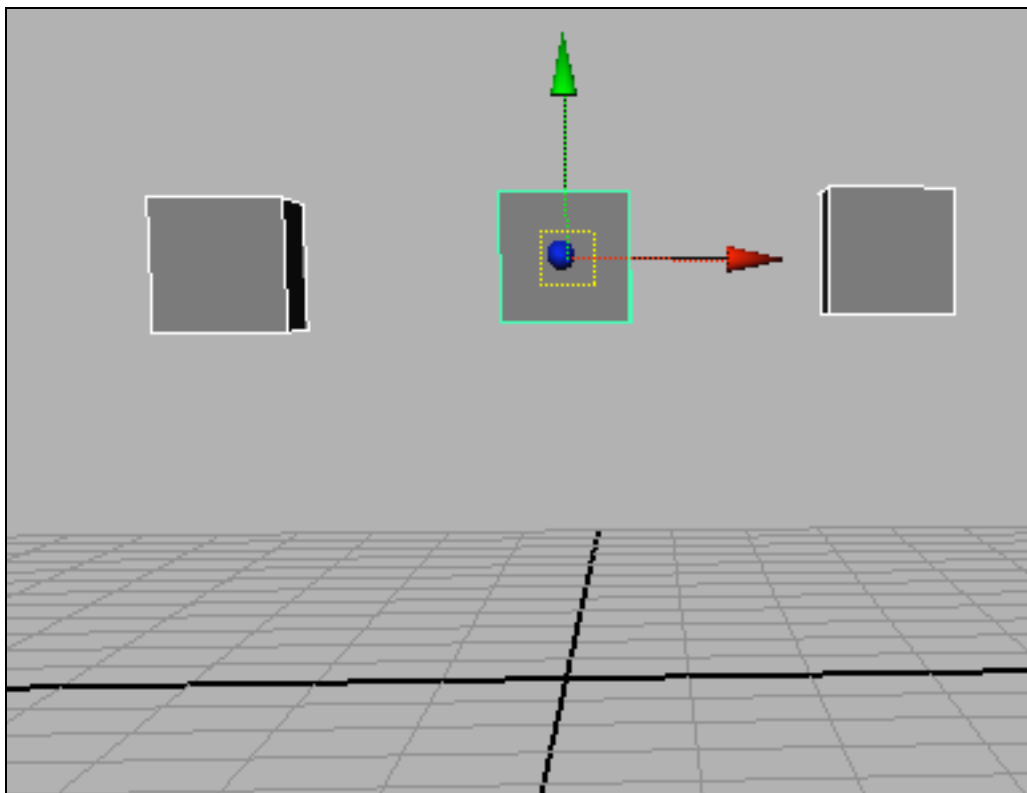


Figure 164 - point constraint added. Notice how pCube3 is halfway between pCube1 and pCube2

#### 4. Modify the constraint values

- Select **pCube3** and look in the **Channel Box**
- Click on **pCube3\_pointConstraint1** to open it's attributes.

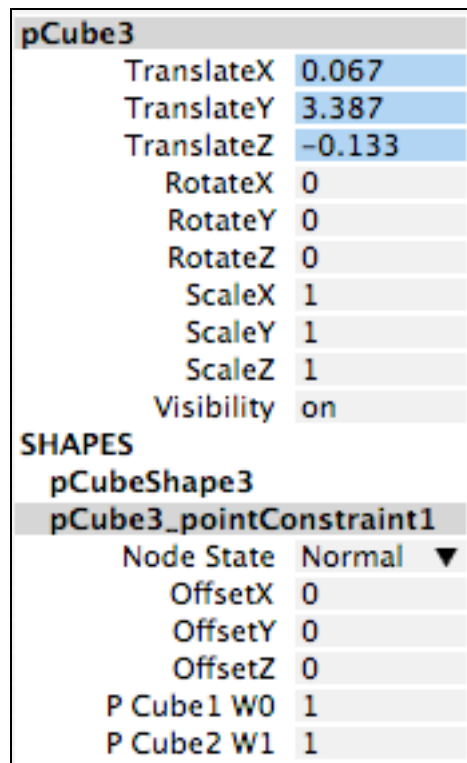


Figure 165 - pCube3 pointConstraint1 animation values.

- Notice the two attributes **P Cube1 W0** and **P Cube2 W1**. These are the two weight values for the point constraint. Right now they're both at **1**, so pCube3 follows them both equally.
- Change **pCube1 w0** to **0**. Notice now that when you move **pCube1** it doesn't affect **pCube3**. When you move **pCube2**, **pCube3** moves with it 100%.

So we now have multiple constraint options for the animator to work with. However, having the animator try and animate these values to get what they want is cumbersome. Wouldn't it be easier if they could just choose what object they wanted the cube to follow by a pulldown?

Of course it would! And that's what we'll give them!



Master Classes

## Instructor Notes – Animator Friendly Rigging Part II

Jason Schleifer

### 5. Add an attribute for choosing what object to follow.

- Select **pCube3**
- Choose **Modify > Add Attribute..**
- Enter the following values:

**Attribute Name:**    **constraint**

**Date Type:**         **Enum**

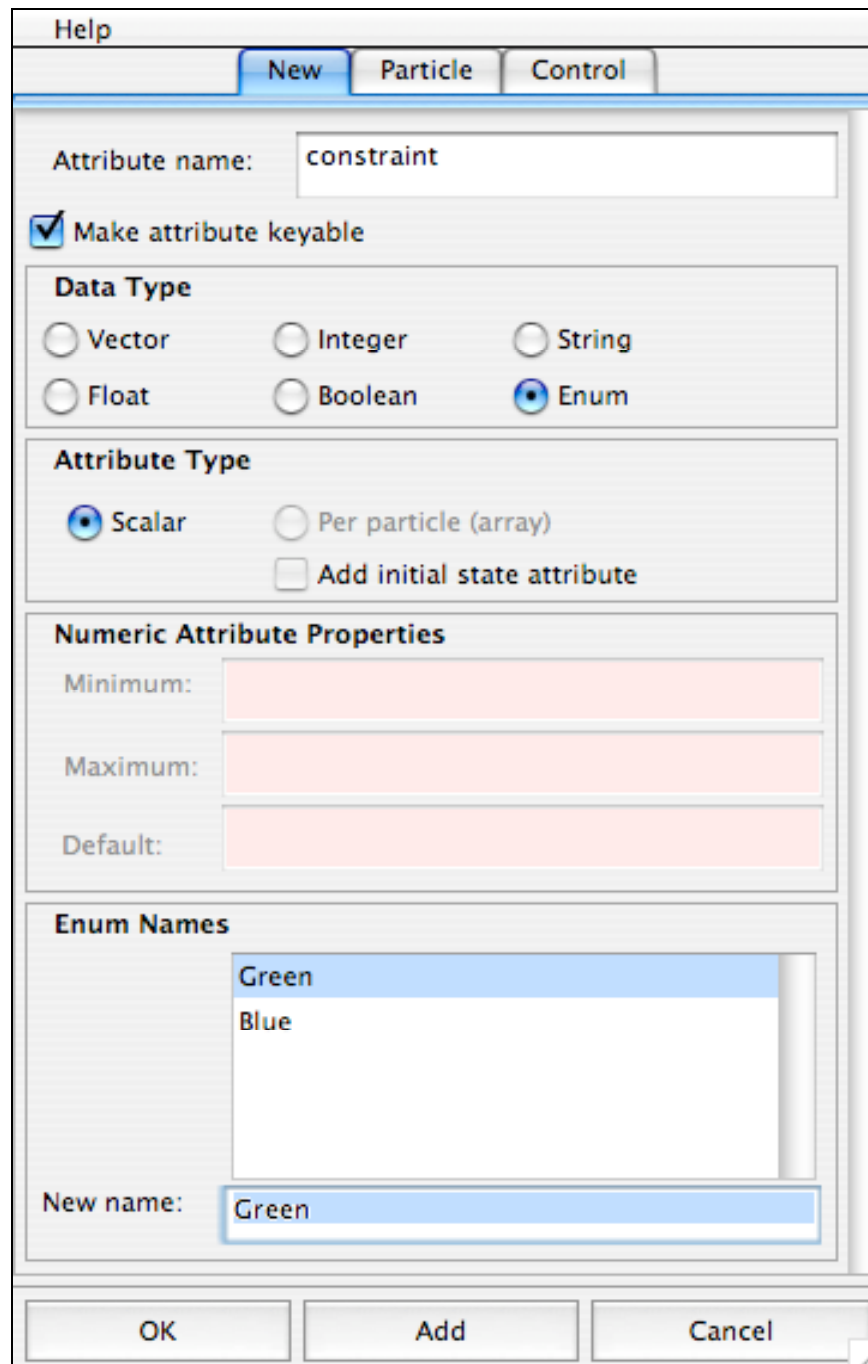


Figure 166 - Creating a new attribute of type enum

- Under Enum Names, select **Green**

- In New Name: type **pCube1**
- Now select **Blue**
- Enter **pCube2**

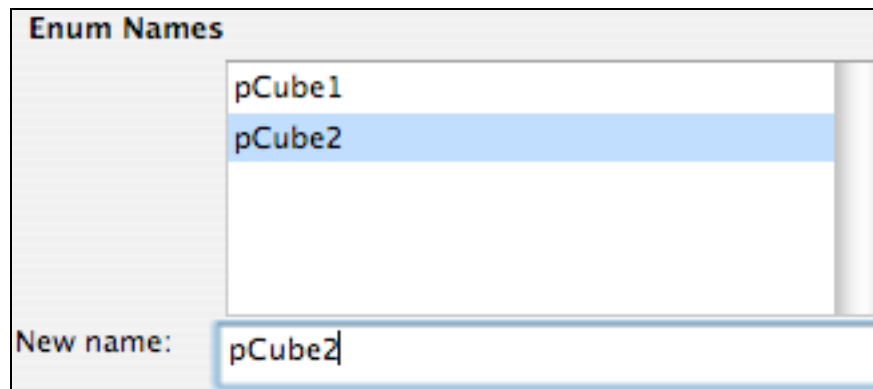


Figure 167- proper enum names chosen

- Click **Ok**

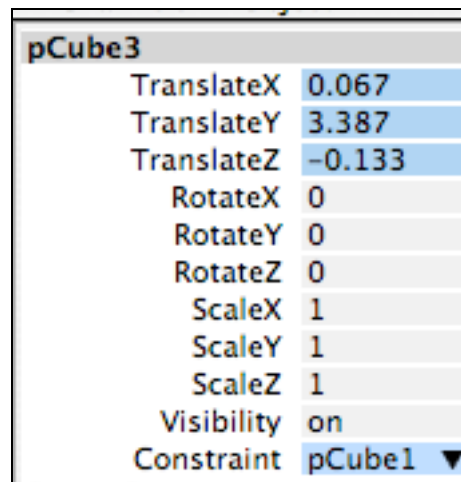


Figure 168- pCube3 now has an attribute called **Constraint** in the channel box, with a pulldown of the names we entered.

#### 6. Connect the constraint value to the constraint weights

- Choose **Animate > Set Driven Keyframe > Set**
- Select **pCube3** and click **Load Driver**
- Select **pCube3\_pointConstraint** in the **outliner** and click **Load Driven**

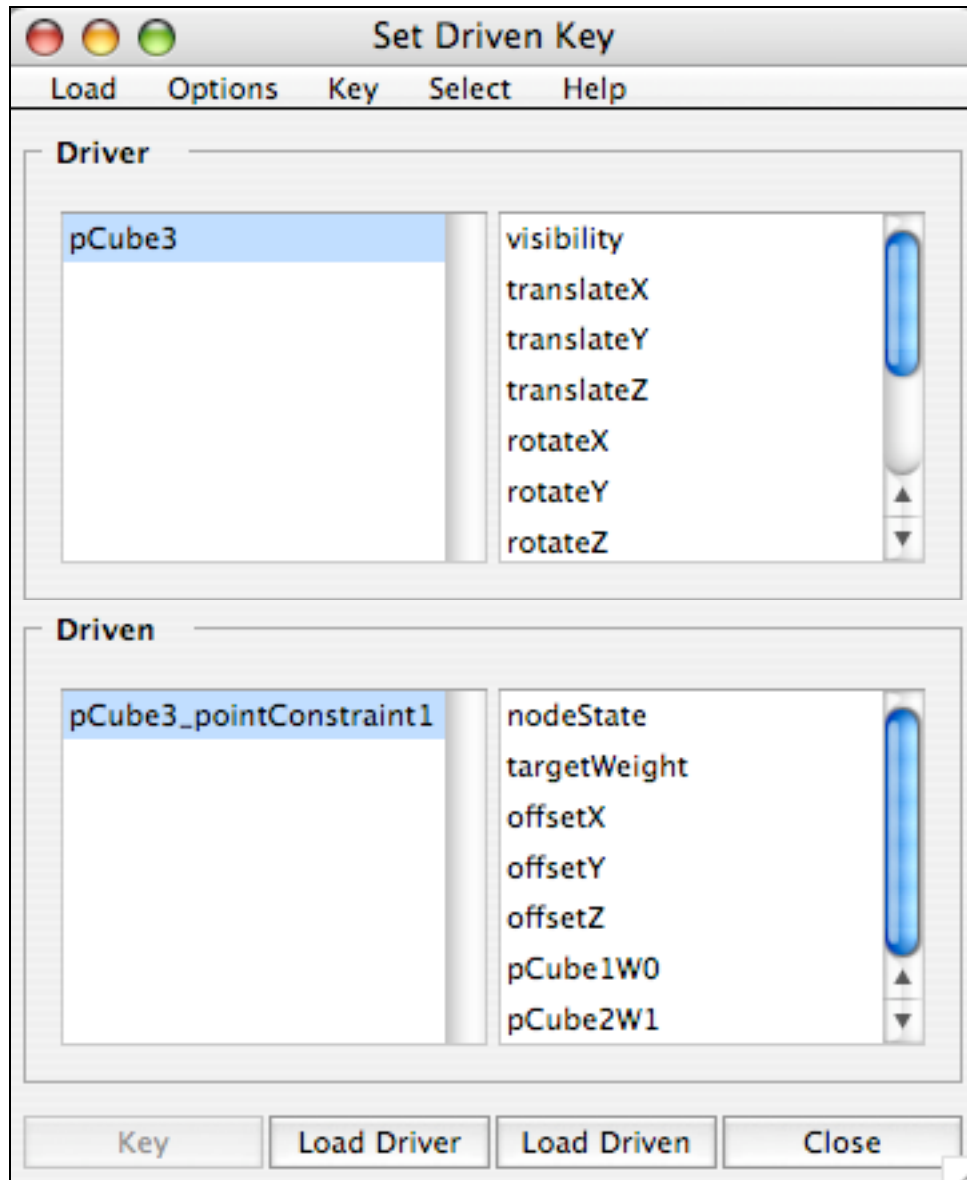


Figure 169 - setDrivenKeyframe window

- In the driver attribute section, scroll down until you see **Constraint** and then **select** it.
- In the driven attribute section, select **pCube1W0** and **pCube2W1**.

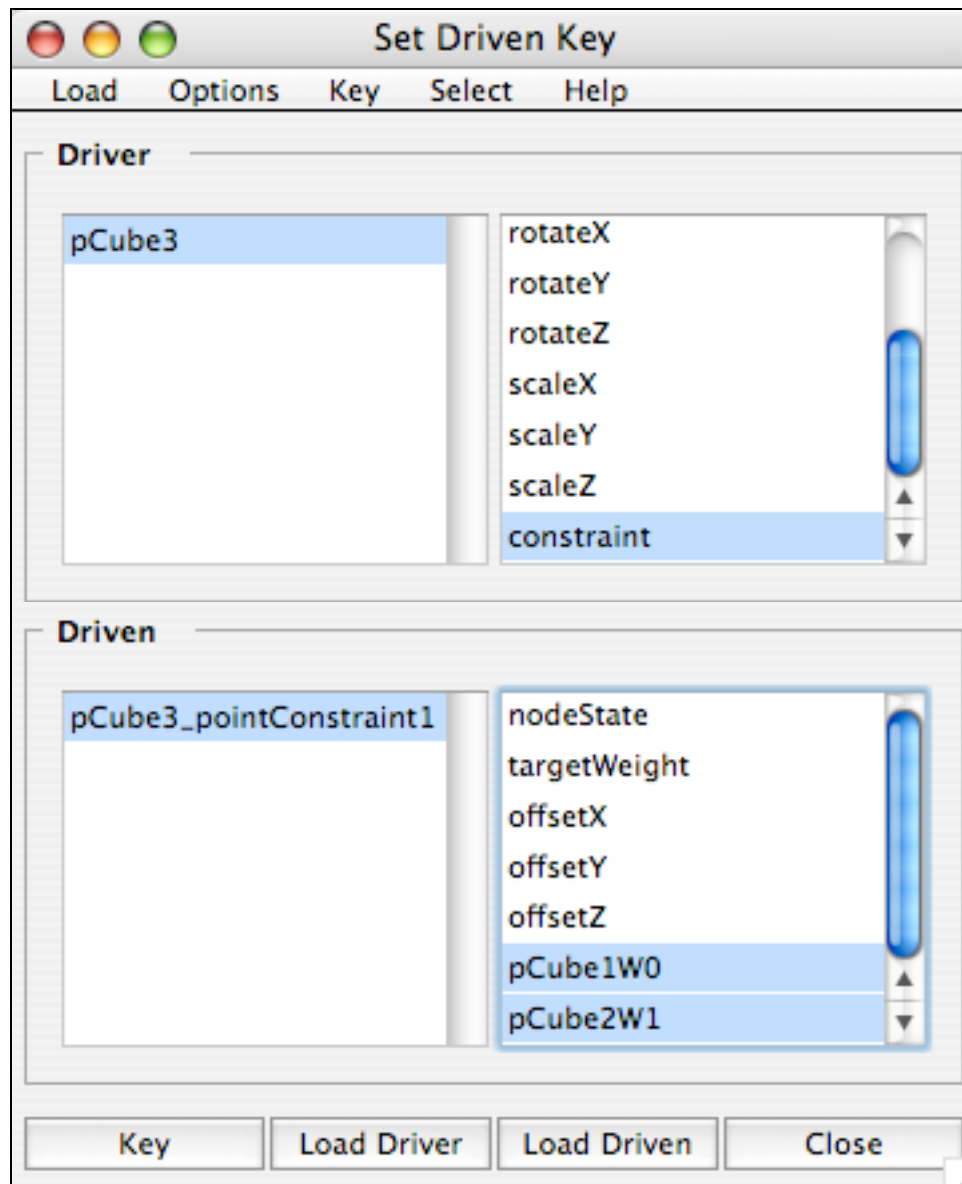


Figure 170 - constraint selected, along with pCube1W0 and pCube2W1

- In the **Channel Box** set the following values

**Constraint to pCube1**

**pCube1W0 to 1**

**pCube2W1 to 0**

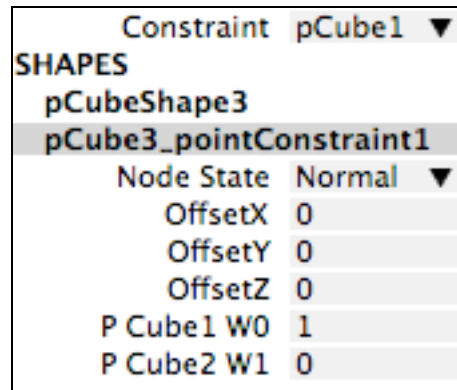


Figure 171- setting the values for pCube1W0 and pCube2W1 for constraint being pCube1

- Click **Key**
- Now set the following values

**Constraint to pCube2**

**pCube1W0 to 0**

**pCube2W1 to 1**

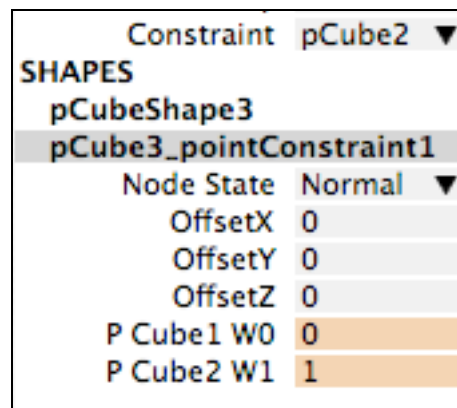


Figure 172 - setting values for pCube1W0 and pCube2W1 for constraint being pCube2

- Click **Key**

Now when you modify the constraint attribute, you can see that pCube3 will jump between following pCube1 and pCube2.

Isn't it easier for the animator to work this way instead of having to hunt and peck to find those weird weight attributes?





## Master Classes

# Instructor Notes – Animator Friendly Rigging Part II

Jason Schleifer

While this is pretty cool, the object does jump every time another object controls it. That makes it a bit difficult for an animator to use, because each time they switch controls, it will jump and they'll have to spend time trying to key the object in a place where it won't jump.

That's not so much fun for the animator, and it can literally waste hours of time!

So much like the pivot control, we have a similar scripted setup that will allow us to add constraints for an object, and then switch between spaces without the object jumping.

We'll discuss more of how this technique works later in the course, so for now let's just set up an example of how we can use it.

### Using `js_multiConstraint` scripts

Included with the DVD are a series of scripts used for dealing with multi-object constraints:

**`js_setUpMultiConstraint.mel`** – script to create multiple constraints on an object

**`js_setUpMultiConstraintUI.mel`** – builds the command to execute `js_setUpMultiConstraint`.

**`js_snapObjToConst.mel`** - snaps the object to the new constraint

**`js_multiConstraintSnapUI.mel`** – UI to build the `js_snapObjToConst` script.

When working interactively, you can use the two UI scripts to do everything you need. The commands that they use to execute the non\_UI scripts will be echoed to the script editor for use in scripts, if you like.

### 7. Create a New Scene

- Choose **File > New**

### 8. Create 4 poly cubes

- Choose **Create > Polygon Primitives > Cube**
- Use the **g** hotkey 3 times until you have 4 cubes in your scene.

#### 9. Create a locator

- This locator will be used for controlling the constraint.
- Choose **Create > Locator**

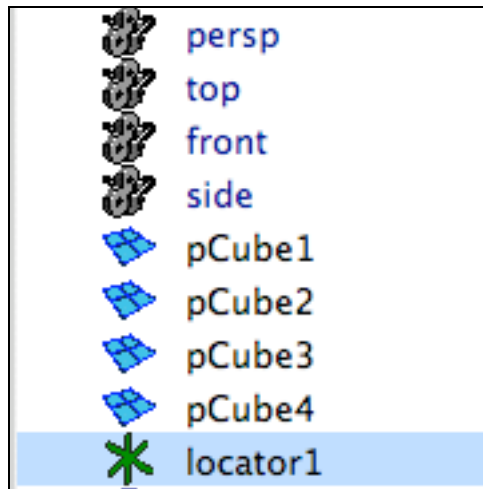


Figure 173 - 4 cubes and one locator created

#### 10. Set Up the Constraints

- Click on the **Setup Multi Constrain UI** button in the **Animator Friendly Rigging Shelf**
- Select **pCube1** and hit **Load Selected** in **Object To Constrain**
- Select **pCube2**, **pCube3**, and **pCube4**. Hit the **+** button in **Target Objects**
- Set **Point** and **Orient** constraints to **on**
- Select **locator1** and click **Load Selected** under **Control Object**
- Type **constraint** into the **Control Attribute** name.
- Click **Apply**



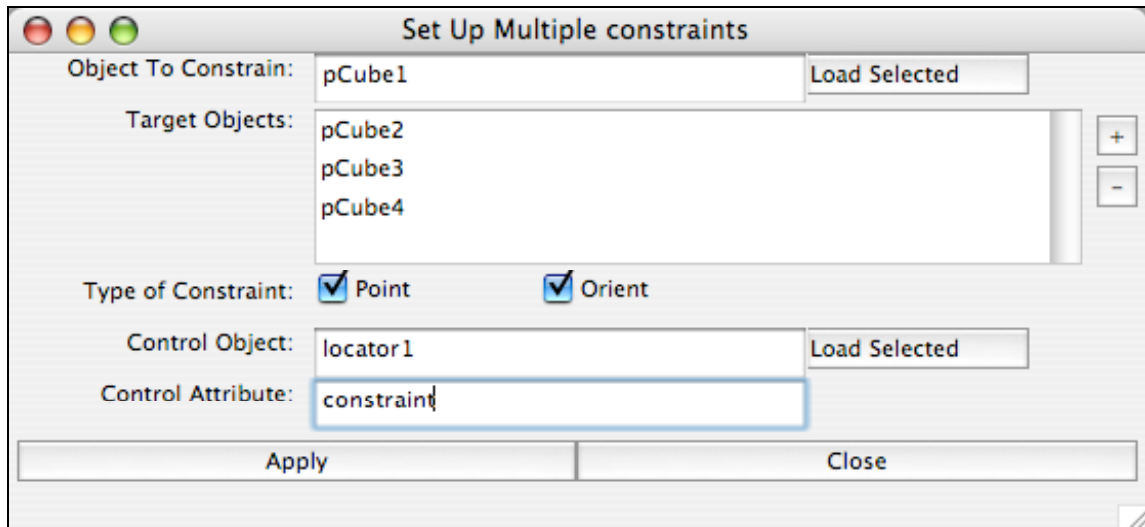


Figure 174 - Multiple Constraint UI

Look at the outliner and take a look at what happened.

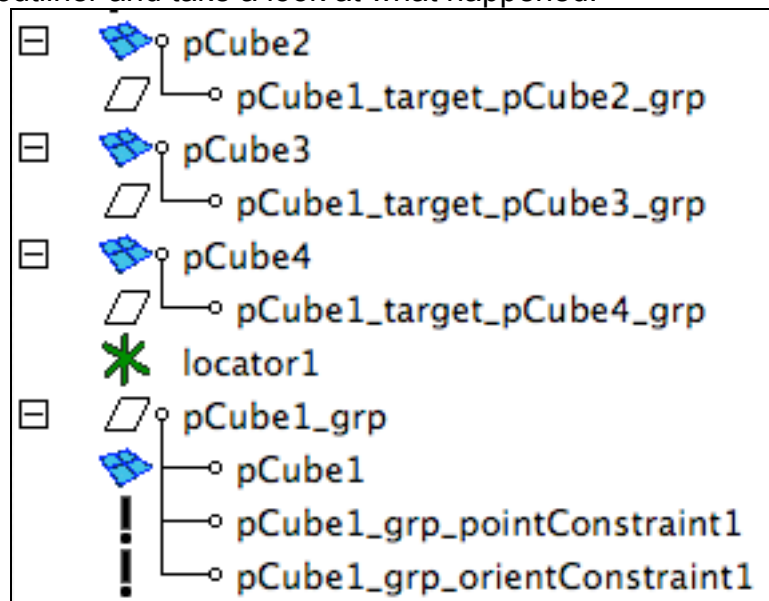


Figure 175 - Outliner after applying the Multi Constraint UI

The script did a number of things. First, it created a group node above pCube1, the object that is being constrained. This is so we can constrain the object to various items, but still animate it.

Second, it made duplicates of the group node and parented them to the target objects. This allows the group node to be constrained and makes sure that all the rotation orders and everything else are the same.

Third, it created a point and orient constraint from pCube1\_grp to all the other target grps.

Fourth, it added an attribute on locator1 (our control object) called Constraint. This attribute will allow us to toggle what object controls pCube1.

#### 11. Test the control

- Select **locator1**
- In the **Channel Box** you will see that the item controlling the cube is **pCube2**.
- Select **pCube2** and move it. **pCube1** should move with it.
- Now select **locator1** again, and switch the **constraint** attribute to **pCube3**. **pCube1** should move back to the location of **pCube3**.
- Set it back to **pCube2**.

#### 12. Use the snap tool

- With **locator1** selected, click on the **Multi Constraint Snap UI** button in the **Animator Friendly Rigging** shelf

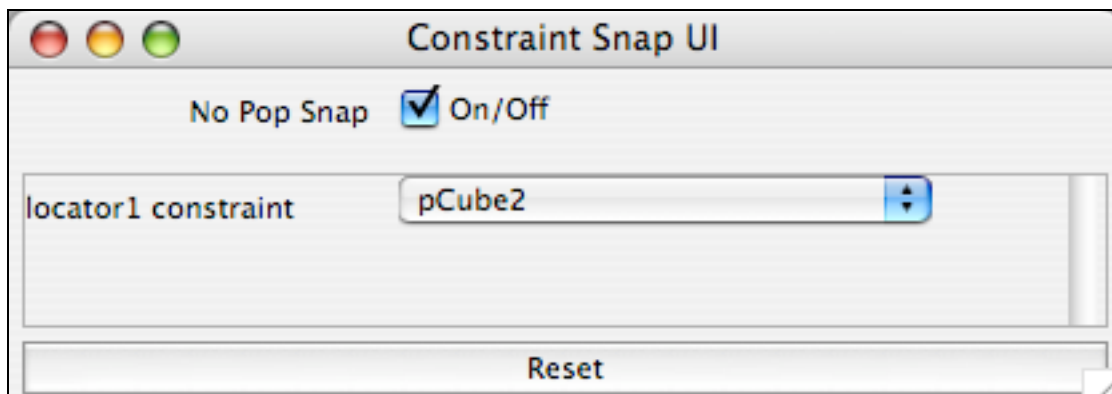


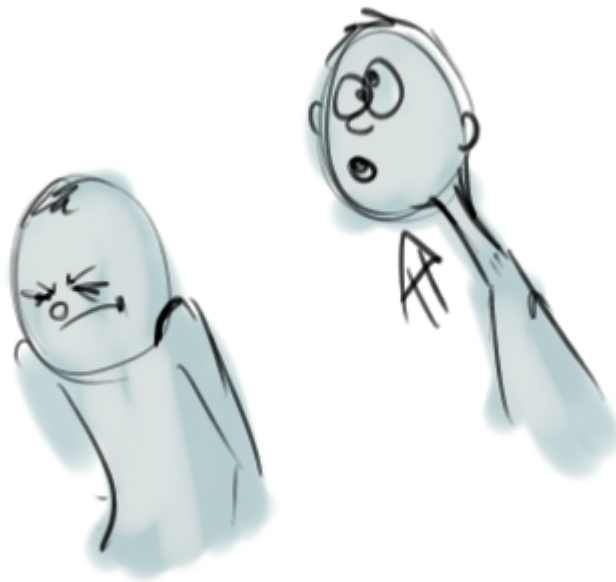
Figure 176 - Constraint Snap UI

- Change **locator1** constraint to **pCube3**
- Notice this time that **pCube1** *did not change position* when the constraint switched.
- That's because this UI sets a keyframe for the objects position the previous frame, and then sets a keyframe on the current frame in with a new position that *matches* the previous position.

Now the animator can snap to a new position without it affecting the object's position!

*Example File: multiConstraint.ma*

#### Neck Control



As I mentioned earlier, the neck is usually one of those things that the animator doesn't want to have to think about. What they want is for the neck to be pretty much as automatic as possible. It's good to have *some* control, but for the most part it's just the connection between the head and the shoulders.

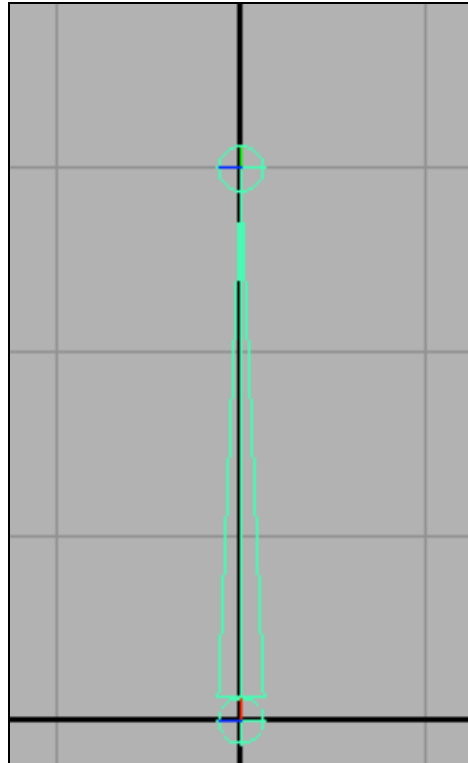
Because of this, it's important to make sure that the neck control is a relatively solid system. I like to use a similar rig to the back control, with slight modifications to ensure that the *base* of the neck doesn't bend too much and cause strange deformations.

#### 1. Create a new scene

- Choose **File > New**

#### 13. Create a joint segment that goes from 0 0 0 to 0 3 0

- Choose **Skeleton > Joint Tool**
- Click at **0 0 0**
- Click at **0 3 0**
- Hit **Enter**



*Figure 177- Joint segment created*

#### 14. Segment the joint into 3 pieces

- Click on the **Split Selected Joint** button in the **Animator Friendly Rigging Shelf**
- Set **Segments** to 3

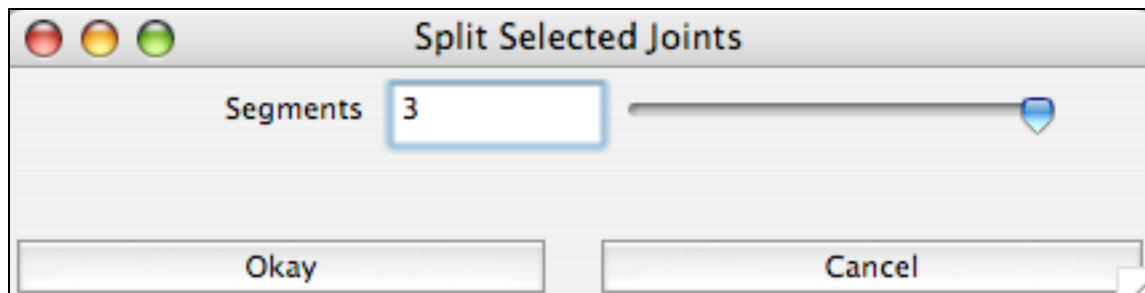


Figure 178 - Split Selected Joint window

- Click **Okay**

#### 15. Use splineIK

- Choose **Skeleton > Spline IK Handle Tool**
- Click on **joint1** and on **joint2** (the end joint).

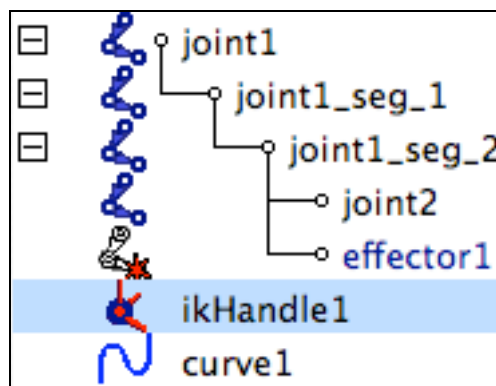
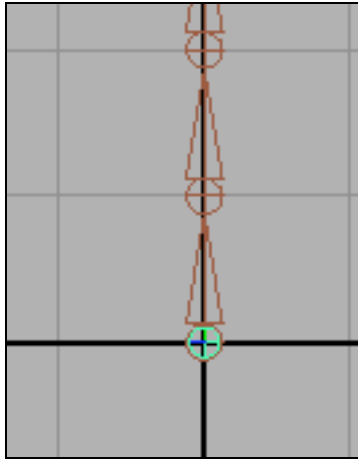


Figure 179 - Spline IK handle created

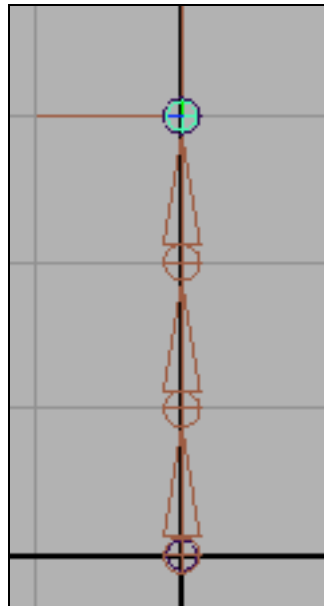
#### 16. Create two joints to use for skinning

- Choose **Skeleton > Joint Tool**
- Holding down the “x” hotkey for grid snap, click and drag near joint1



*Figure 180 - Creating the base joint for controlling the curve*

- Hit **y** to complete the action and put you back into the Joint tool.
- Hold down **x** and click and drag near **joint2** to create another joint for the top of the splineK.



*Figure 181 - creating an end joint for the splineK*

#### 17. Skin the curve to the two joints.

- Select **joint3 joint4** and **curve1**



- Choose **Skin > Bind Skin > Smooth Bind**

#### 18. Create the stretch rule on the spine

- Select **curve1**
- Click on the **Create Stretchy Spline** button the **Animator Friendly Rigging Shelf**

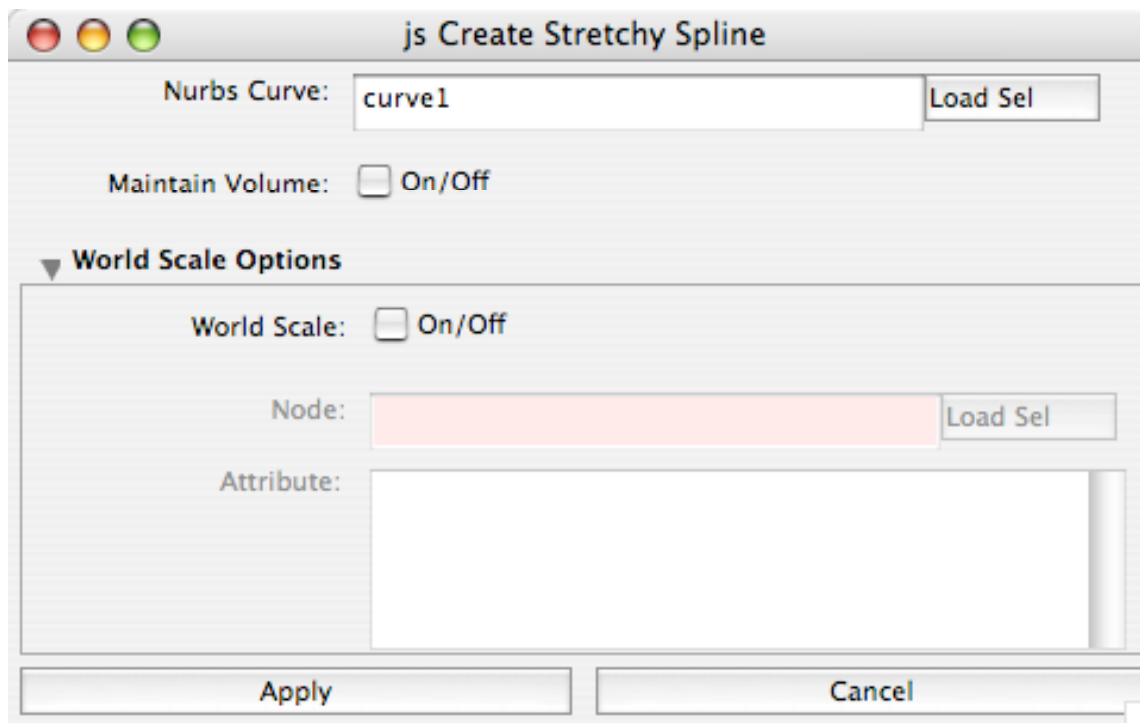
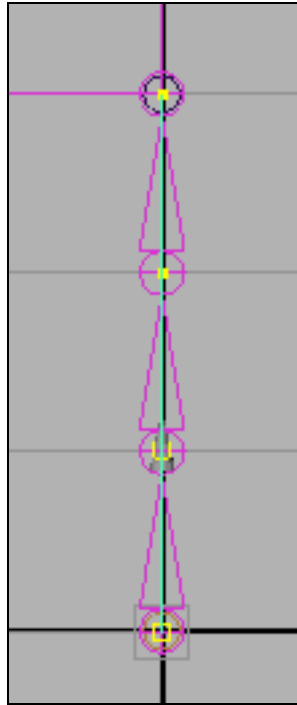


Figure 182 - js Create Stretch Spline Options

Drag **joint4** around. Notice how the base of the joints rotates quite a bit? We want to limit that rotation so we don't get a strange kink in the neck.

#### 19. Display the cvs on the curve

- Select **curve1**
- Choose **Display > NURBS > Cvs**



*Figure 183 - cvs displayed on the curve*

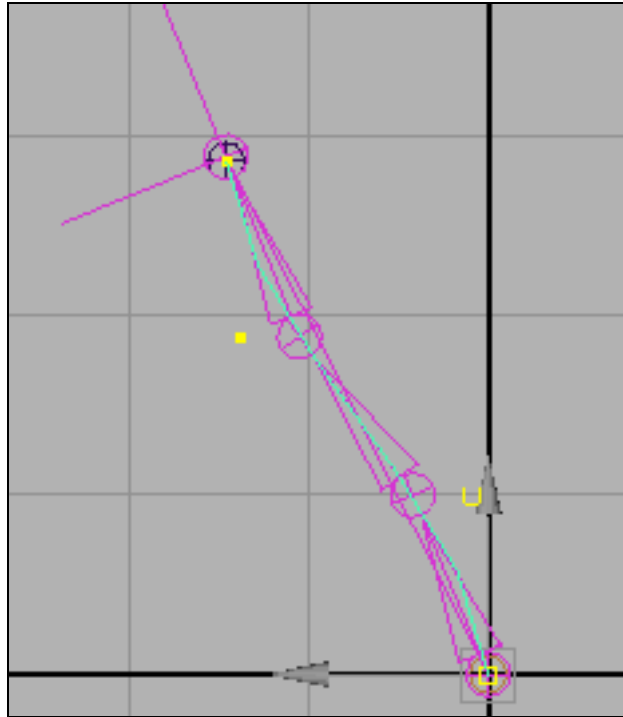


Figure 184 - what the cvs are doing when we pull joint4

If we can have the bottom portion of the rig control more of the cvs, it will make appear that the bottom of the spline doesn't bend so much.

## 20. Create a joint near the U cv

- Choose **Skeleton > Joint Tool**
- Click near **0 1 0** while holding down the **x** hotkey to create a joint

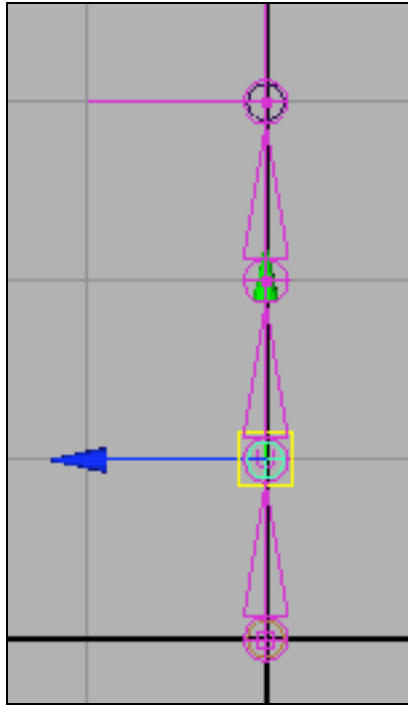


Figure 185 - creating another joint to help influence the curve.

#### 21. Add that joint as an influence to the bind skin

- Select **joint5** and **curve1**
- Choose **Skin > Edit Smooth Skin > Add Influence**

#### 22. Parent joint5 to joint3

- In the Outliner, drag **joint5** onto **joint3** so it's now a child.

Now when you manipulate **joint4**, you should see much better behavior along the bottom of the spline.

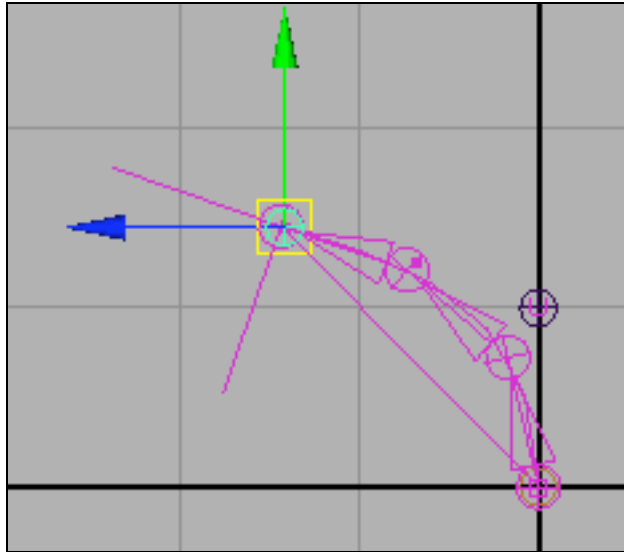


Figure 186 - base of neck doesn't deform as much

Notice, however, that if you bring **joint4** close to the base, the spline will twist and break.

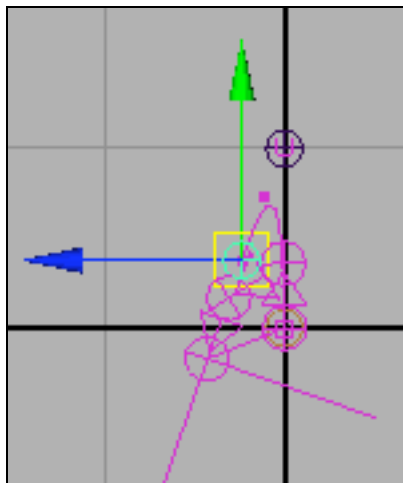
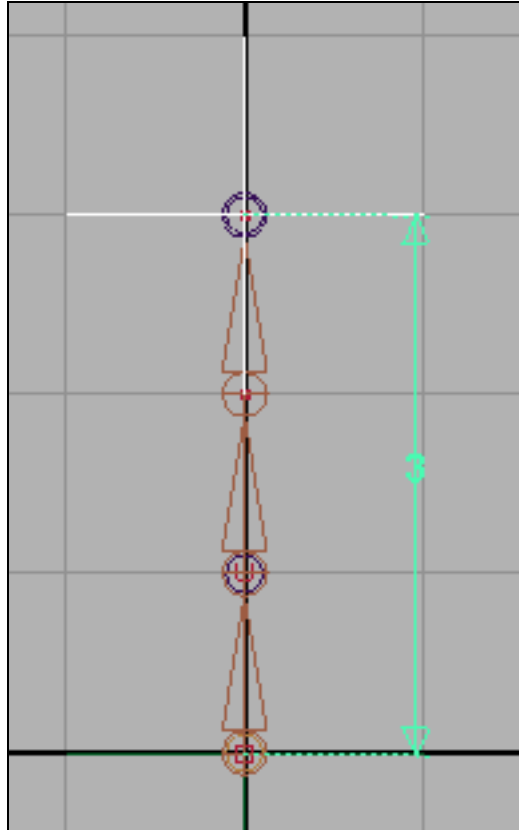


Figure 187 - Spline twisting and breaking when brought too close together

This is because the cvs on the curve are doubling up, causing the curve to pinch. We can solve this problem by scaling the bottom joint to move the cvs closer to the bottom whenever the joint gets close enough.

#### 23. Create a distance node to measure the distance between the two joints

- Choose **Create > Measure Tools > Distance Tool**
- Holding down **x**, click once at the base of the spline, and once again at the top.



*Figure 188- Creating the measure tool*

#### 24. Parent the tool to the appropriate joints

- Parent **locator2** under **joint4**

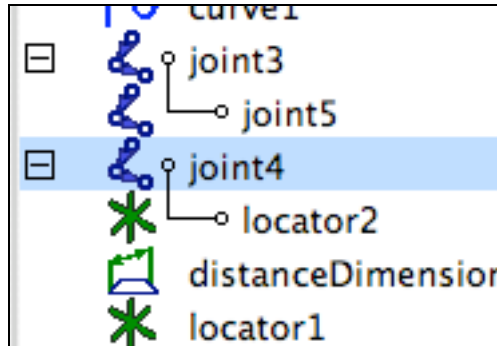


Figure 189 - Parenting the distance tool

Notice now that the distance measurement will change as you move around joint4.

#### 25. Create a setDrivenKeyframe from the distance to the scale

- Select **joint3**
- Highlight the scale attributes in the channel box
- Choose **RMB > SetDrivenKey..**
- Select **distanceDimension1** in the **outliner**
- Pickwalk DOWN to the shape node by hitting the **down arrow** on the keyboard.
- Click **Load Driver**
- Select the **distance** attribute in the right side of the **Driver** section.

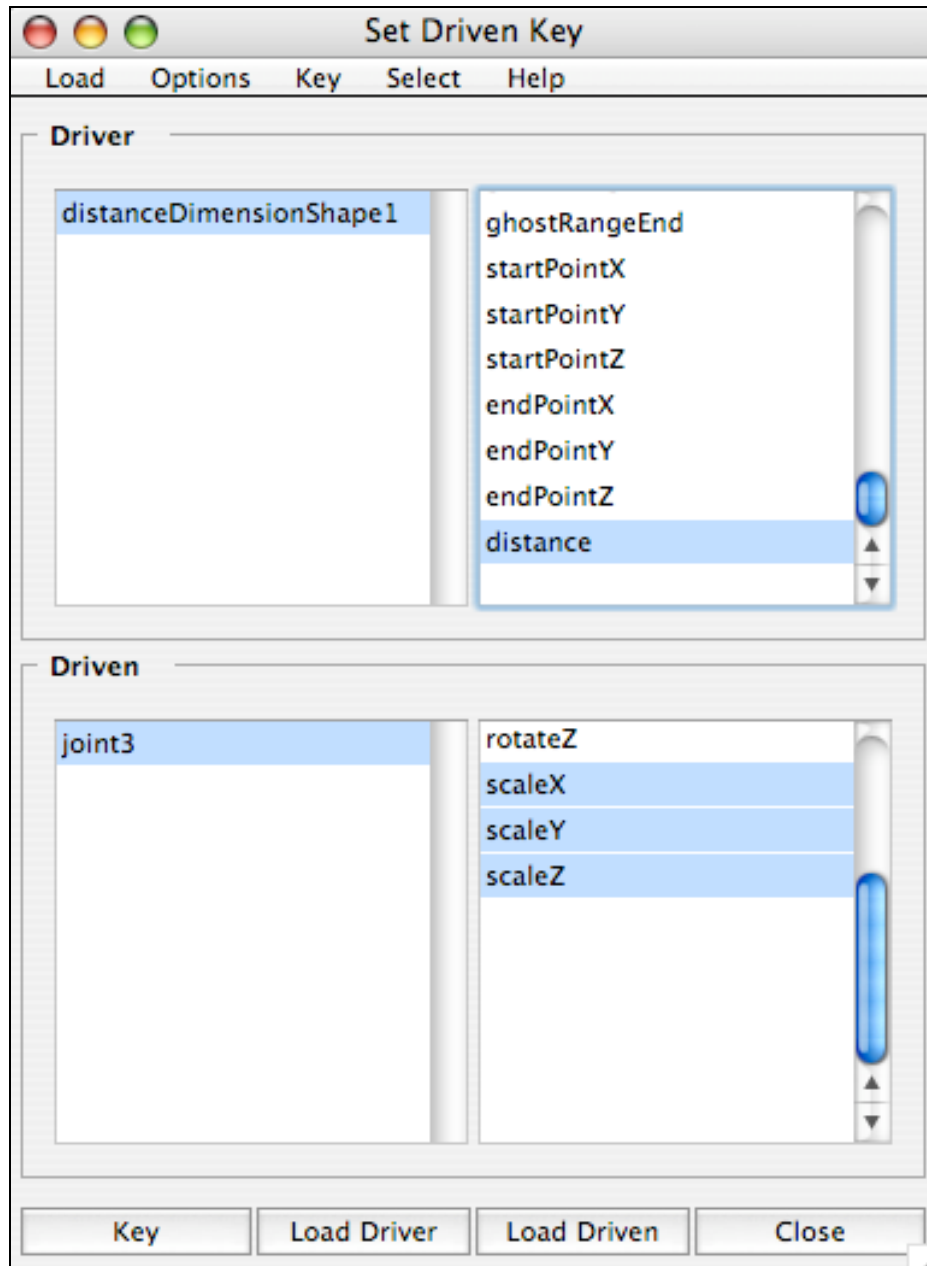


Figure 190 - DistanceDimensionShape1.distance will control joint3.scaleX, scaleY, and scaleZ

- Click **Key** to key the scales at a value of 1 for this distance.
- Select **joint4** and move it close to **joint3** so the spline looks completely worked out.



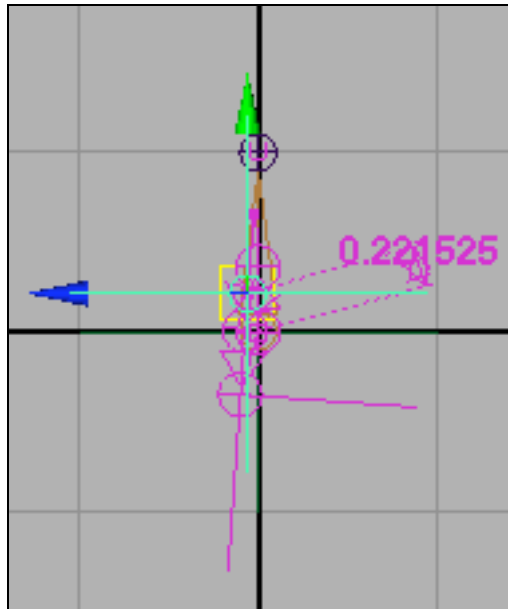


Figure 191 - joint4 and joint3 too close together

- Select **joint3**
- Scale **joint3** so it's values are about **.01**

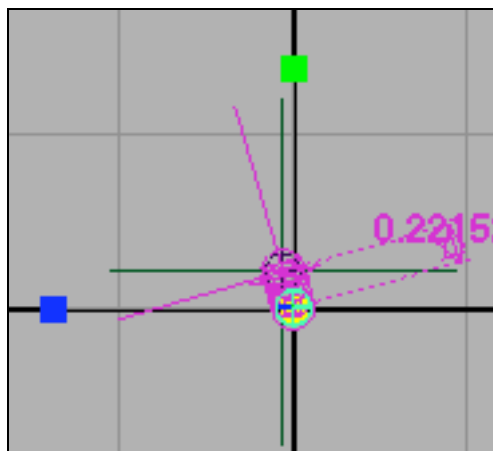


Figure 192- scaling joint3 down to .01

- Click **Key**

Now watch as you move joint4 around. The when it gets closer to the base, joint3 scales down making sure the neck doesn't explode! We can use this technique to guarantee a solid neck solution.



Master Classes

## Instructor Notes – Animator Friendly Rigging Part II

Jason Schleifer

*Example File: exploringNeckIdea.ma*

### Creating the Neck Rig

Now that we've explored the constraints and a neck skeletal solution, let's go ahead and create the neck rig for JJ!

#### Import the Geometry

##### 1. Open the latest rig file

- Choose **File > Open**
- Navigate to your WIP directory
- Open *jj\_backRig\_v1.ma* (or whatever your latest version is)

##### 26. Show the head geometry

- Open the **Outliner**
- Select **l\_brow\_geo, r\_brow\_geo, hair\_1\_geo, hair\_2\_geo, hair\_3\_geo, hair\_4\_geo, head\_geo, l\_eye\_grp, r\_eye\_grp**

# Autodesk®

## Master Classes

### Instructor Notes – Animator Friendly Rigging Part II

Jason Schleifer

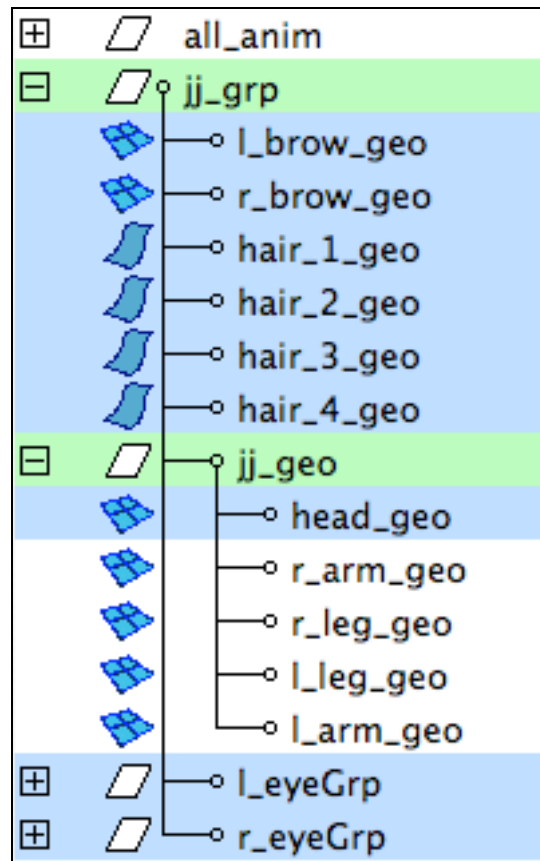


Figure 193 - geometry for jj's head

- Hit **CTRL+g** to group all the pieces together. Then hit **SHIFT+P** to unparent the new group so it's not a child of **jj\_grp** anymore.

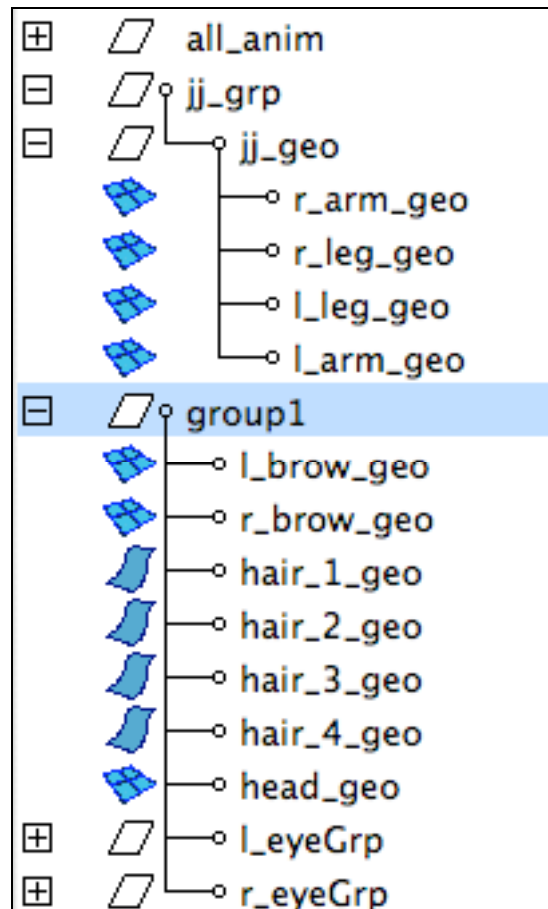


Figure 194 - head items ungrouped from jj\_grp

#### 27. Hide jj\_grp so we don't see unnecessary geometry

- Hide jj\_grp.

Example file: jj\_neck\_rig\_wip\_v1.ma

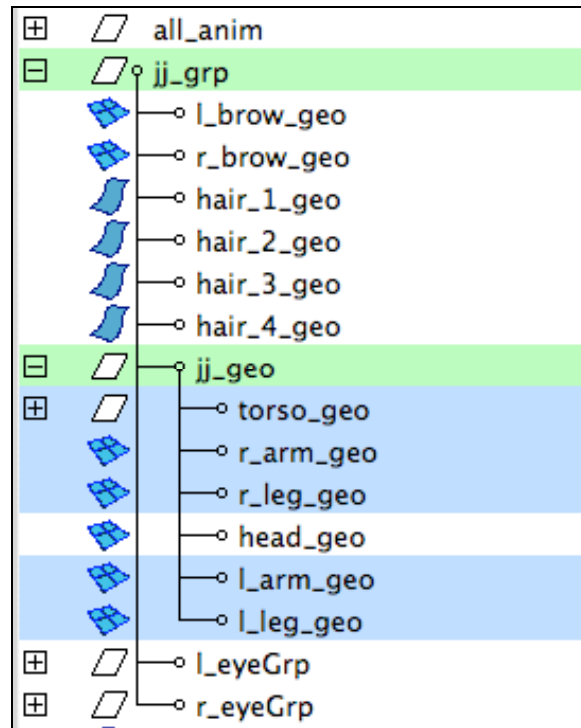
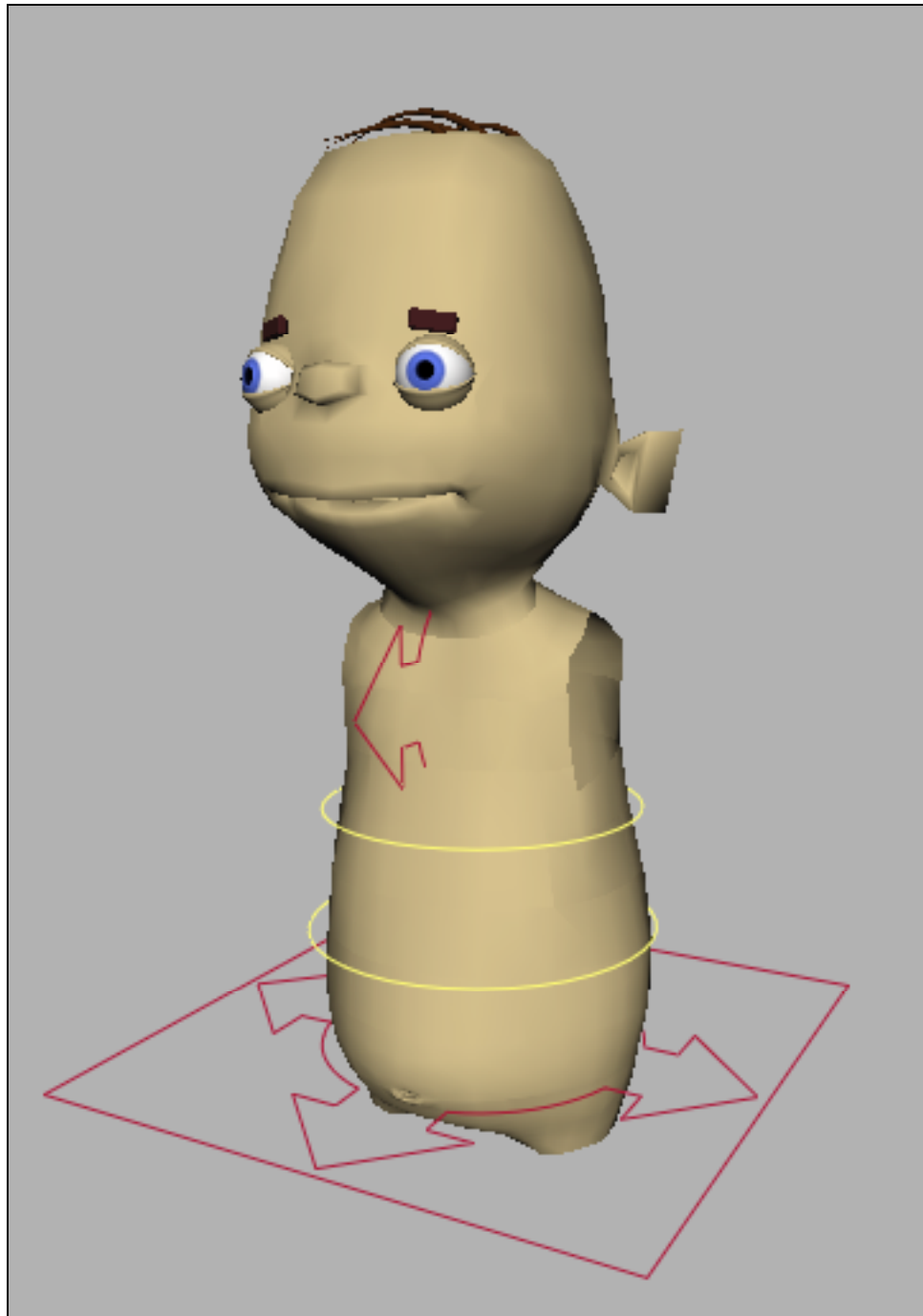


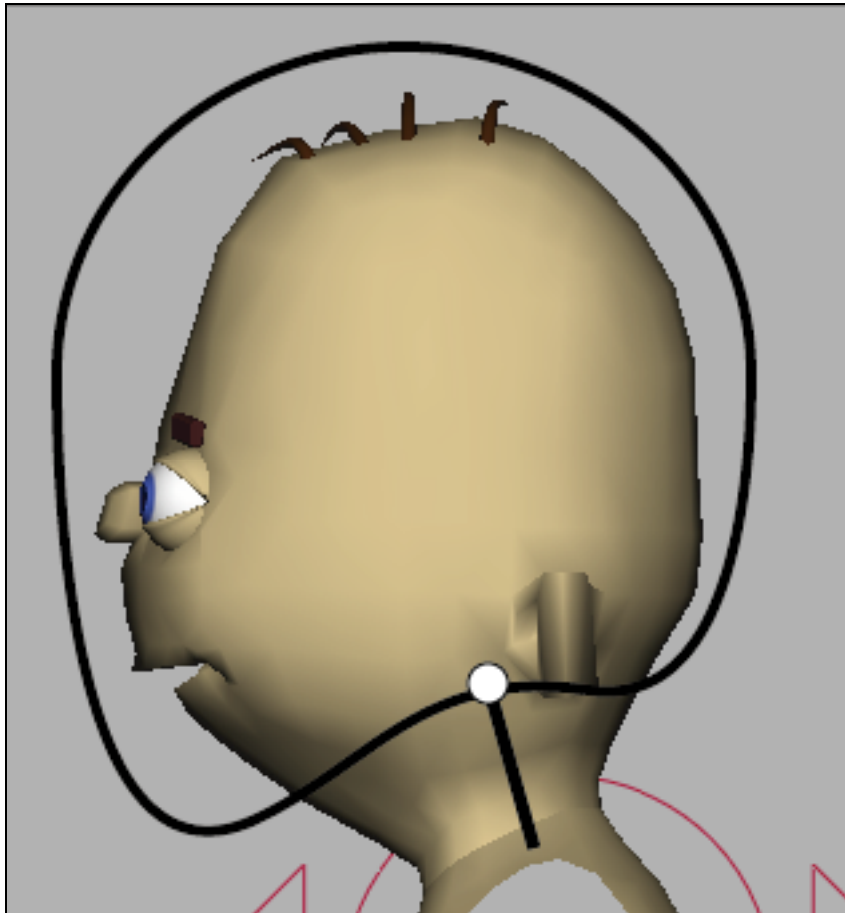
Figure 195 - hide geometry that we don't need for the neck rig



*Figure 196 - the rest of JJ*

In order to create the head rig, we're going to need to dice up the head and neck into a few pieces.

First, we'll need a section of just the head. Then, we'll need a few slices of neck to go with the joint structure we have in mind.



*Figure 197 - the location of the head pivot*

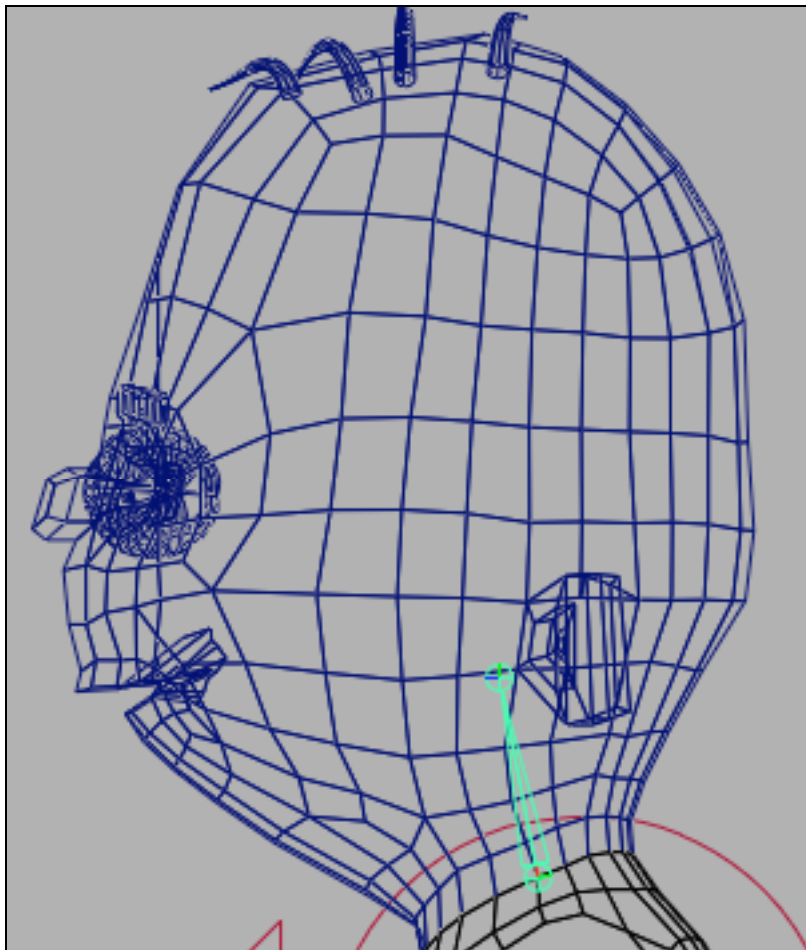
Choosing the location of where to pivot the head is an important one. You want the head to appear natural when it looks up and down, but you also want to make it easy for the animator to isolate the head to get the kind of motion they desire. If the joint is too far down, then the head will appear top heavy when they rotate it. If the joint is too high, the head will look like it's swinging around the middle of the skull.

It's okay to play with different locations until you get exactly what works best for your character.

#### Segment the Head

**1. Create a joint structure that will represent the location of the head pivot.**

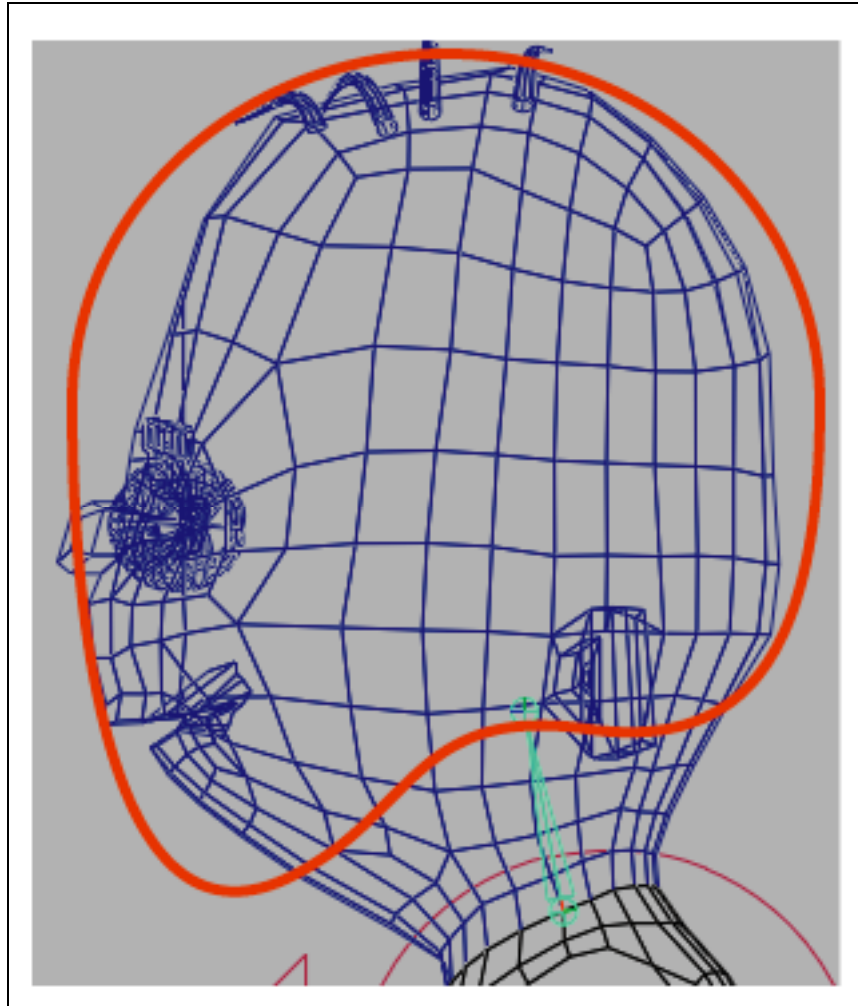
- Choose **Skeleton > Joint Tool**
- Click once at the base of the neck, and then again just in front of and a tiny bit lower than the ear. This will be the main joint segment for the neck.



*Figure 198 - Creating the main joint structure for the neck*



Based on the location of the joint, we can now decide how best to segment the head for animation.



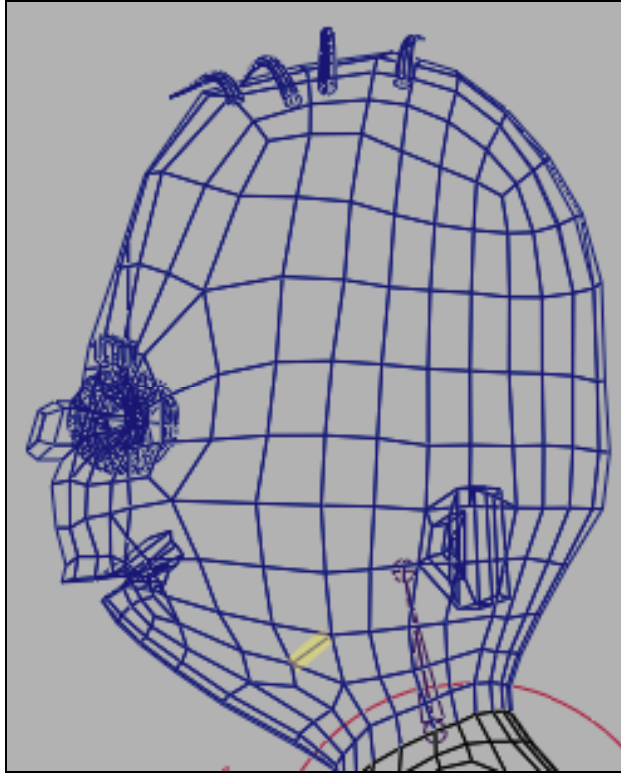
*Figure 199 - The area of the head that we want to extract from the neck*

If you look at the above image, you can see that the flow of the polys in the head don't allow us a very good place to cut the head where we'd want to. So we'll need to split some of the polygons to allow us the ability to cut in the right places.

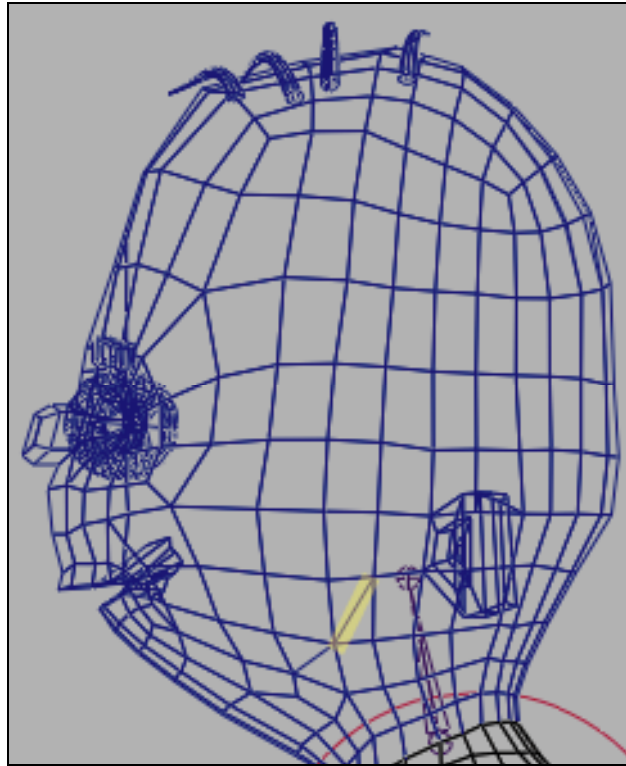
#### 28. Split the head

- Choose **Mesh > Split Polygon Tool**

- Split the polygons in the head following the images:

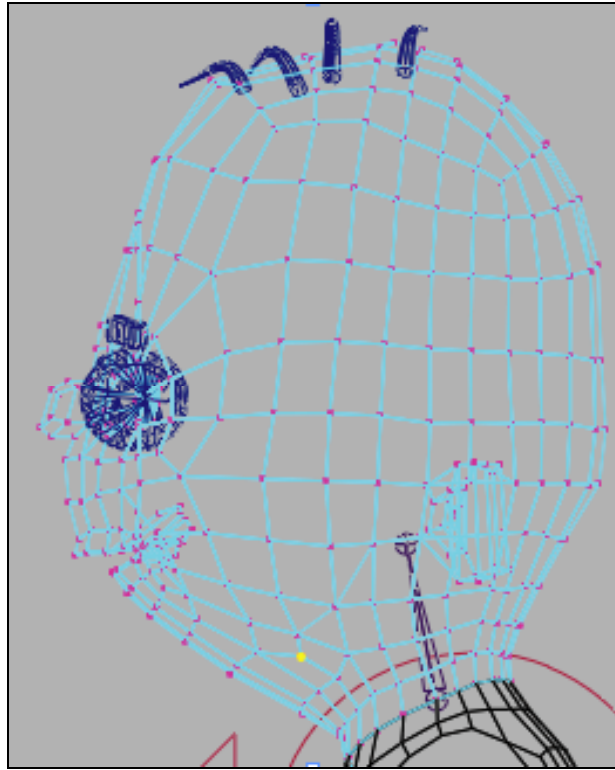


*Figure 200- Split #1*

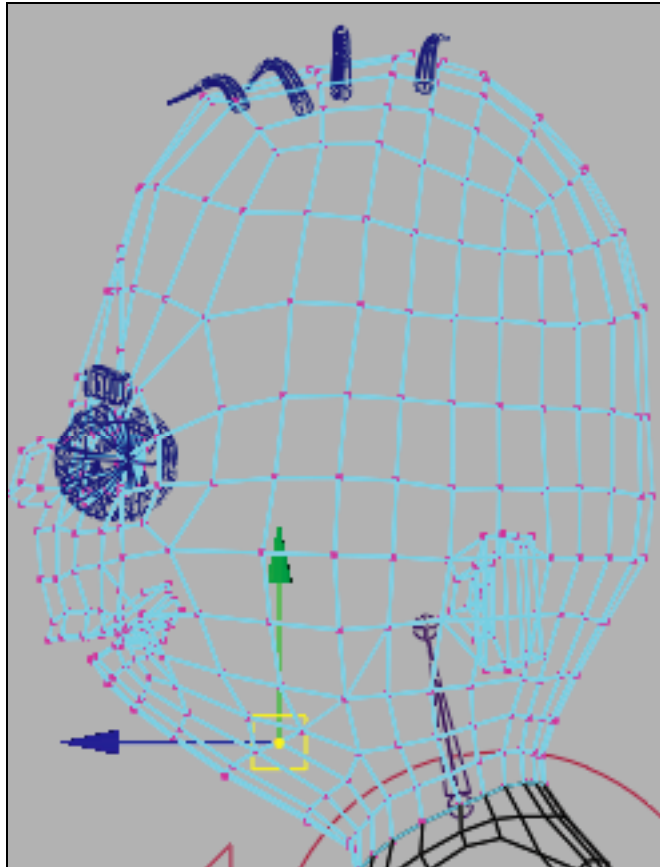


*Figure 201 - Split #2*

- Now fix that diamond shaped polygon next to the split by moving the vert.

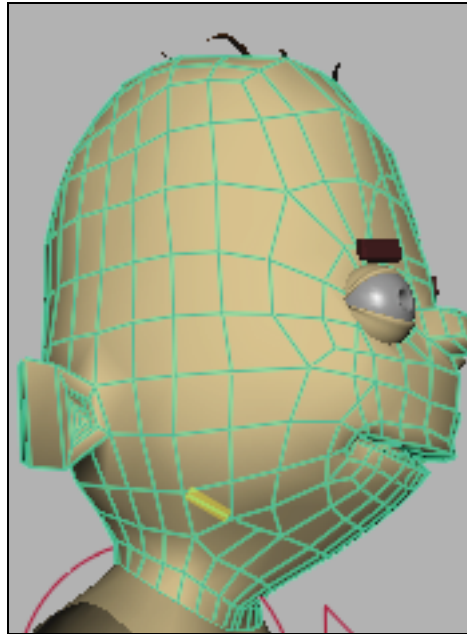


*Figure 202 - Vert selected to fix diamond shaped polygon*

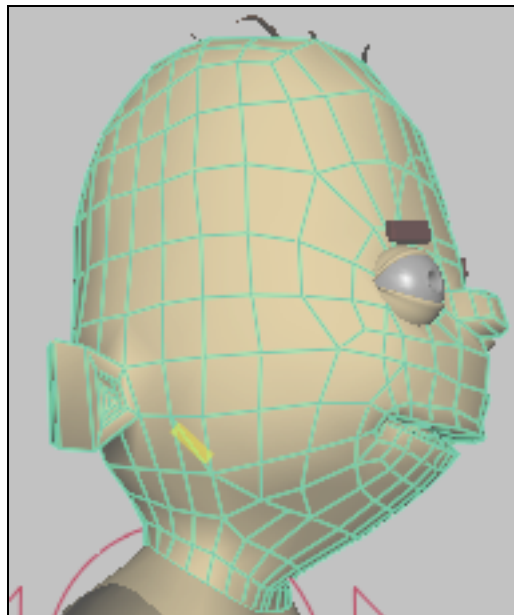


*Figure 203 - Vert adjusted so diamond shaped polygon is now correct*

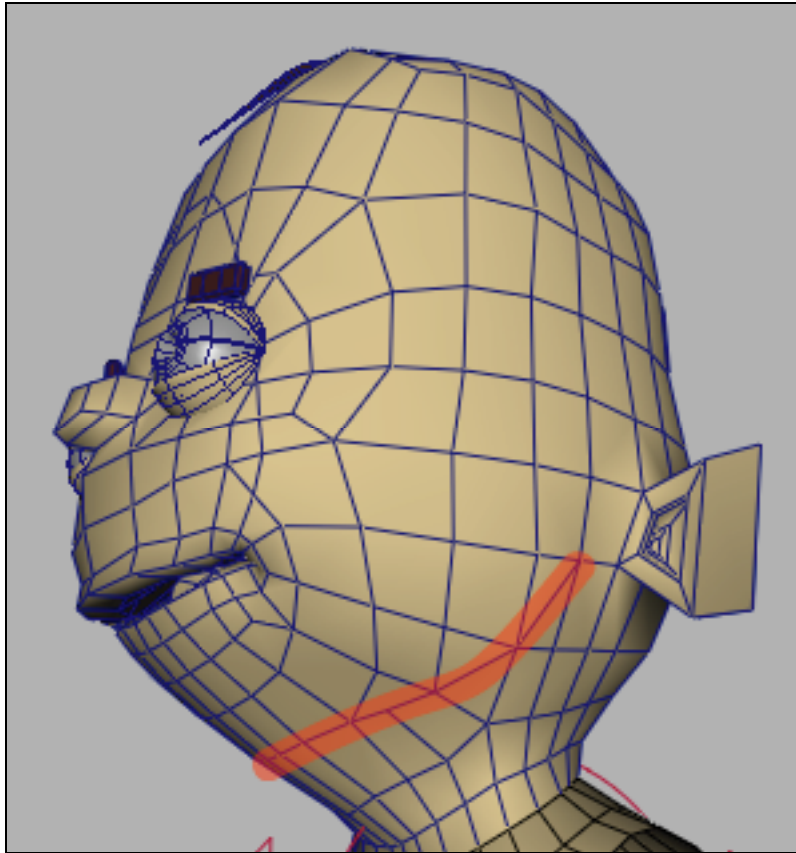
- Continue splitting polygons on the other side of the head. Use the selected vert on the diamond shaped head as a reference as to where to start.



*Figure 204- The next split on the other side of the head to match the split on the first side.*



*Figure 205 - the final split.*



*Figure 206 - The split and lined up verts*

#### 29. Select the faces for the head

- Select the **head\_geo** and hit the **F8** hotkey to switch to component mode.
- Select the **faces for component** toggle in the **status line**
- Select all the faces above the cut line we just made, including the ears.



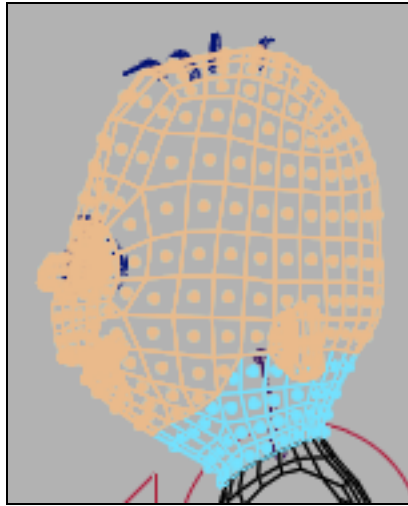


Figure 207 - Faces above the cut line selected (including ears)

#### 30. Extract the faces

- Choose **Mesh > Extract > Option Box**
- Set **Separate Extracted Faces** to on

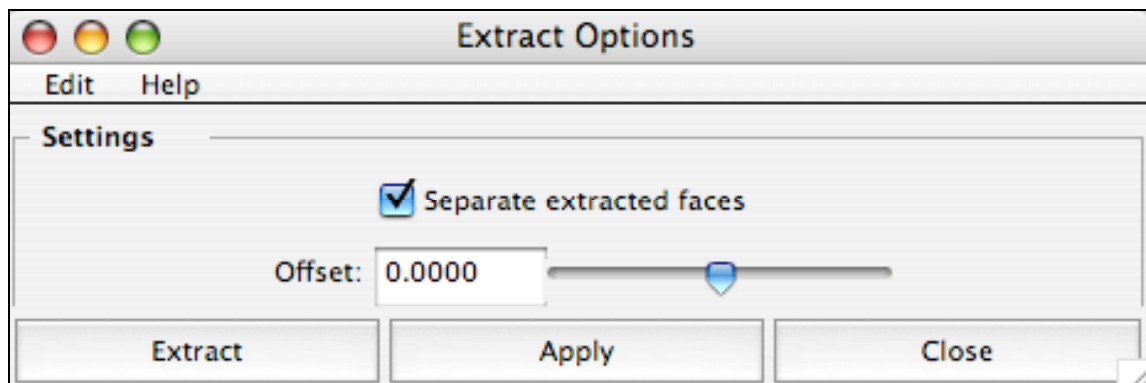


Figure 208 - Extract Face Options

- Click **Extract**



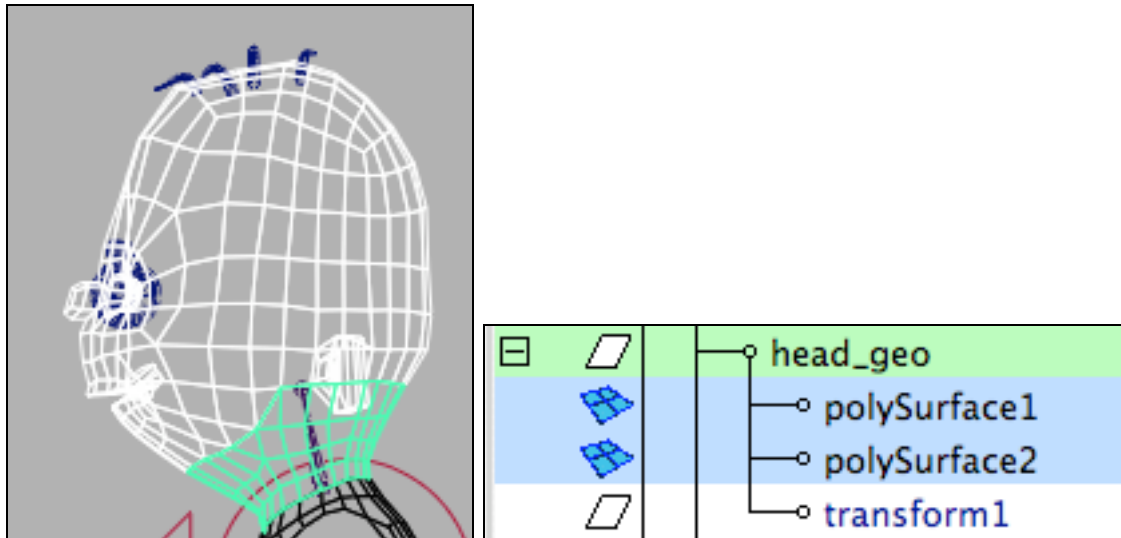


Figure 209 - Faces Extracted

#### 31. Delete History on the pieces

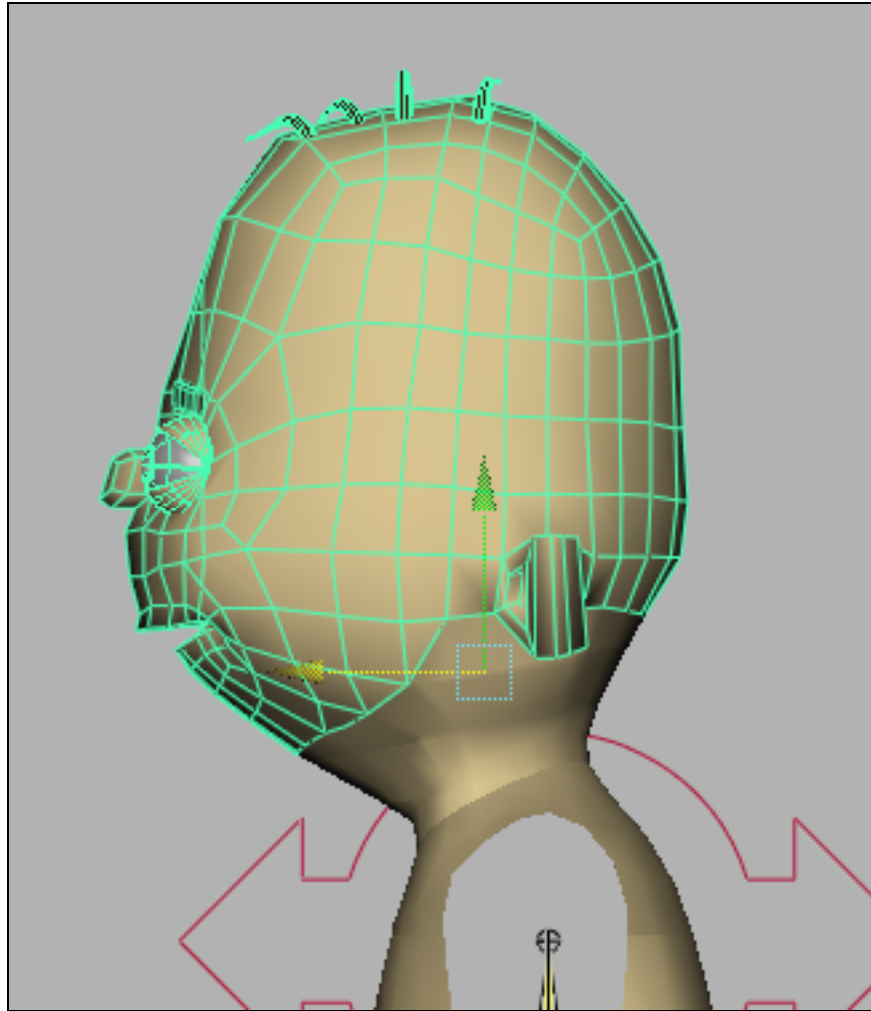
- Select **polySurface1** and **polySurface2**
- Choose **Edit > Delete by type > History**

#### 32. Rename the geometry

- Rename **polySurface1** **head\_geo**
- Rename **polySurface2** **neck\_geo**

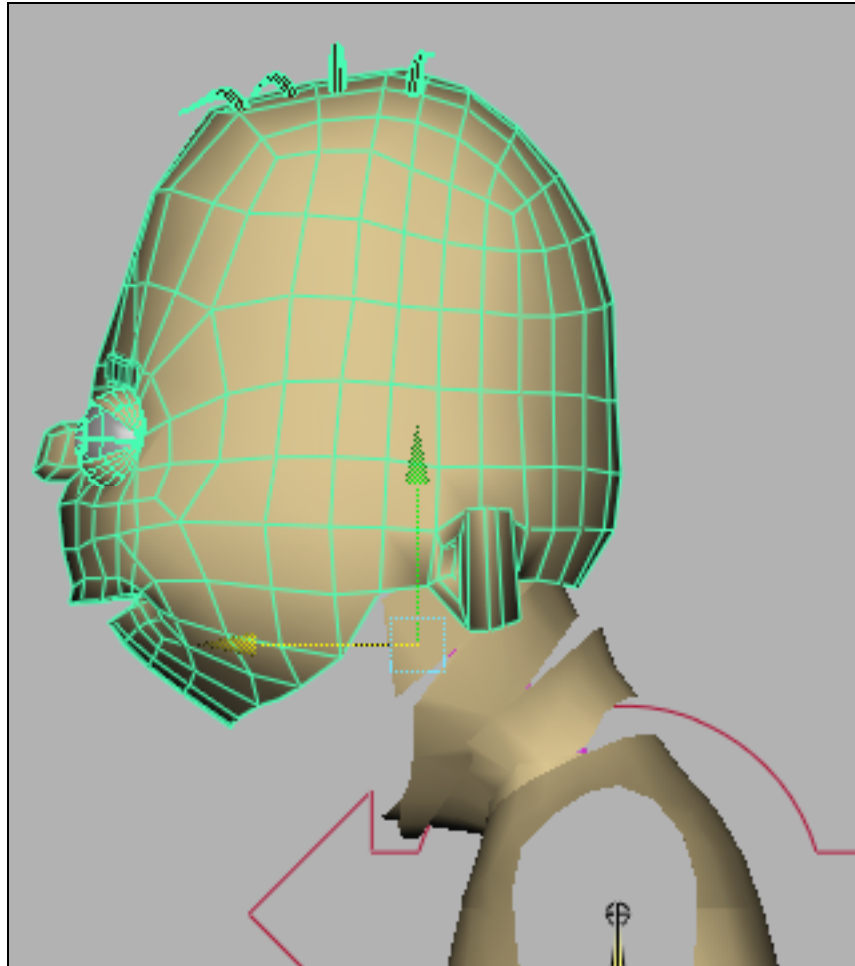
Normally we would segment out each piece of geometry for the neck as well as the head. But in this case, the geometry for our character doesn't lend itself well to a broken apart neck.

Let's take a look at what would happen if we broke apart the neck geometry. The first image is the head in it's default position. Looks perfectly fine, right?



*Figure 210- Head in default position*

Now look at what happens when the head is translated forward (*Figure 211*). Notice how the pieces of the neck break apart? This just looks bad! It becomes distracting to the animator that there will be a huge gaping hole between the chin and the neck.



*Figure 211 - Head translated forward.. with broken neck nastiness!*

So instead of breaking the neck apart for this rig, we're going to skin the neck quickly using Maya's smooth skinning. Note, this is *not* the final skinning that the animator should expect to see, but it will give them an idea of what the neck **might** look like, and it won't be as distracting as a segmented neck.

In addition, if it slows down the rig too much, we can give the animator the option of turning off the neck display. That should speed things right up!

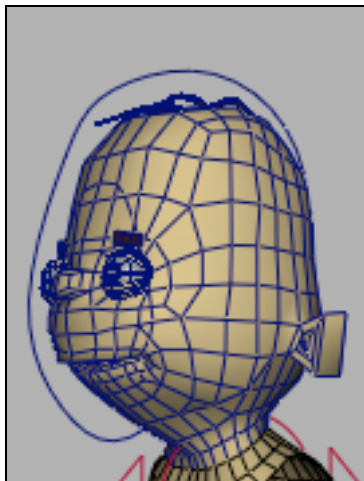
*Example File: jj\_neck\_rig\_wip\_v2.ma*

#### Create the head Control

Before making the neck fit to the head, we should create the main head control that we'll use for our character.

Again, we want a head control that will allow the animator to recognize that this control is for the head, but one that isn't too irritating to look at.

One option is to simply create a curve that circles around the head like this:



*Figure 212 - nurbs curve that circles the head*

Again, you can create any type of shape that you like. Personally, I think that this curve is simple and easy to create. We'll go with it .

#### 1. Create a control for the head

- Go to a side view
- Choose **Create > EP Curve tool**
- Create a curve that's roughly the shape of the head.

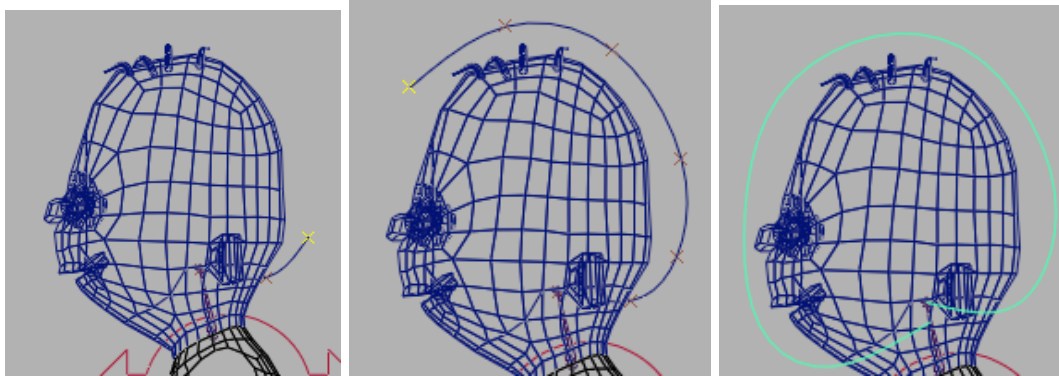


Figure 213 - creating a head control curve

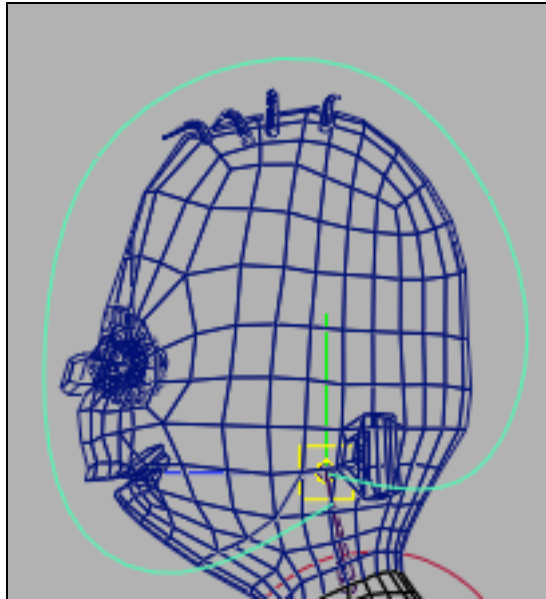
#### 33. Move the pivot to the location of the top joint

- Switch to the **Move** tool
- Hit the **Ins** (**home** on **OS X**) key to allow you to modify the pivot
- Hold down the **v** hotkey to turn on **point snap**
- Click and drag with the **middle mouse button** near **joint2** to snap the pivot there.
- Hit **Ins** (or **home**) to return to normal mode.

- Or -

- You can use the **js\_copyPivot** button on the shelf. To do this, select **joint2**, then select the curve and click the **js\_copyPivot** button. The pivot will automatically be copied to the location of joint2.





*Figure 214- Moving the pivot to the base of the head*

#### 34. Rename the control

- Rename **curve1** to **head\_anim**

#### 35. Group all the head bits together

- Select **l\_brow\_geo**, **r\_brow\_geo**, **hair\_\*\_geo**, **l\_eyeGrp**, **r\_eyeGrp**, and **head\_geo** (the geometry, not the mis-named group).
- Group them together using **ctrl+g**

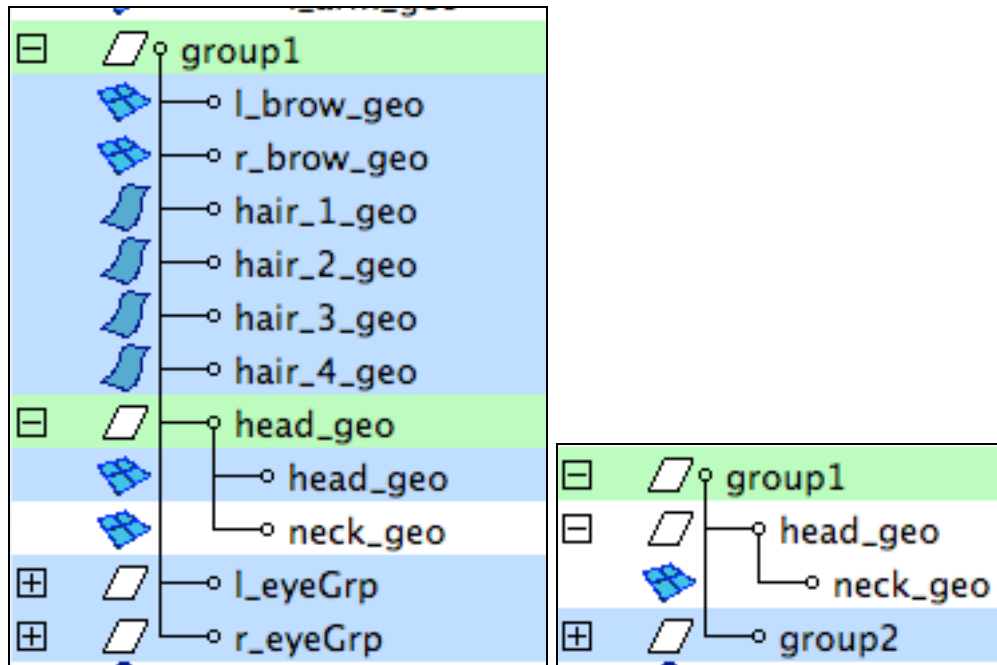


Figure 215 - Grouping all the head bits together into one group

#### 36. Rename group1

- Rename **group1** to **head\_geo\_grp**

#### 37. Match Rotation Orders

- Select **head\_geo\_grp** and **head\_anim**
- Load the **Change Rotation Order On Selected Objects** button on the **Animator Friendly Rigging** shelf.
- Set the rotation orders to **ZXY**, the same as **shldr\_anim**.

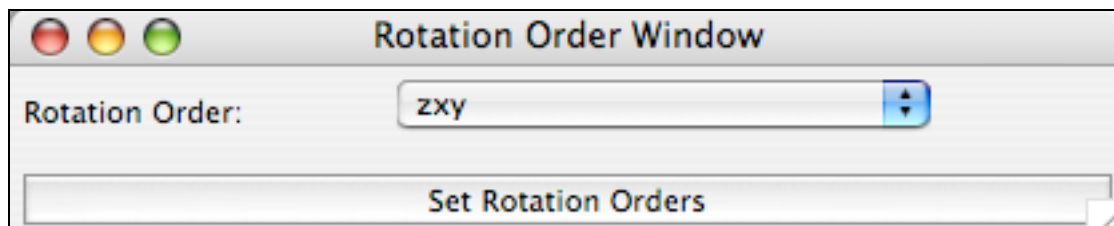


Figure 216 - Rotation Orders Matching

#### 38. Match Pivots

- Select **head\_anim** and then **head\_geo\_grp**
- Click on the **Match Pivots** icons in the **Animator Friendly Rigging** shelf to copy the pivot location from **head\_anim** to **head\_geo\_grp**.



#### 39. Parent Constrain head\_geo\_grp to head\_anim

- Select **head\_anim** and then **head\_geo\_grp**
- Choose **Constrain > Parent**

The head geometry will now move around with the animation control **head\_anim**.

*Example File: jj\_neck\_rig\_wip\_v3.ma*

#### Add Spline IK To the Neck

Now we get to break up the neck into tiny parts and put the spline ik in there to make it bend nicely.

##### 1. Break the neck into 3 parts

- Select **joint1**
- Click on the **Split Selected Joint** button in the **Animator Friendly Rigging** shelf
- Set **Segments** to **3**.
- Click **Okay**



#### 40. Rename the neck joints

- Rename **joint1** to **neck\_base\_joint**
- Rename **joint1\_seg\_1** to **neck\_1\_joint**
- Rename **joint1\_seg\_2** to **neck\_2\_joint**
- Rename **joint2** to **neck\_end\_joint**



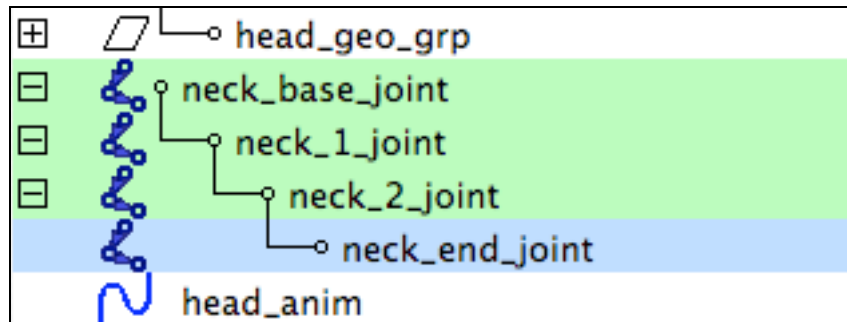


Figure 217 - Neck joints renamed

#### 41. Create spline ik

- Choose **Skeleton > Spline Ik Handle Tool**
- Click on **neck\_base\_joint** and **neck\_end\_joint**

#### 42. Rename Spline Ik Parts

- Rename **ikHandle1** to **neck\_ikHandle**
- Rename **curve1** to **neck\_curve**

#### 43. Add Advanced Twist Controls

- Select **neck\_ikHandle**
- Open the **Attribute Editor**
- Open the **ikSolver Attributes**
- Open **Advanced Twist Controls**
- Set the following options:

<b>Enable Twist Controls</b>	<b>on</b>
<b>World Up Type</b>	<b>Object Rotation Up [start   end]</b>
<b>Up Axis</b>	<b>Positive Y</b>
<b>Up Vector</b>	<b>0 0 -1</b>
<b>Up Vector 2</b>	<b>0 0 -1</b>
<b>World Up Object</b>	<b>shldr_anim</b>
<b>World Up Object 2</b>	<b>head_anim</b>

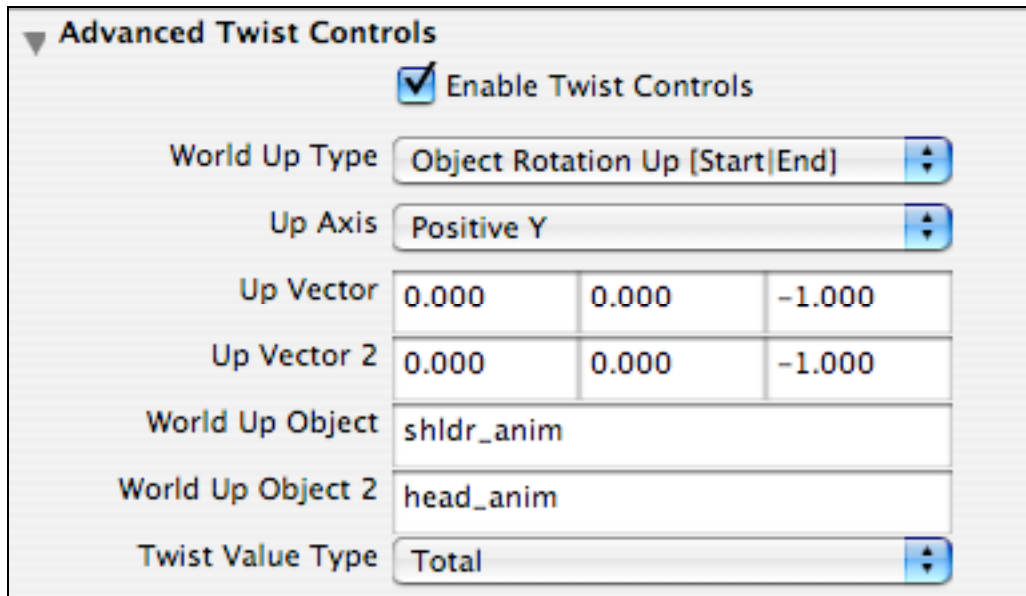
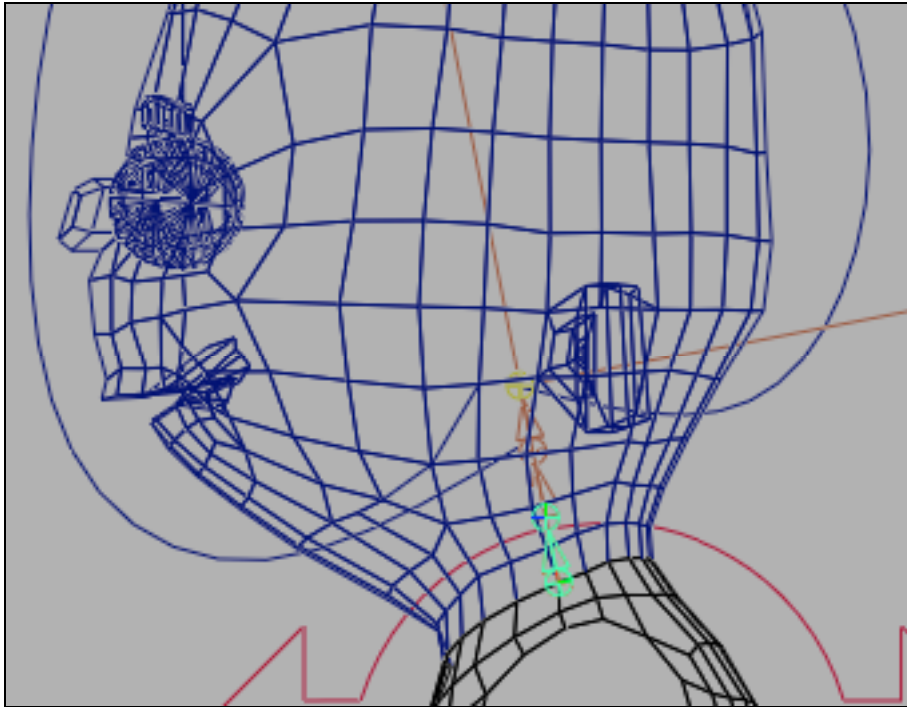


Figure 218 - Advanced Twist Controls Settings for the Neck\_ikHandle

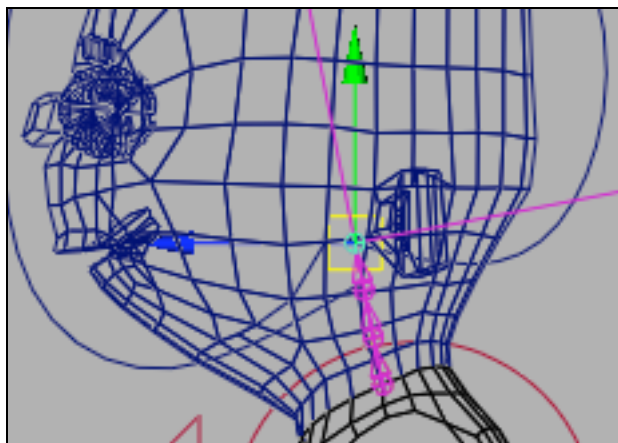
#### 44. Create joints to control neck\_curve

- Choose **Skeleton > Joint Tool**
- While holding down the **v** hotkey for point snap, click and drag near **neck\_base\_joint**.
- Then click and drag again near **neck\_1\_joint**. You should have a joint segment that looks like the following:



*Figure 219 - bottom curve control joints*

- Hit the **y** hotkey to accept.
- Holding down **v** again for point snap, click and drag near the end of the neck joints to create a joint for the top of the curve.



*Figure 220- top curve joint*

#### 45. Rename the joints

- Rename **joint1** to **neck\_curve\_base\_bind**
- Rename **joint2** to **neck\_curve\_1\_bind**
- Rename **joint3** to **neck\_curve\_top\_bind**

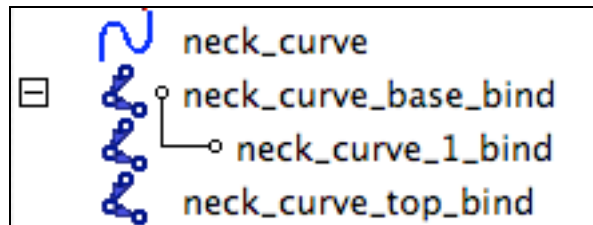


Figure 221 - renamed curves

#### 46. Skin the curve to the joints

- Select **neck\_curve\_base\_bind**, **neck\_curve\_1\_bind**, **neck\_curve\_top\_bind**, and **neck\_curve**
- Choose **Skin > Bind Skin > Smooth Bind**

#### 47. Apply the stretch to the neck

- Select **neck\_curve**
- Click on the **Create Stretchy Spline** curve in the **Animator Friendly Rigging Shelf**
- Set the following options:

**Nurbs Curve:** **neck\_curve**

**Maintain Volume:** **off**

**World Scale:** **on**

**Node:** **all\_anim**

**Attribute:** **globalScale**



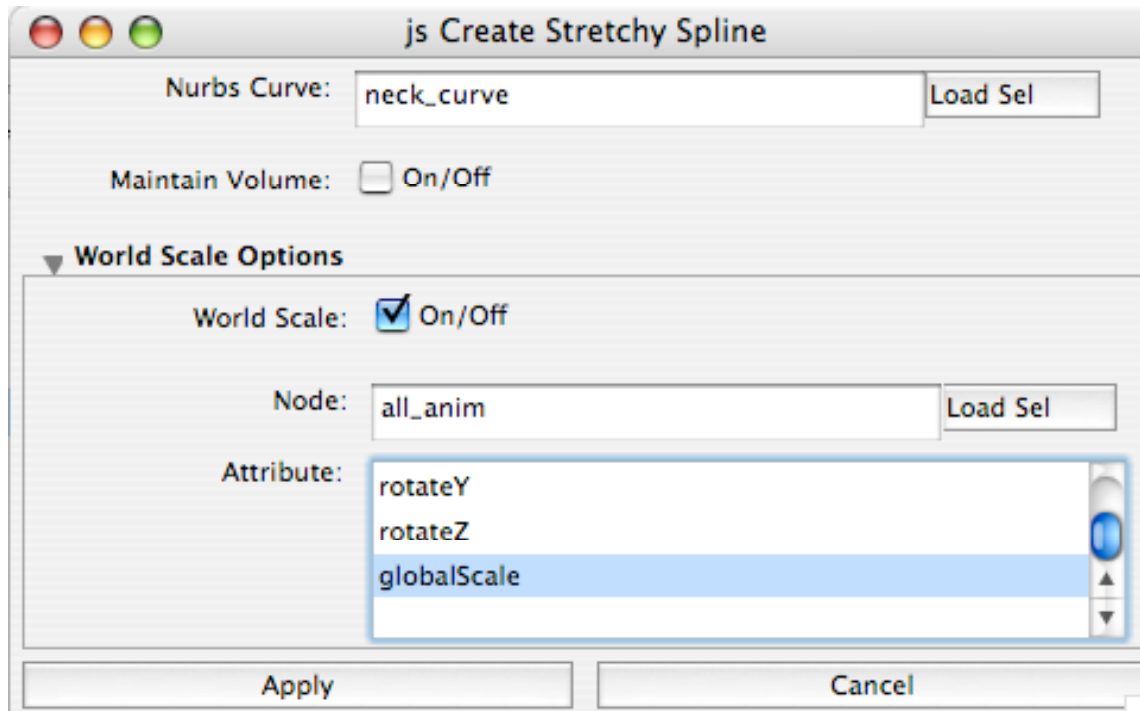


Figure 222 - js Create stretchy spline options

The neck will now stretch when we move the neck\_curve\_bind joints.

#### 48. Parent neck\_curve\_bind joints to controls.

- Parent neck\_curve\_top\_bind to head\_anim
- Parent neck\_curve\_base\_bind to shldr\_anim

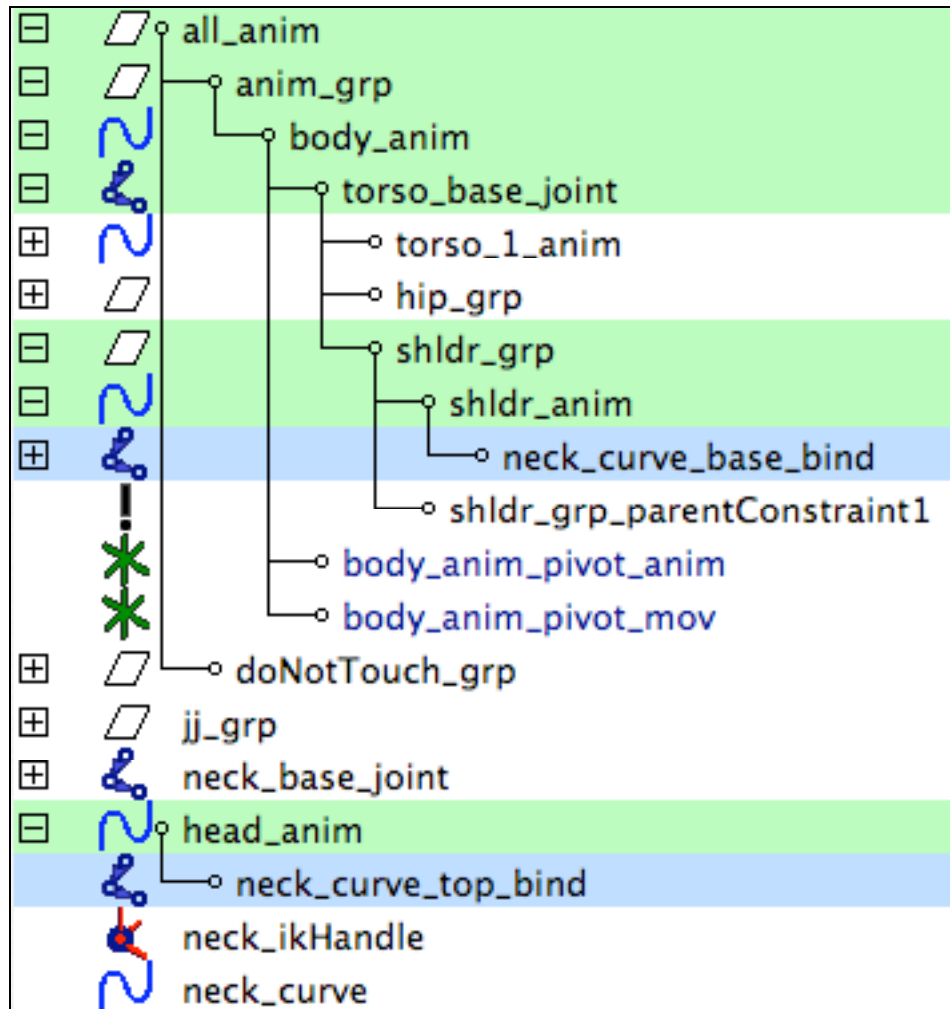


Figure 223 - Parenting the bind joints

Now as you move the head and body around, you should see the neck joints moving with the head and shoulders. Our next step should be pretty obvious.. make the *neck geometry* move, too!

*Example File: jj\_neck\_rig\_wip\_v4.ma*

#### Skin the neck

Our next step is to skin the neck geometry so it just blends between the head and body. The goal *isn't* to create the best skinning possible, it's to create something that isn't distracting to the animator.

#### 1. Smooth Skin the neck geometry

- Select **neck\_base\_joint** and **neck\_geo**
- Choose **Skin > Bind Skin > Smooth Bind > Option Box**
- Make sure the settings match the following image:

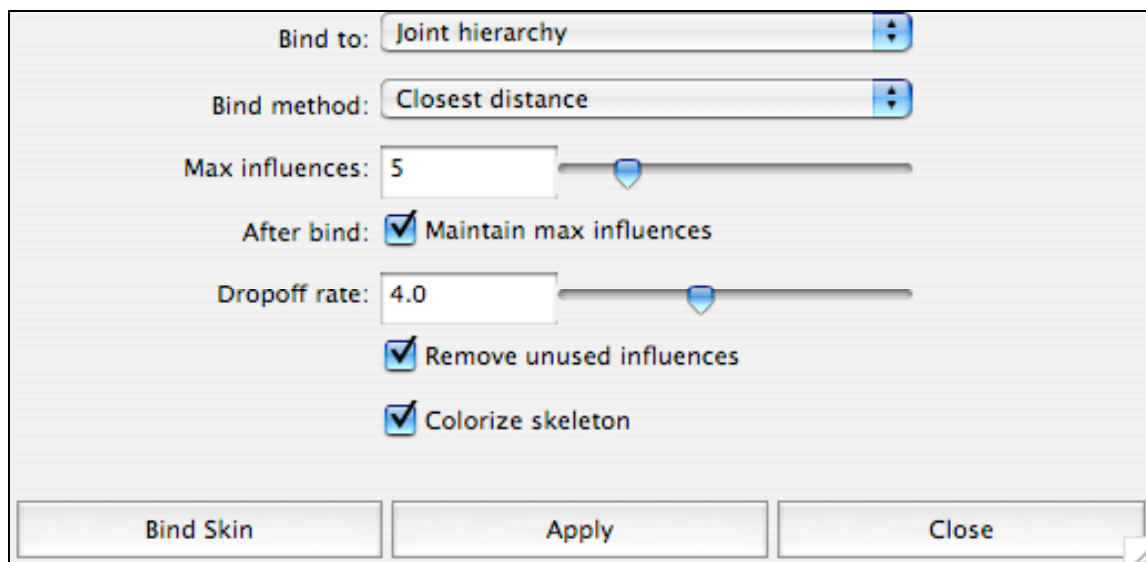


Figure 224 - Bind Skin Settings

- Click **Bind Skin**

The skin is now bound, but as you can tell, it doesn't do the best job by default

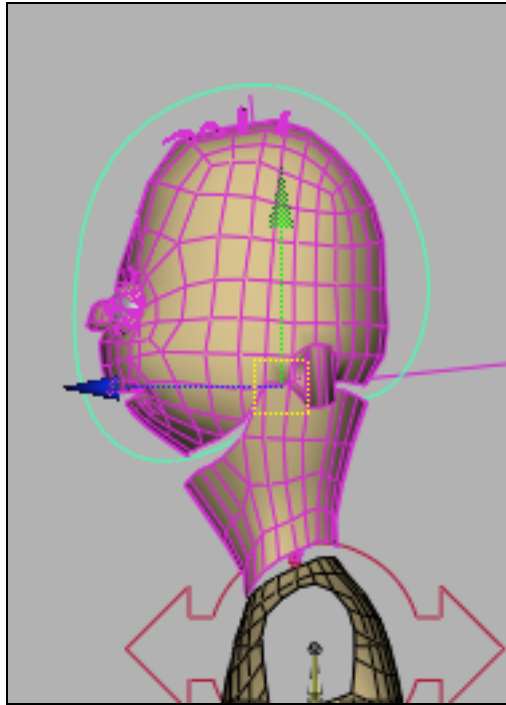


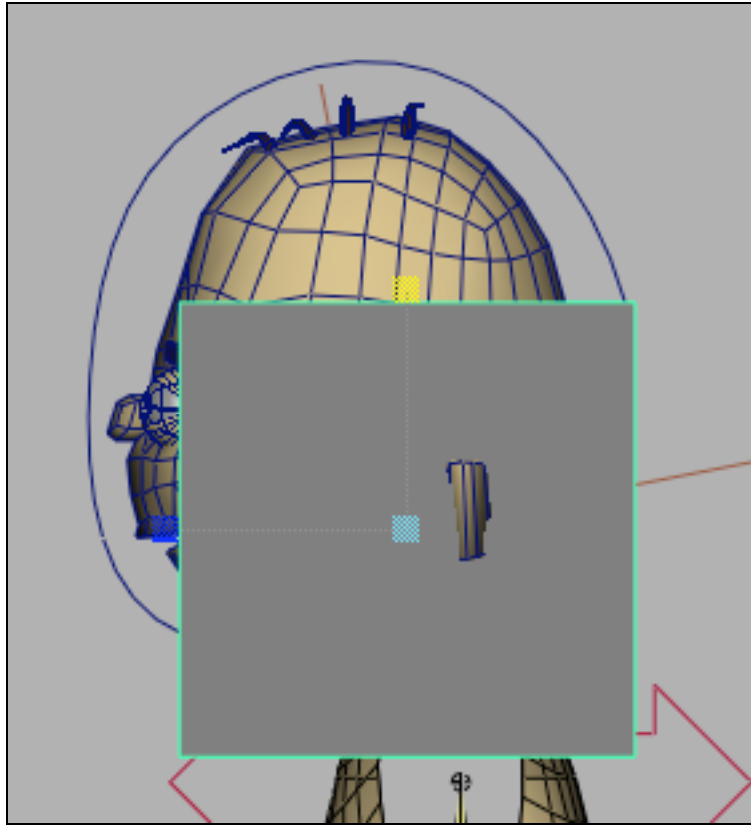
Figure 225 - Default bind skin

So what we want to do is fix the skinning by using a couple of *influence objects* to help make it easier. An influence object is basically a piece of geometry that can also influence the skinning. They're extremely helpful when adding volume control to skinning solutions.. and in fact, can even be used to simulate *muscles* if enough care is taken in creating and controlling them. In our case, we're just going to use them to help us get nice clean skinning on our neck.

#### 49. Add Influence Objects

- Create a polygon cube by choosing **Create > Polygon Primitives > Cube**
- Position the cube at the top of the neck





*Figure 226 - Cube positioned at the top of the neck*

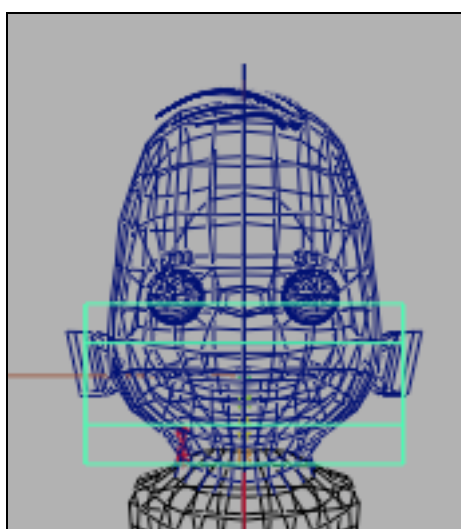
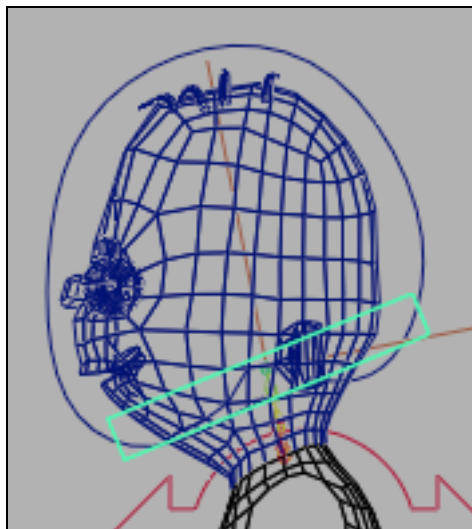
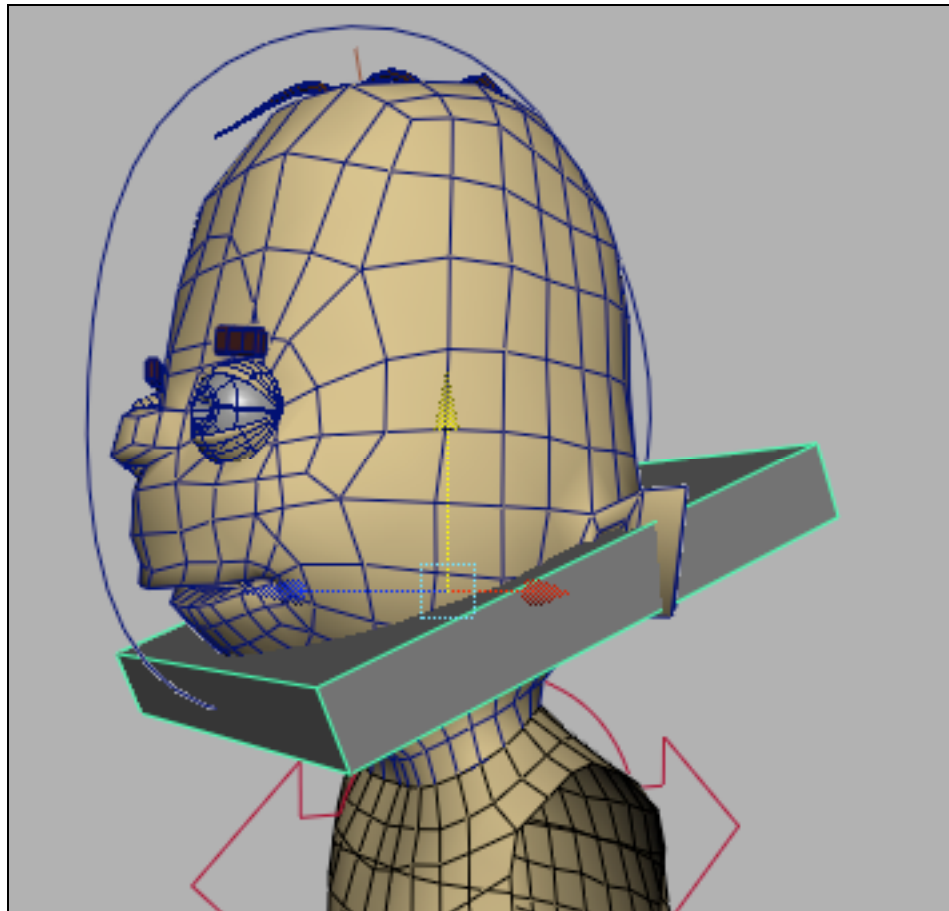
- Scale and rotate the cube until it fits around the base of the head, where the neck geometry and the head geometry join together.



Master Classes

## Instructor Notes – Animator Friendly Rigging Part II

Jason Schleifer

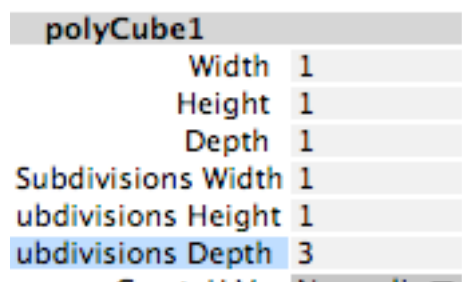


Autodesk® Maya® Master Classes – Instructor Notes

SIGGRAPH™ 2006

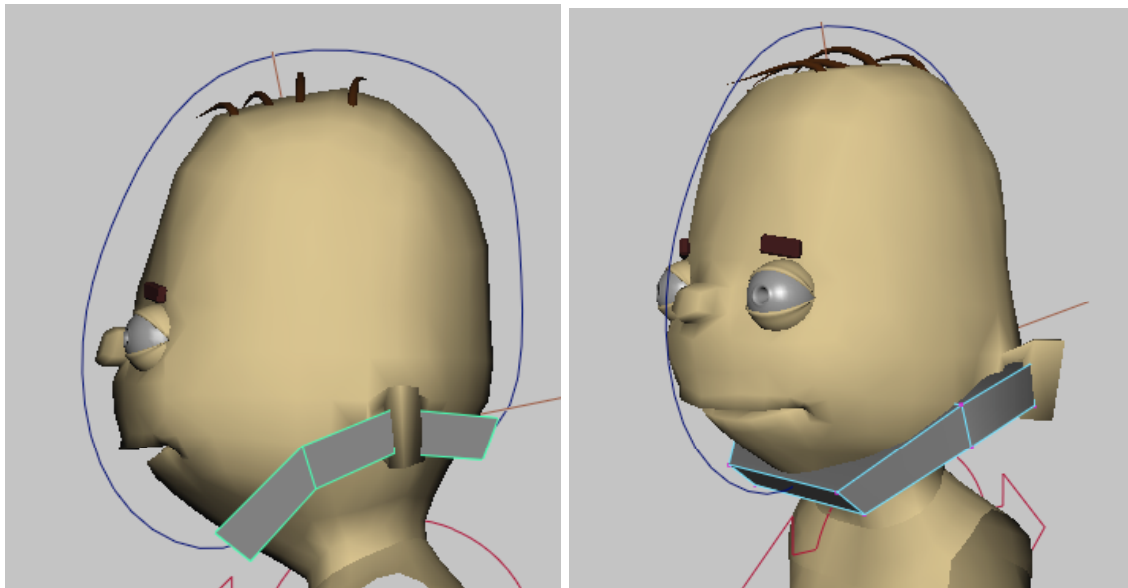
*Figure 227 - cube positioned at the base of the head and the top of the neck*

- You can also subdivide the cube and modify the vertices themselves to make a better fit.
- To do that, modify the subdivisions Depth attribute in the channel box.



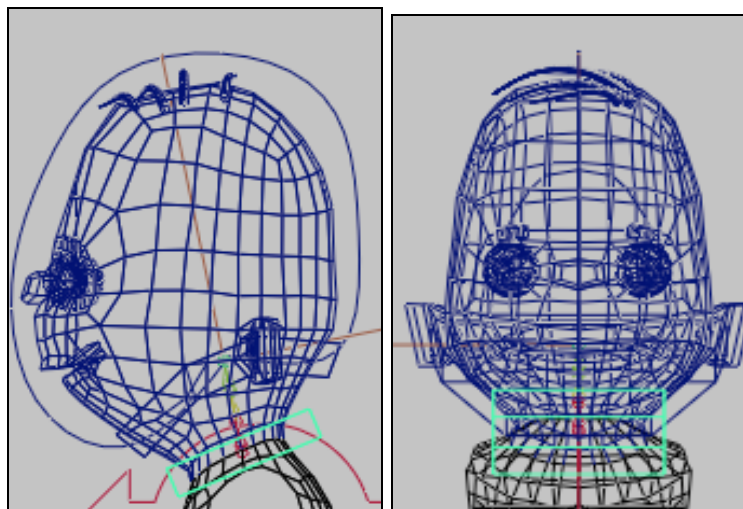
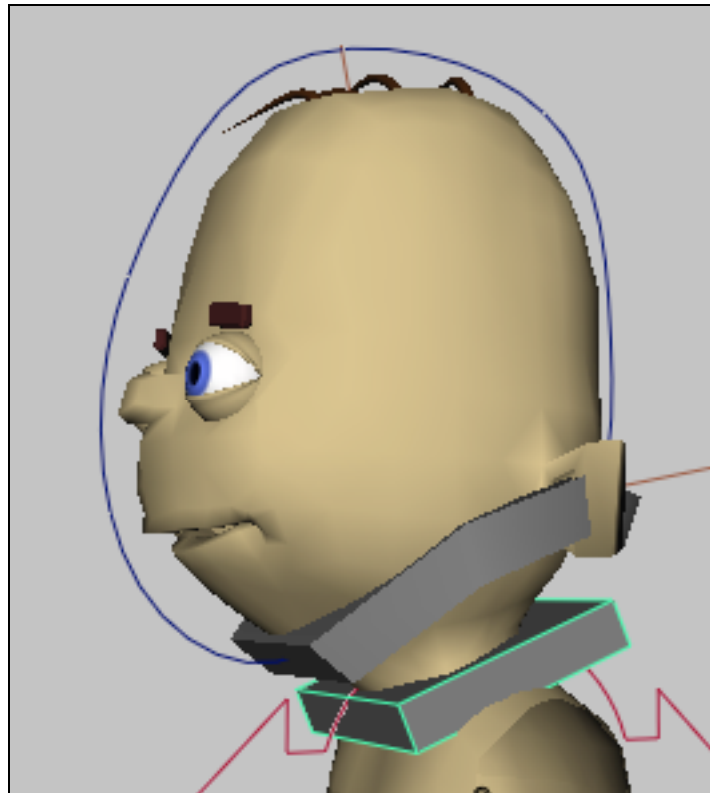
*Figure 228 - subdivisions depth attribute for the poly cube.*

- Then manipulate the verts until it fits the base of the head better.



*Figure 229 - shaping the geometry for the neck*

- Create another cube and then position it around the base of the neck.



*Figure 230 - bottom neck cube*

#### **50. Rename the cubes**



## Master Classes

# Instructor Notes – Animator Friendly Rigging Part II

Jason Schleifer

- Rename **pCube1** to **neck\_top\_inf**
- Rename **pCube2** to **neck\_bot\_inf**

### 51. Add the cubes as influence objects

- Select **neck\_geo** and **neck\_top\_inf**
- Choose **Skin > Edit Smooth skin > Add Influence**
- Select **neck\_geo** and **neck\_bot\_inf**
- Choose **Skin > Edit Smooth Skin > Add Influence**

### 52. Parent influence cubes

- Parent **neck\_top\_inf** to **head\_anim**
- Parent **neck\_bot\_inf** to **shldr\_anim**

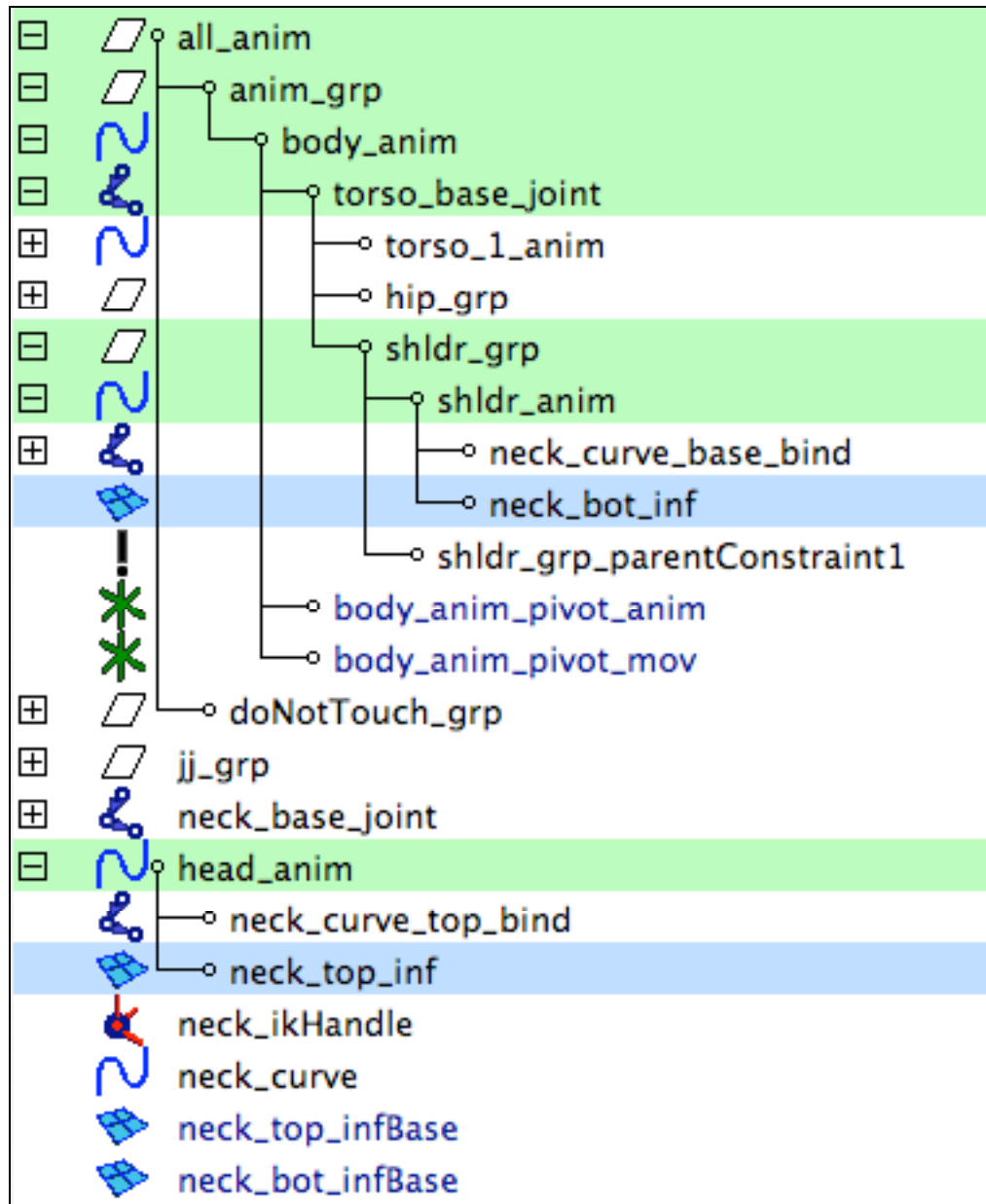
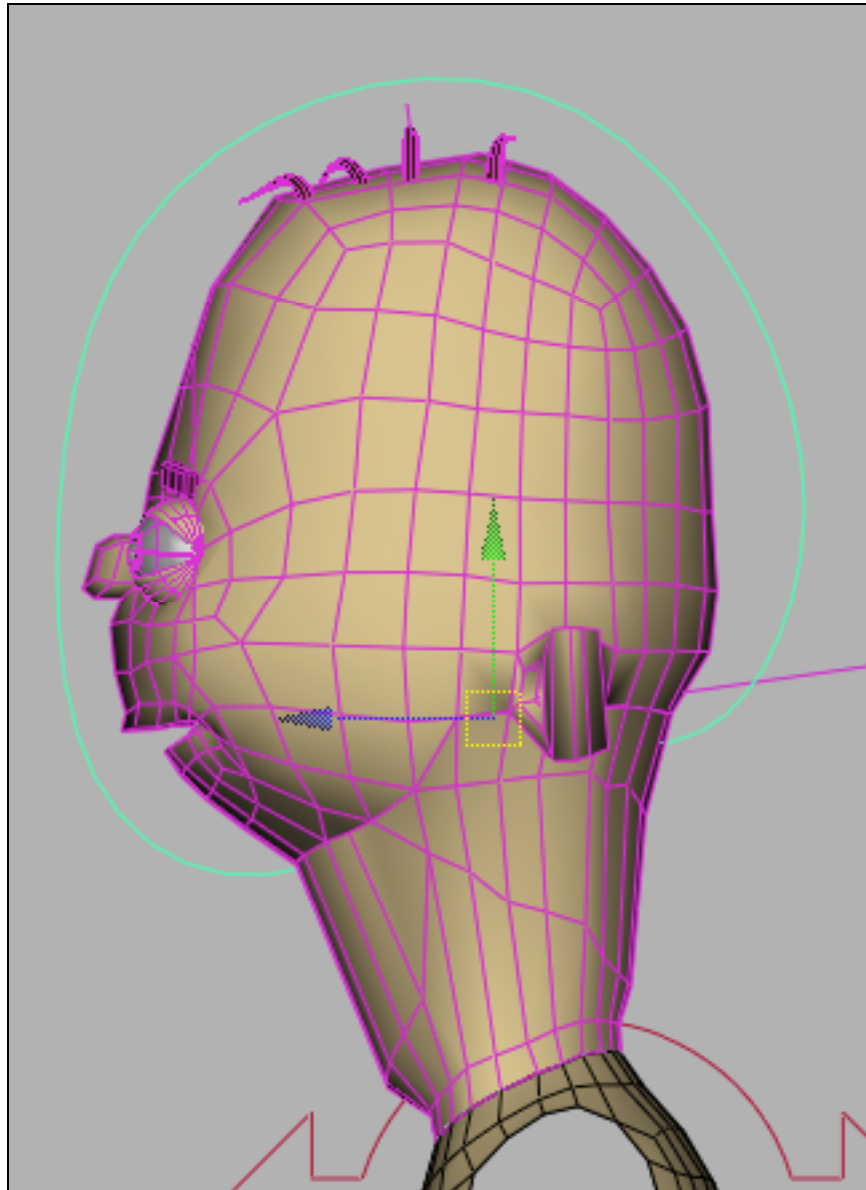


Figure 231 - Influence Geometry Parented

#### 53. Hide influence cubes

- Hide **neck\_bot\_inf** and **neck\_top\_inf**

The skinning is better, but it needs a bit of tweaking. Notice the harsh angles. We want something that looks a bit smoother, so we'll paint the weights to get exactly what we want.

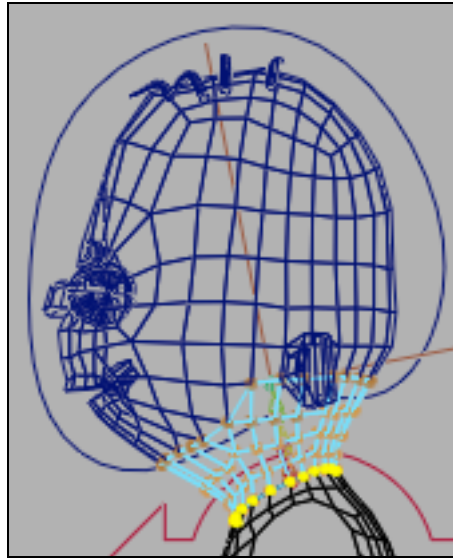


*Figure 232 - Improved skinning*

#### **54. Adjust Skin Weights**

Our goal is to make sure that the bottom row of verts are all 100% controlled by neck\_bot\_inf, and that the top row is 100% controlled by neck\_top\_inf. All the other verts should be smoothly deformed based on what they've been assigned.

- Switch to Component mode by selecting **neck\_geo** and hitting **F8**.
- Select the **bottom row of verts** on the **neck\_geo**.



*Figure 233- bottom row of verts selected*

- Choose **Window > General Editors > Component Editor**
- Find the **Smooth Skins** tab, and then the **neck\_bot\_inf** column. Change all the values in that column to **1**.



# Autodesk®

## Master Classes

### Instructor Notes – Animator Friendly Rigging Part II

Jason Schleifer

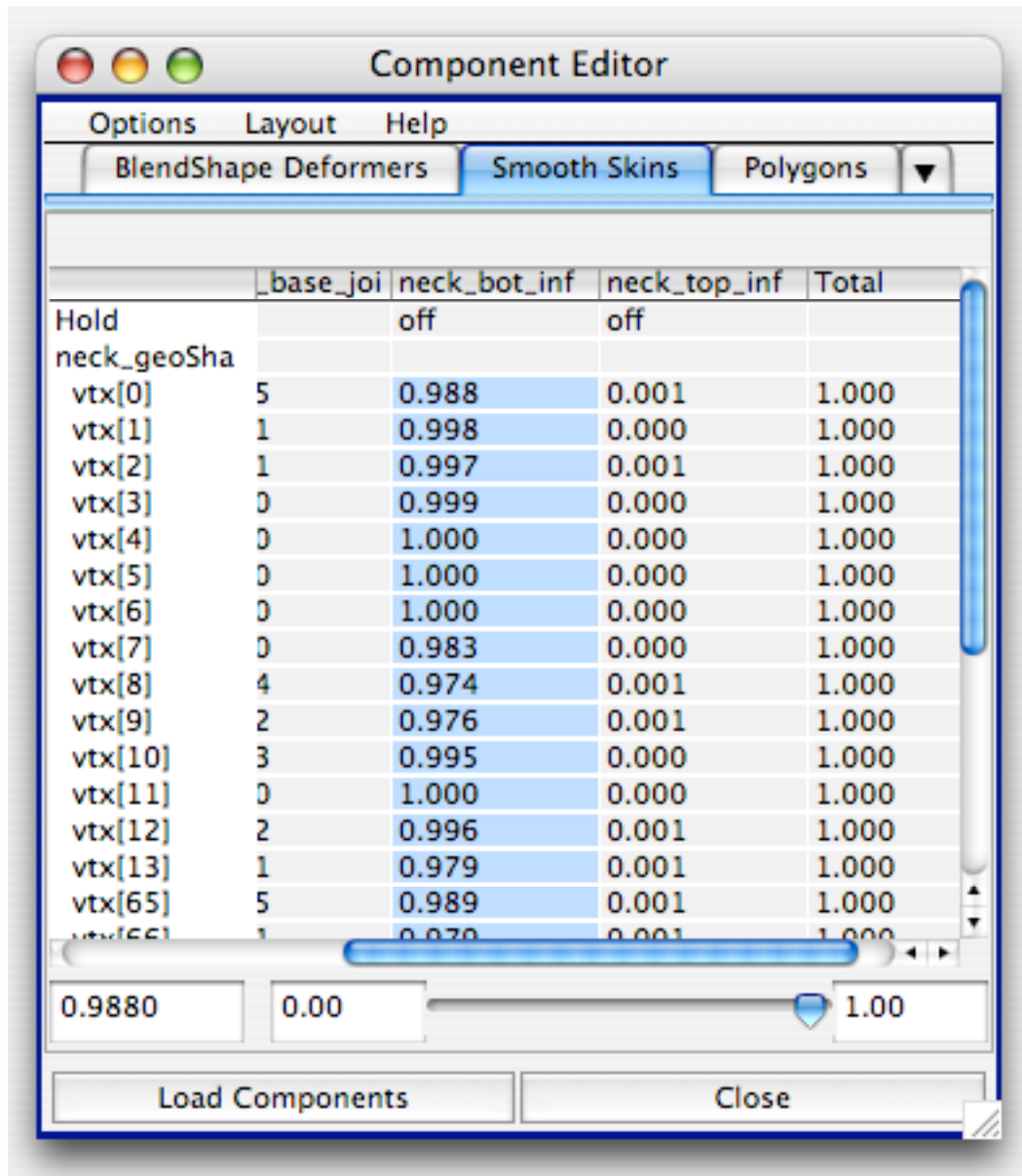


Figure 234 - Component editor, showing the skin weights of the bottom row of verts

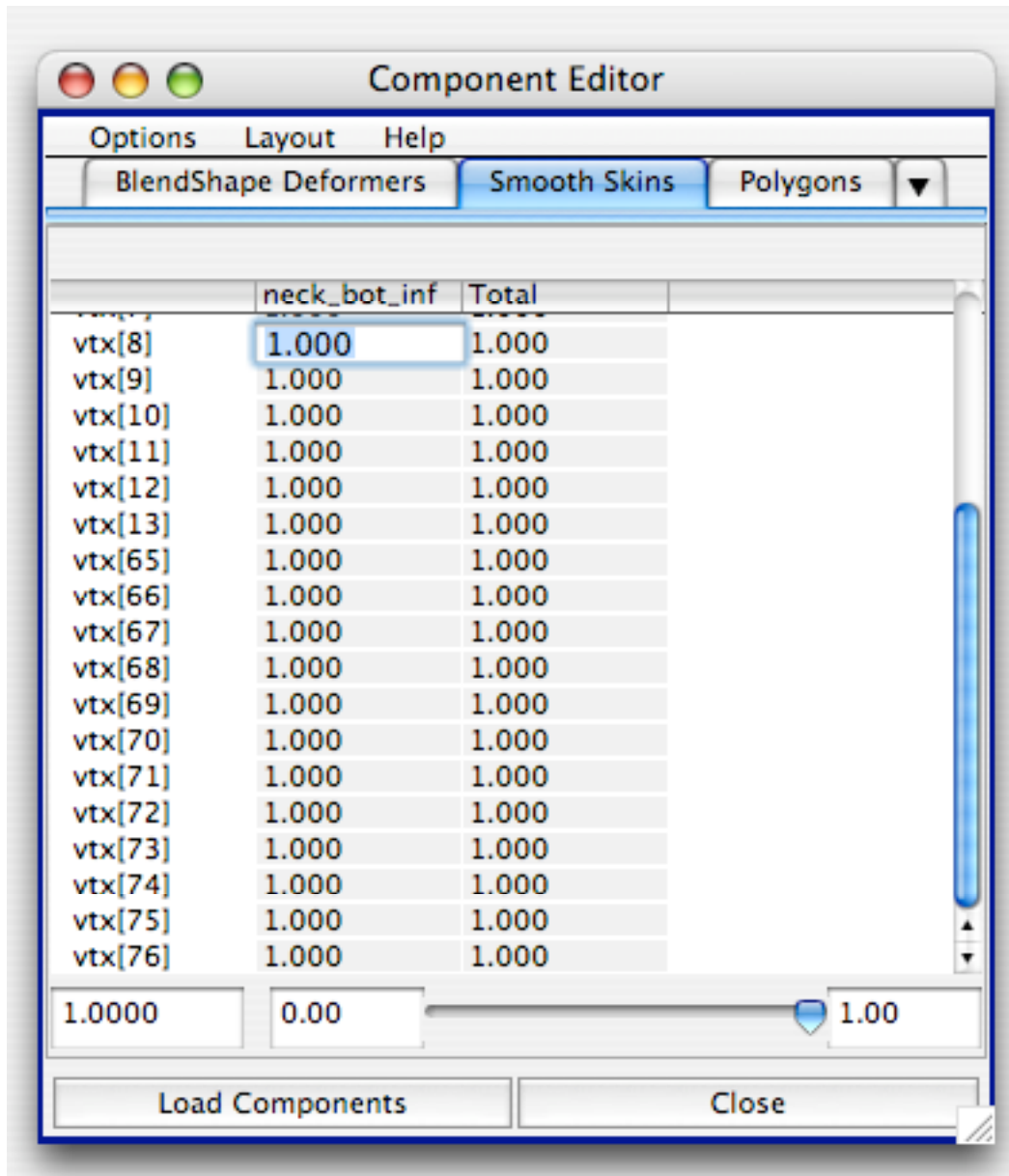
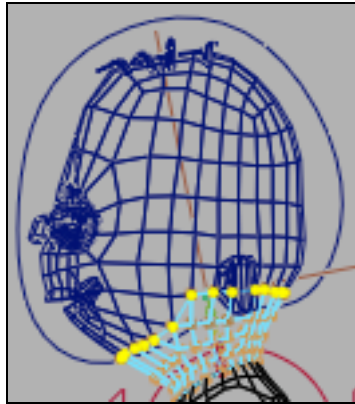


Figure 235 - after changing the weight values to 1.0

- Select the **top** row of verts for **neck\_geo**



*Figure 236 - top row of verts selected*

- In the component editor, find the **neck\_top\_inf** column and change all the values to 1.

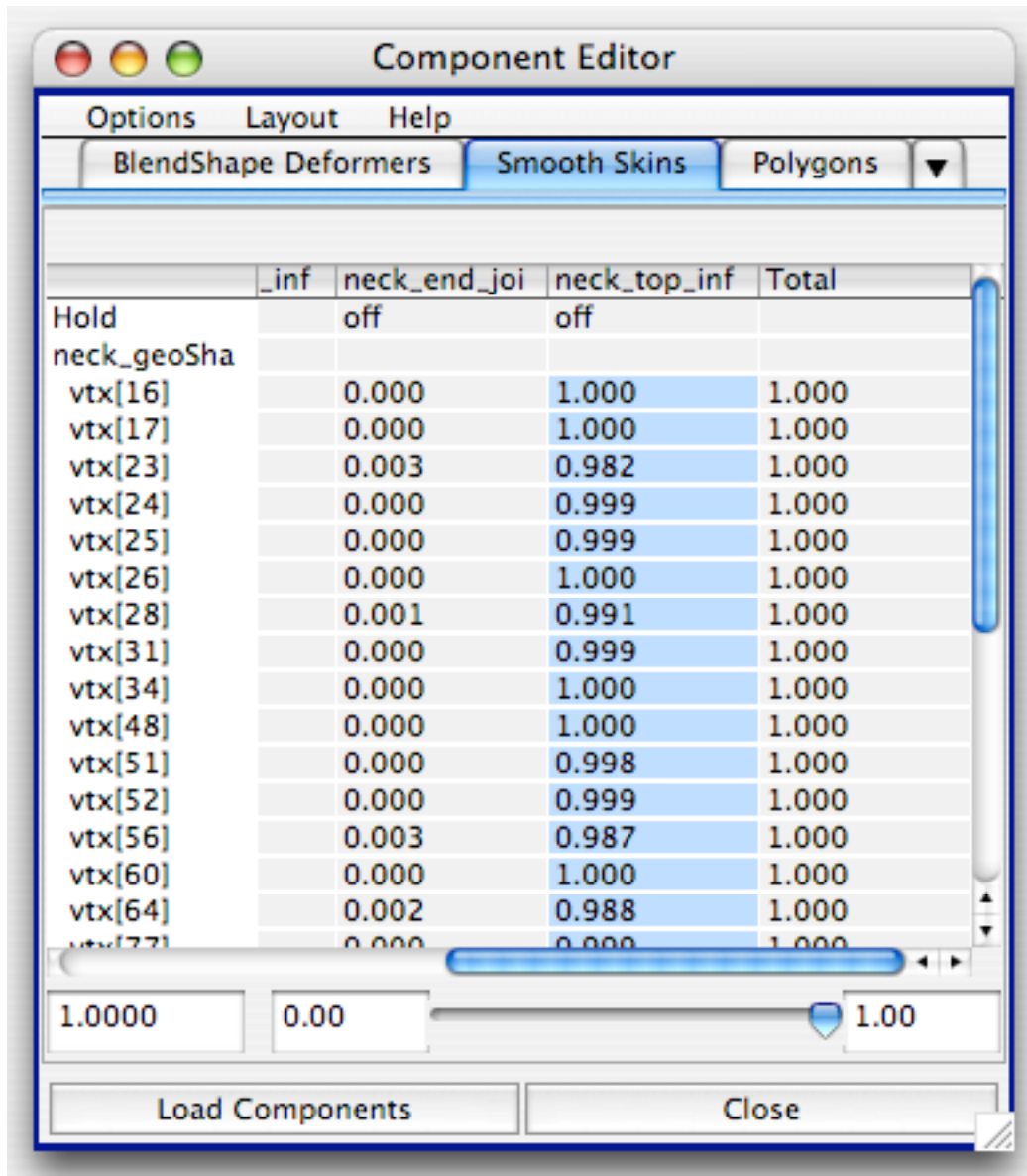


Figure 237 - Top row of verts and their skin weights

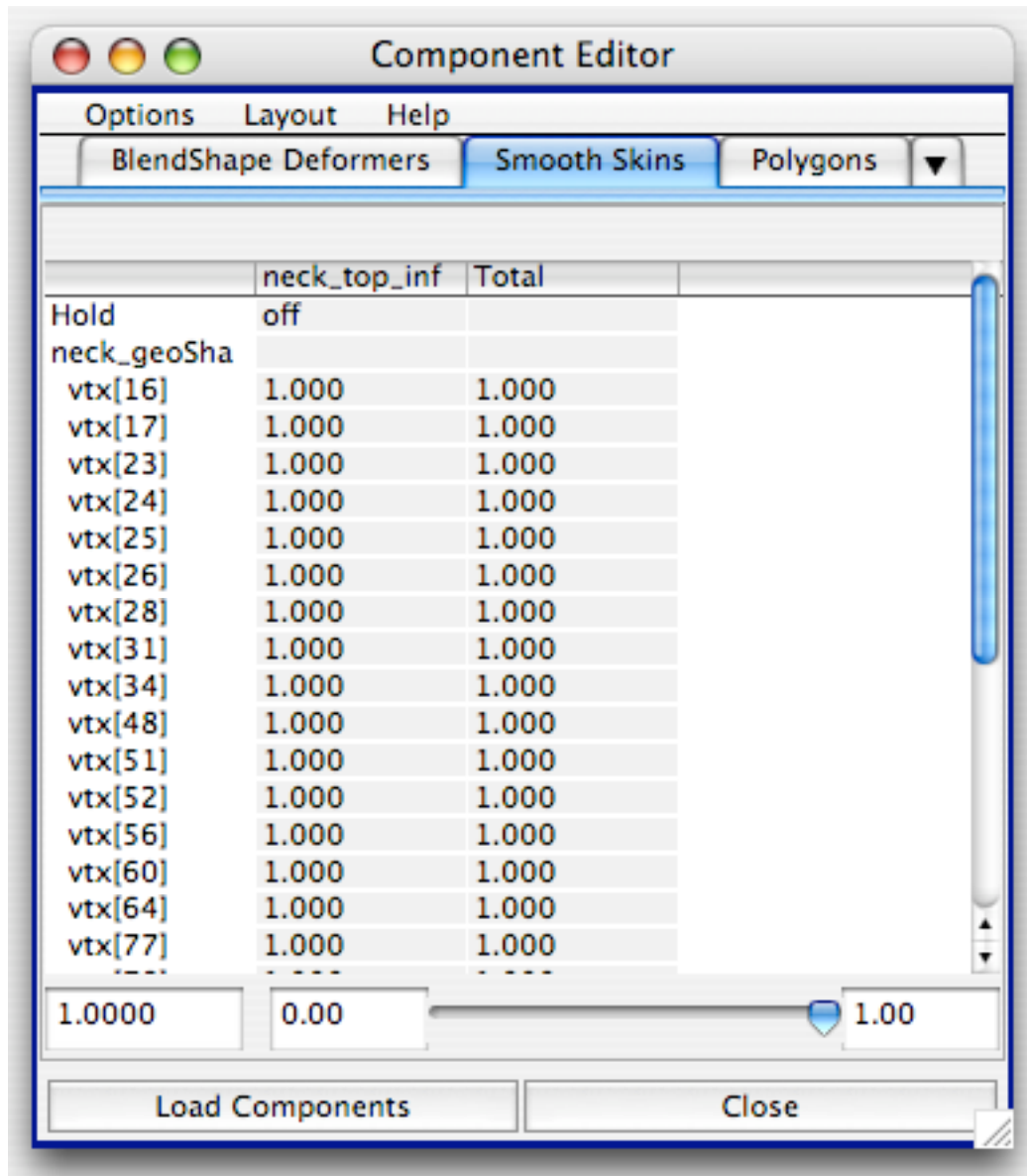


Figure 238 - Change the skin weights to 1

#### 55. Smooth the skin weights

- Select the rest of the verts, so only the *inside* verts are selected (not the top or bottom).

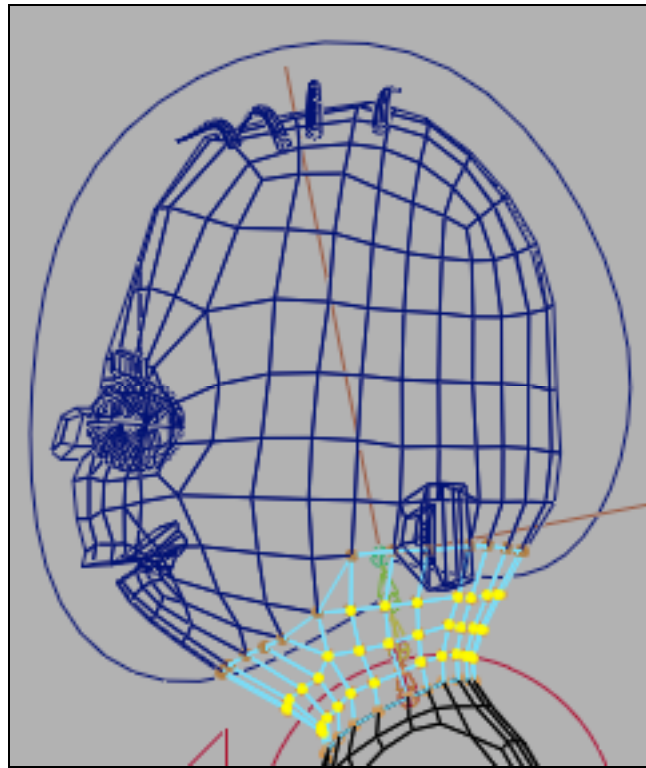



Figure 239 - inside verts selected

- Choose **Skin > Edit Smooth Skin > Paint Skin Weights Tool**
- Double-click on the Tool in the tool bar to bring up the Tool Options dialogue. 
- Set **Influence** to **neck\_top\_inf** and set **Paint Operation** to **Smooth**.
- Click **Flood** a few times to flood smooth the weights.

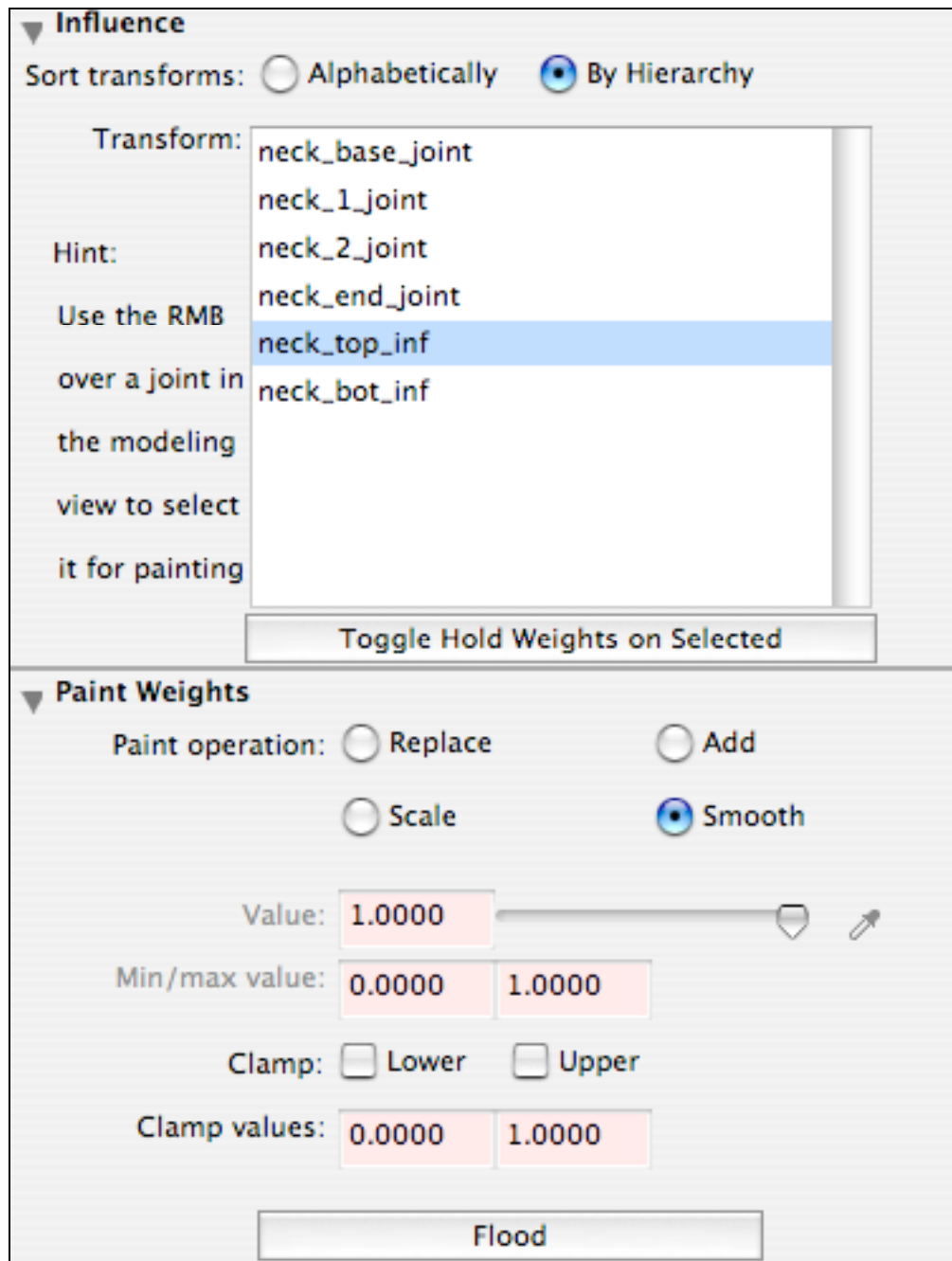


Figure 240 - Smooth Weights settings for the top influence object

- Now do the same for **neck\_bot\_inf**



- Select **neck\_bot\_inf** and click **Flood** a few times.

Now when you manipulate the head, the skinning for the neck should be a lot smoother.

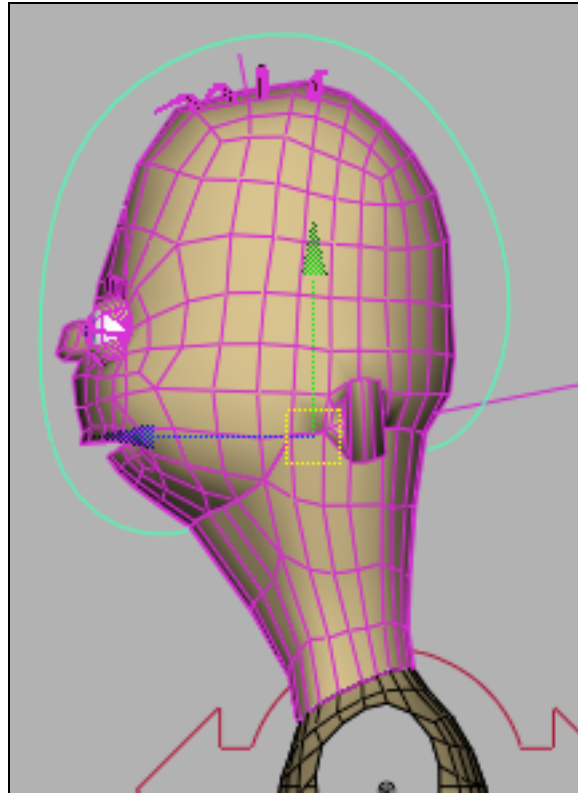


Figure 241 - Smooth Neck Skinning

*Example File: jj\_neck\_rig\_wip\_v5.ma*

#### Fix Neck Joint Flipping

If you watch the skeleton inside the neck when we bring the head closer and further away, you'll notice it flipping. This is what we were demonstrating earlier, where the fix is to measure the distance between the head and neck, and then scale the skinning joints appropriately.



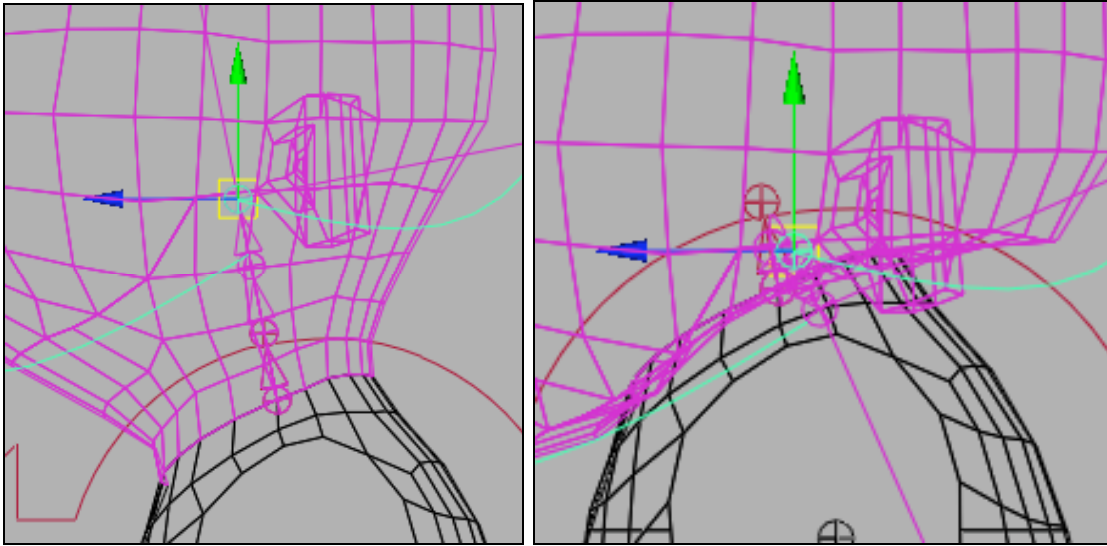


Figure 242 - neck joints flipping incorrectly

#### 1. Create a Distance Node

- Choose **Create > measure tools > Distance Tool**
- Create the distance tool from the top of the neck joint to the base.

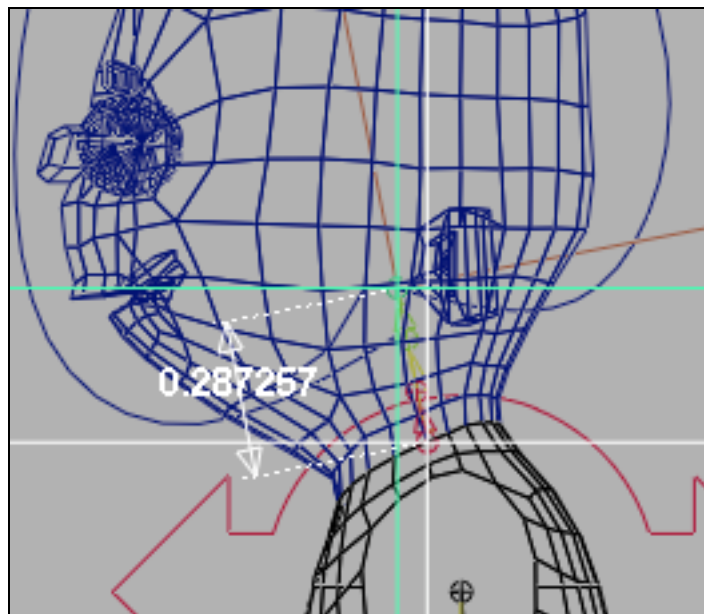


Figure 243- distance tool created from top of neck to base

Just as before, we're going to hook up the distance to the scale of the skinning joints. However, this time we're not just going to put a `setDrivenKeyframe` on the joints based on the distance.. we want to make sure that the manipulation will work no matter when our *global scale* attribute is scaled.

#### 56. Rename the distance nodes

- Rename **distanceDimension1** to **neck\_distance**
- Rename **locator1** to **neck\_length\_top**
- Rename **locator2** to **neck\_length\_bot**

#### 57. Create a multiplyDivide node for world scale

- Select **neck\_distance**
- Open up the Hypergraph by going **Window > Hypergraph Input and Output Connections**

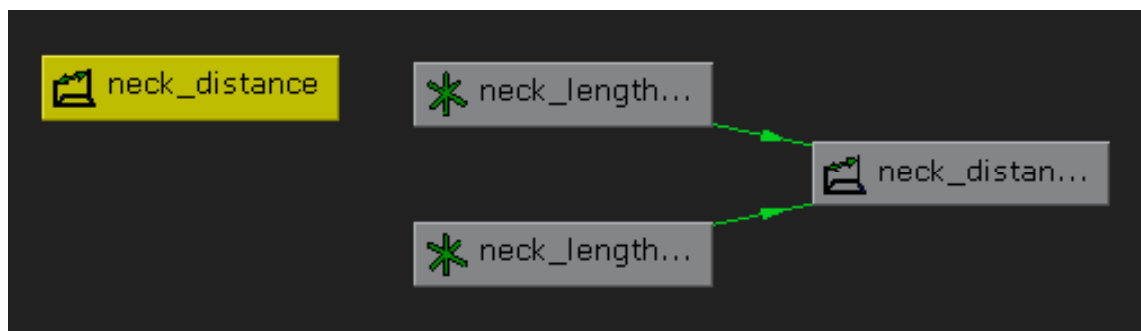
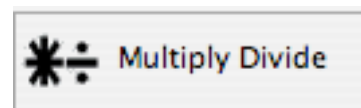


Figure 244 - Hypergraph Distance Dimension Node

- Choose **Rendering > Create Render Node**
- Switch to the **Utilities** tab.
- Click **Multiply Divide**



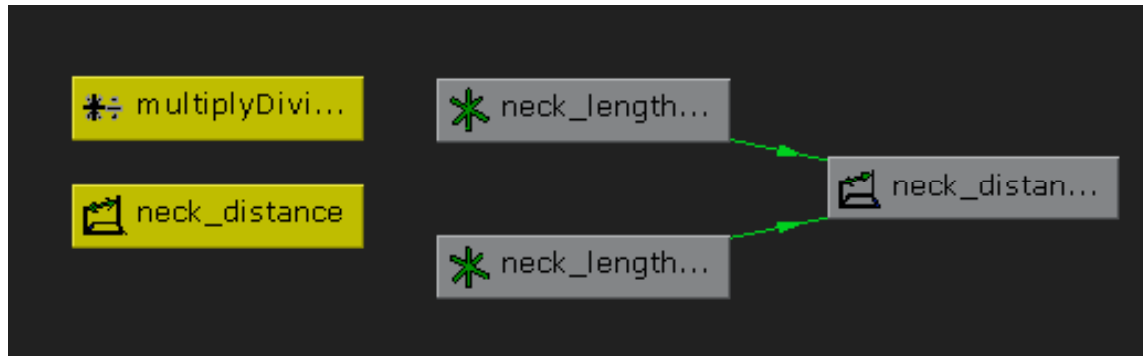


Figure 245 - Adding MultiplyDivide1

#### 58. Rename the node

- Rename **multiplyDivide1** to **neck\_normalized\_scale**



Figure 246 - Renaming all the nodes

#### 59. Add all\_anim to the hypergraph

- Click on **all\_anim** in the **Outliner** with the middle mouse button and **drag** and **release** on the **Hypergraph**.

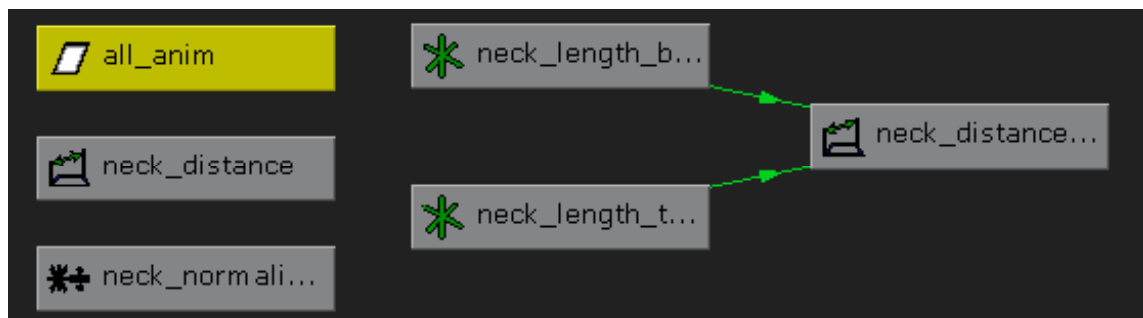


Figure 247- Dragging all\_anim into the Hypergraph

#### 60. Connect neck\_distanceShape and all\_anim to neck\_normalized\_scale

- Use the Middle Mouse Button to drag **neck\_distanceShape** onto **neck\_normalized\_scale**.
- When the menu pops up, click **Other**. This will open the **Connection Editor**.
- Connect the **neck\_distanceShape.distance** to **neck\_normalized\_scale.input1X**

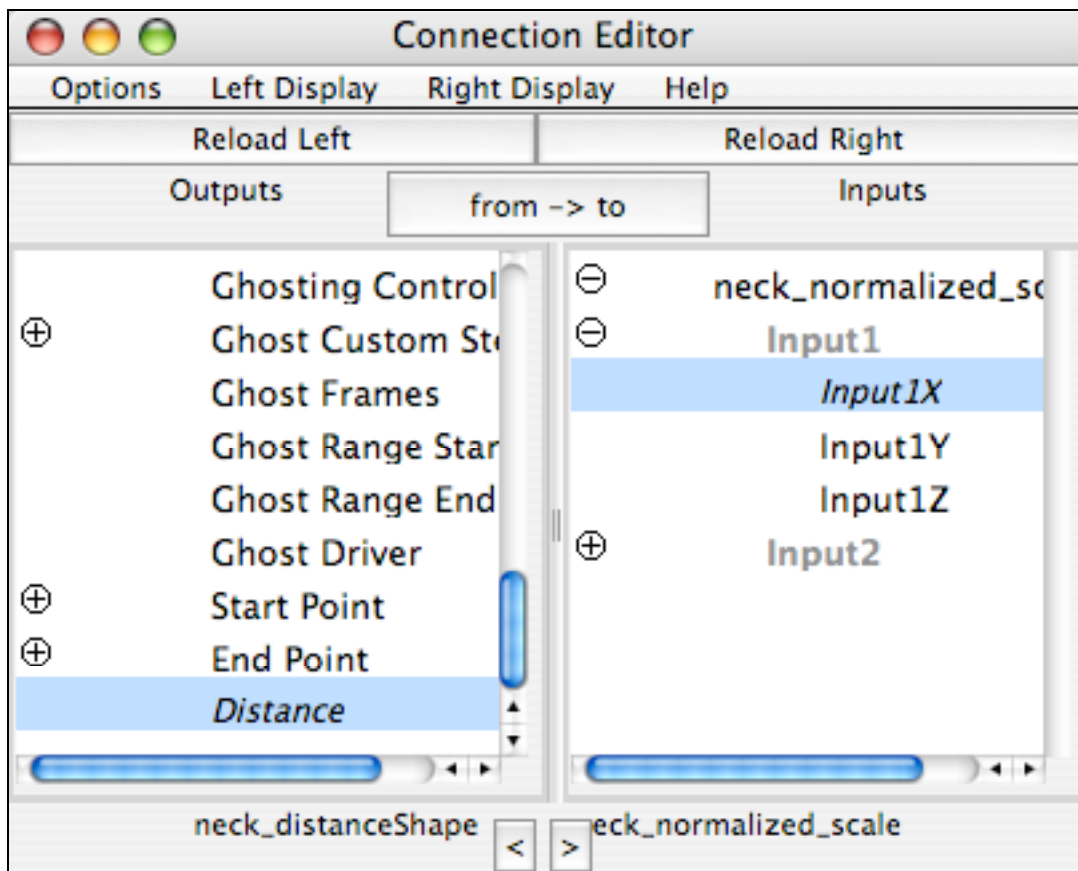


Figure 248 - Connecting **neck\_distanceShape.distance** to **neck\_normalized\_scale.input1X**

- Use the Middle Mouse Button to drag **all\_anim** onto **neck\_normalized\_scale**.
- When the menu pops up, click **Other**. This will open the **Connection Editor**.
- Connect **globalScale** (remember, it used to be **scaleY**) to **input2X**.

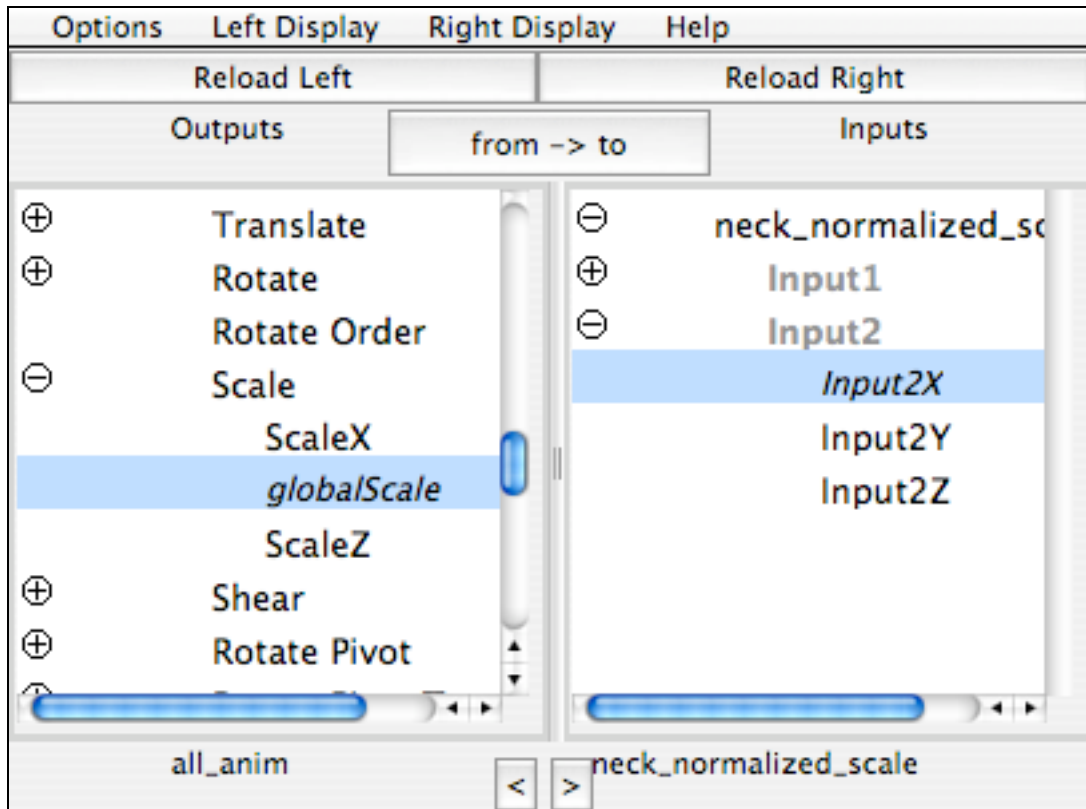


Figure 249 - connecting all\_anim.globalScale to neck\_normalized\_scale.input2X

#### 61. Set the multiplyDivide node to divide

- Select **neck\_normalized\_scale** in the Hypergraph.
- Hit **Ctrl+a** to open the Attribute Editor
- Change **Operation** to **Divide**

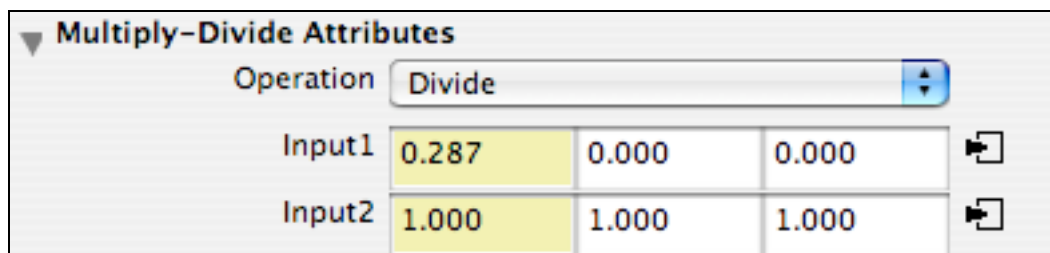


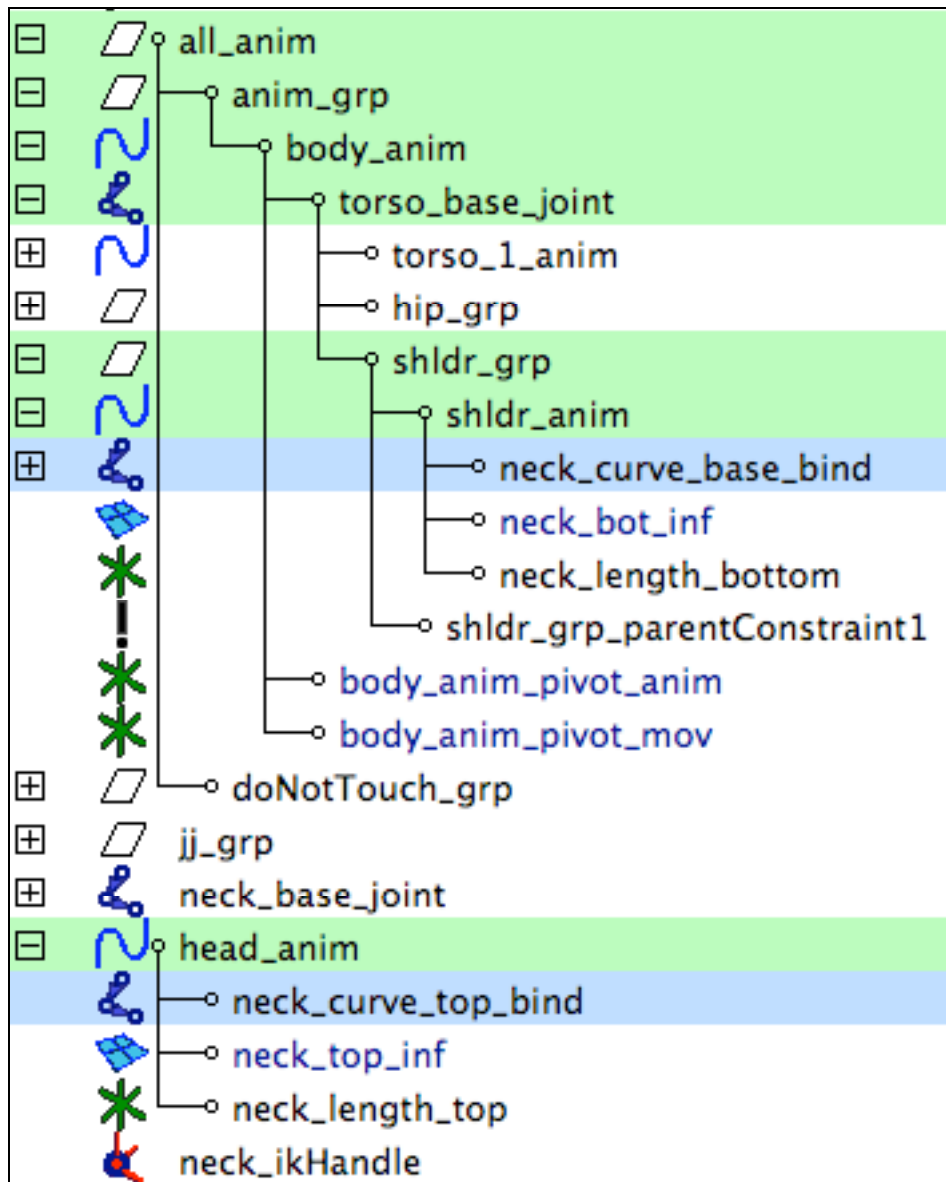
Figure 250 - Changing the operation to Divide.

#### 62. Parent the locators to the correct controls

- Parent **neck\_length\_top** to **head\_anim**
- Parent **neck\_length\_bottom** to **shldr\_anim**

### 63. Create a setDrivenKeyframe on the scale of the joints

- Select **neck\_curve\_top\_bind** and **neck\_curve\_base\_bind**



*Figure 251 - Selecting the proper joints*



## Master Classes

# Instructor Notes – Animator Friendly Rigging Part II

Jason Schleifer

- Select the scale attributes in the channel box.
- Choose RMB > **Set Driven Key**
- Select **neck\_normalized\_scale** in the Hypergraph.
- Click **Load Driver** in the Set Driven Key menu.
- In the top pane select **outputX** as the driver
- In the bottom pane, select both joints and the **scale** attributes as the driven.

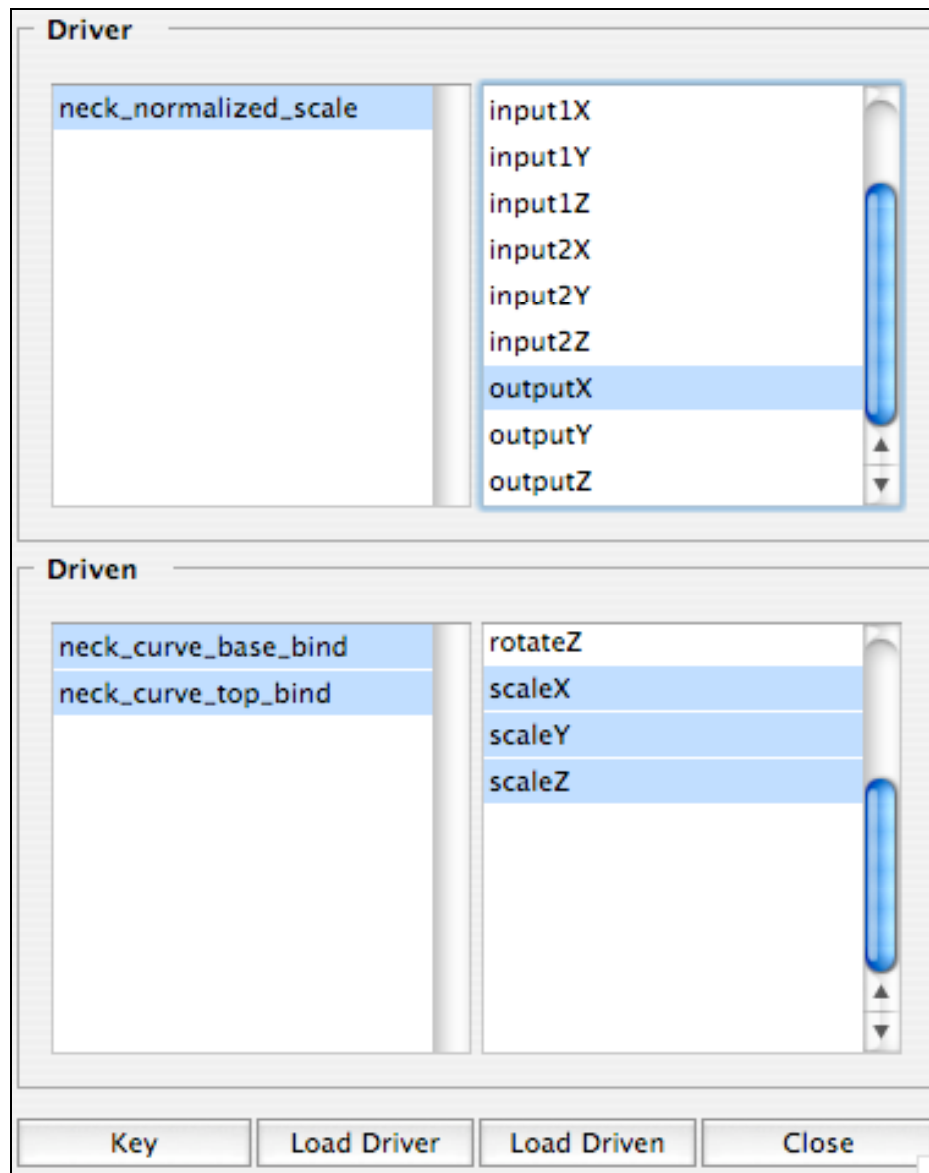
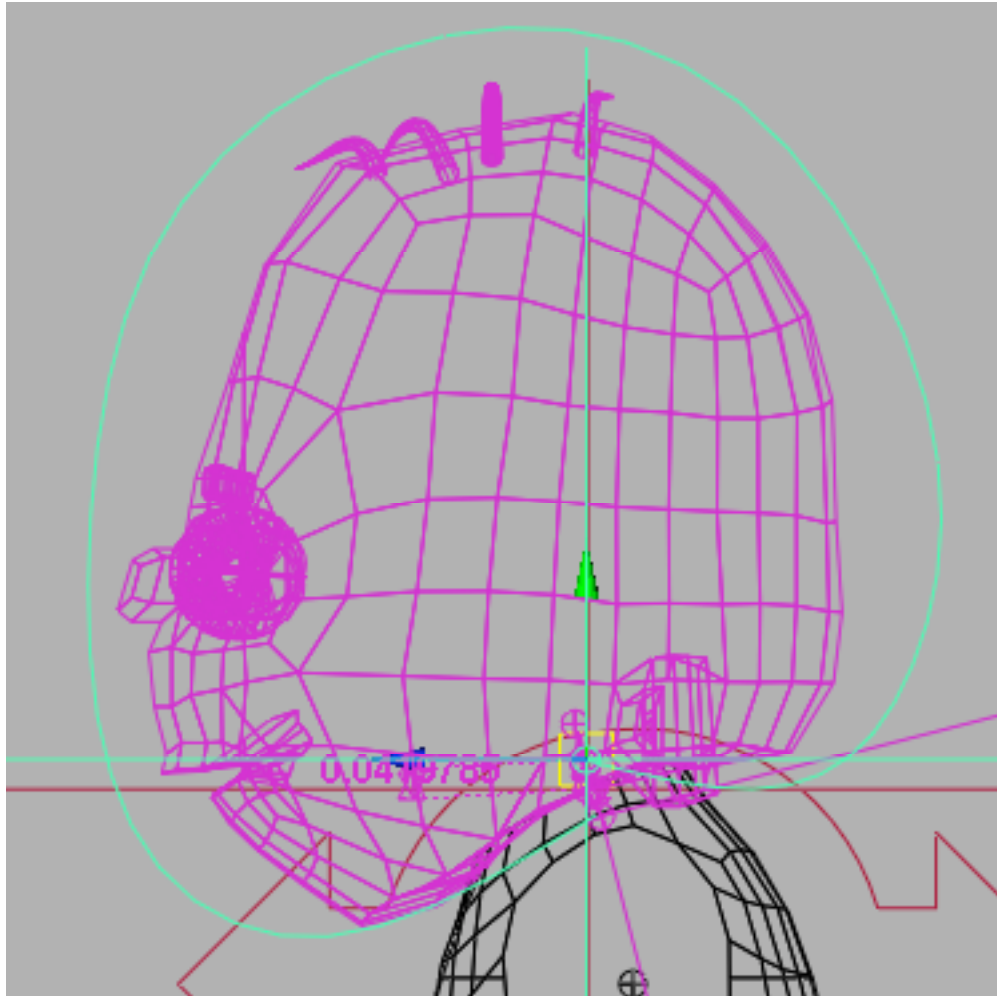


Figure 252 - Set Driven Key settings

- Click **Key** to set the default setting.
- Move the head closer to the neck until you see the joints start to flip.





*Figure 253 - neck joints starting to flip*

- Now select **neck\_curve\_base\_bind** and **neck\_curve\_top\_bind**
- Set their scale values down to **.001**
- Click **Key**.
- Now when you manipulate the head, the joints shouldn't flip.

*Note: You may find that you actually prefer to not have the extra joint under **neck\_curve\_base\_bind**. That's totally fine! Simply don't create it in the beginning and proceed as normal.*

*Example File: jj\_neck\_rig\_wip\_v6.ma*

#### Neck Control?

For this character, a more sophisticated neck control isn't really necessary, as we can get all the motion we want out of just the head itself. However, it may be necessary for your character to *add* some extra controls for the neck. If this is the case, you can use the same techniques used for the back control to add *fk* controls for the neck. It works just as well and gives you just as much control.

However, as mentioned in the previous paragraph.. that level of control isn't necessary for this character. Our neck control is going to be much simpler..

#### 1. Create a neck control

- Choose **Joint > Create Joint Tool**
- Holding down the **v** key for point snap, click at the base of the neck\_base\_joint and then again at the top of the neck.

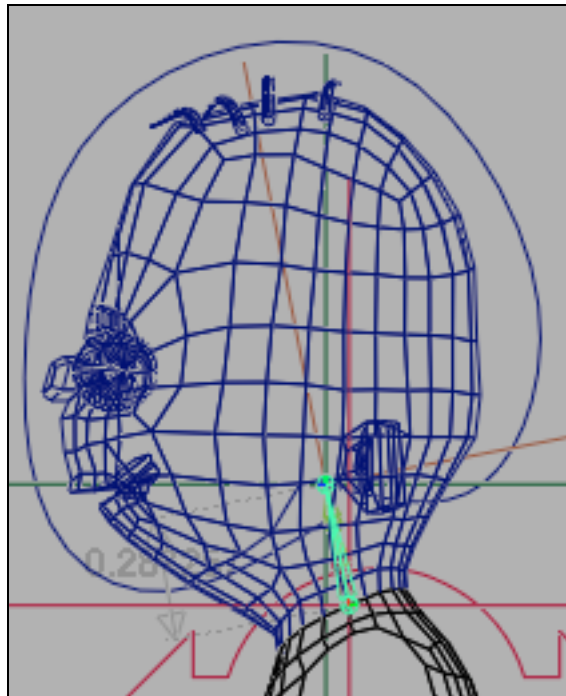


Figure 254 - building a neck joint control

#### 64. Rename the joints

- Rename **joint1** to **neck\_anim**

- Rename **joint2** to **neck\_anim\_end\_joint**

#### 65. Parent **neck\_anim** to **shldr\_anim**

- Parent **neck\_aim** to **shldr\_anim**

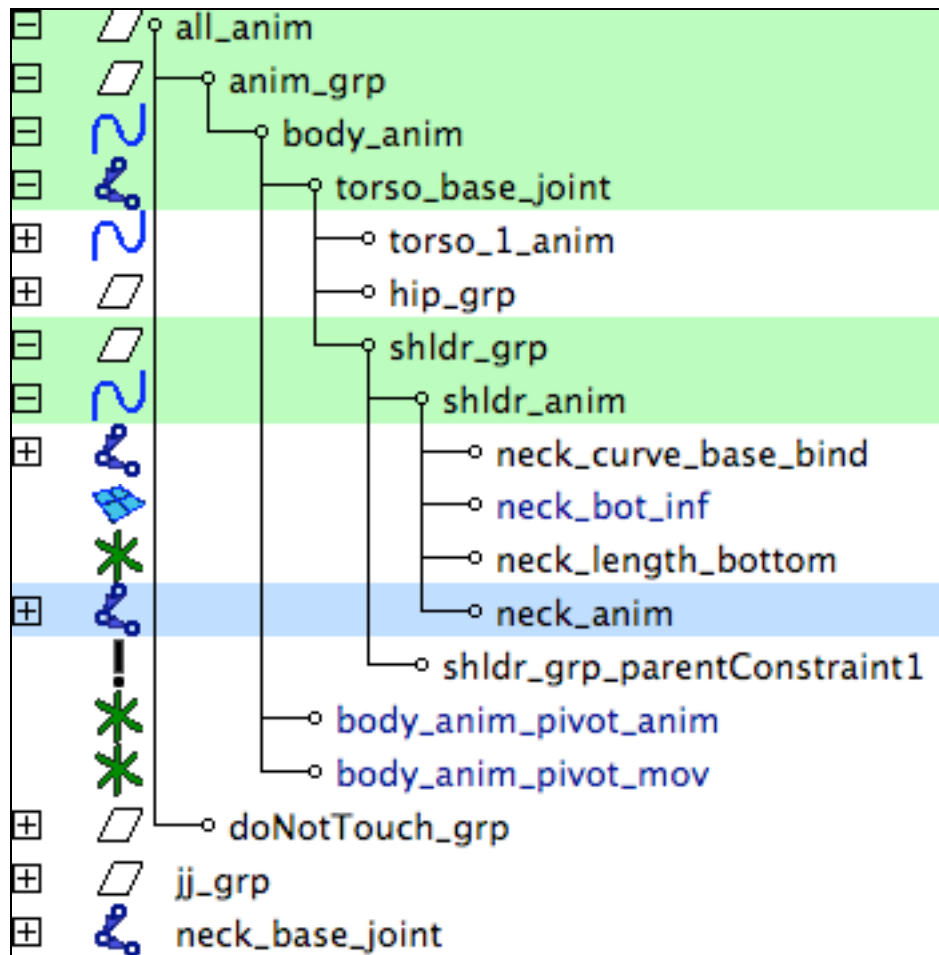
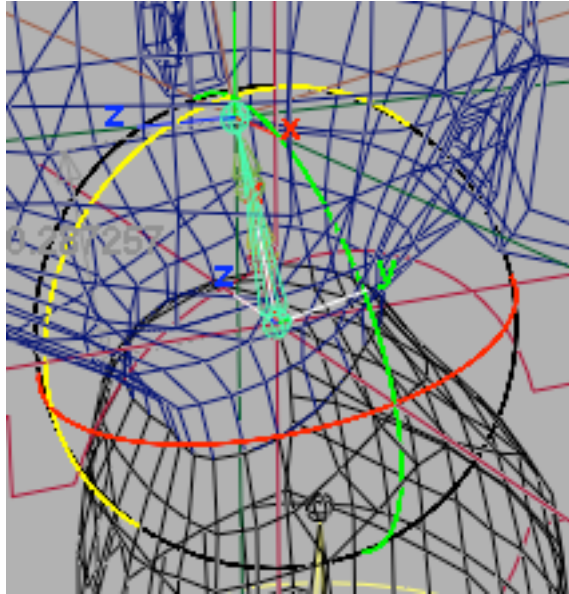


Figure 255 - Neck\_anim parented to shldr\_anim

#### 66. Set the Joint Orient for the Neck

One of the important things about setting up FK controls is to make sure that their rotations make sense. For the fk neck, we want it's rotations to match the head and body, so when the animator uses **rotateX** in the neck, it bends forward just as it does in the head.

Currently, the joint orient is incorrect. Notice how the **rotateX** actually twists along the joint, instead of rotating it forward.



*Figure 256 - neck\_anim rotation orders don't match the head or shoulders.*

- Select **neck\_anim**
- Choose **Skeleton > Joint Orient > Option Box**
- Set **Orientation** to **YXZ**
- Set **Second Axis World** to **+x**
- Click **Orient**

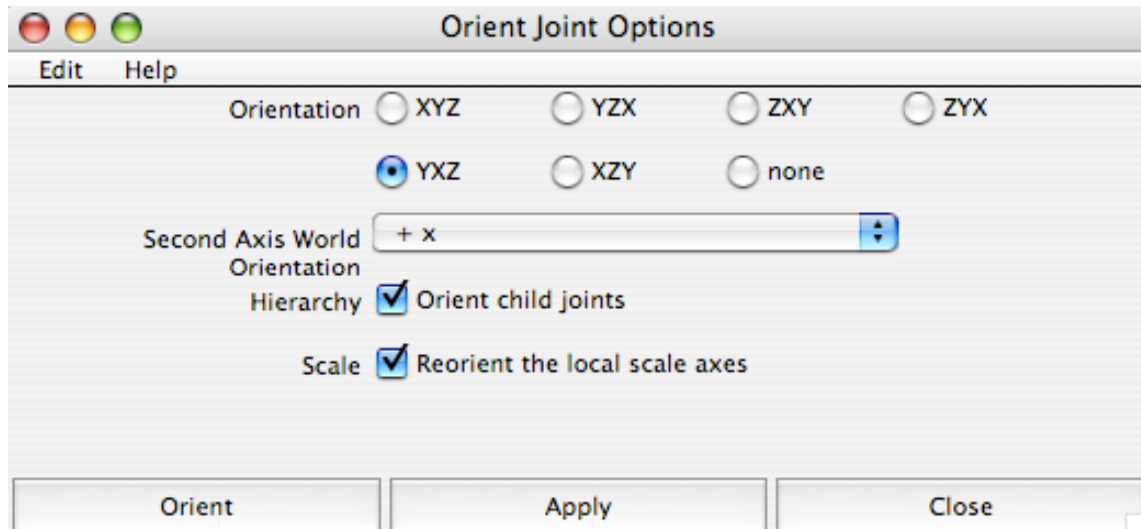


Figure 257 - Orient Joint Options

#### 67. Set the neck\_anim rotation order

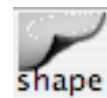
- Now we have to match the rotation order for the head. **Head\_anim** has a rotation order of **zxy**. Since an fk joint should have the *last* rotation be the one that's aiming down the joint, and y is the axis that aims down the joint, we just want to switch it slightly.
- Set the rotation order to **yzx**.

#### 68. Create a curve control for the neck

- Create a nurbs Circle using **Create > NURBS > Circle**
- Parent the nurbs circle shape to neck\_anim by typing:  
`parent -add -shape nurbsCircleShape1 neck_anim;`

or

Select nurbsCircle1 and neck\_anim and click the shape button on the Animator Friendly Rigging Shelf



- Select the **Cvs** of the circle by hitting the **F8** hotkey.
- Select the CVS.

- Scale them down closer to the neck.

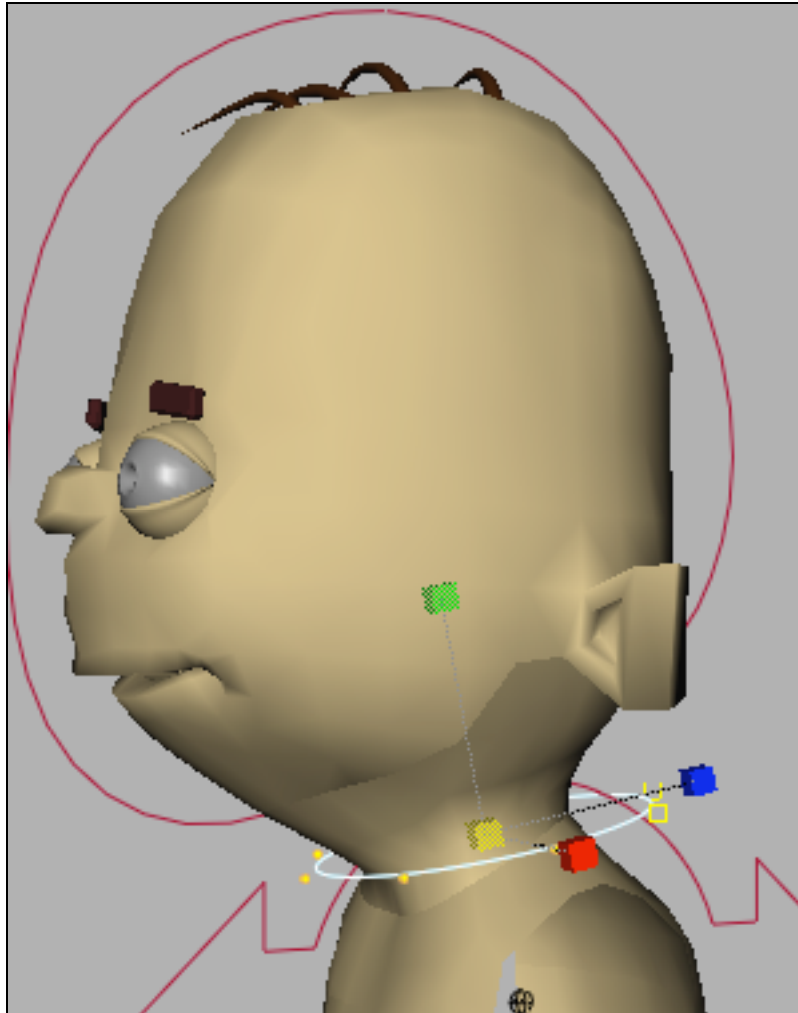


Figure 258- nurbs curve for the neck

- Delete **nurbsCircle1**

*Example File: jj\_neck\_rig\_wip\_v7.ma*

#### Create Constraints For Head Space

Our head control is just about ready to use. The next step is to create the various spaces that we want the head to operate in. This means we'll create the

possibility for the head to follow the neck, the shoulders, the body, or the all control, for either translation *and/or* rotation.

The animator will be able to decide if they want the head's orientation to follow the shoulders, but the translation to follow the neck, or the orientation to stay in all space, or everything in all space. It puts the power in *their* hands to make this decision, depending on the needs for their shot.

#### 1. Create a Multi-Constraint for Translation

- Click on the **Setup Multi Constraint UI** button in the **Animator Friendly Rigging** shelf.
- Select **head\_anim** and click **Load Selected** for **Object To Constraint** and for **Control Object**.
- Select **neck\_anim\_end\_joint**, **shldr\_anim**, **body\_anim**, and **all\_anim** and click the **+** button to add them as **Target Objects**
- Turn **OFF** Orient, and turn **ON** Point
- Set **Control Attribute** to **t\_space**

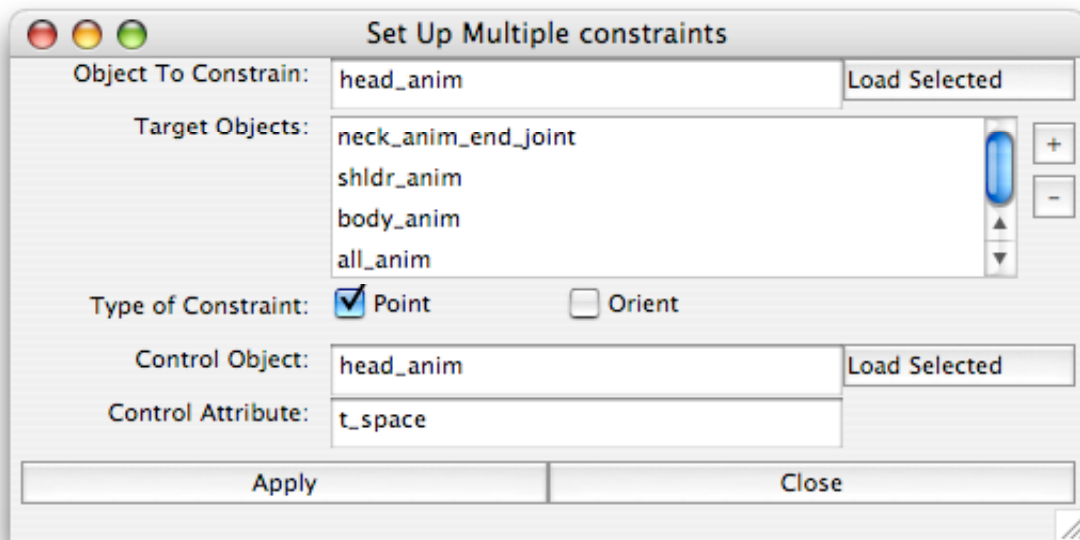


Figure 259 - Settings for **t\_space** for Multiple Constraints

- Click **Apply**



#### 69. Create a Multi-Constraint for Orientation

- Turn **OFF** Point and turn **ON** Orient.
- Change **Control Attribute** to **o\_space**.

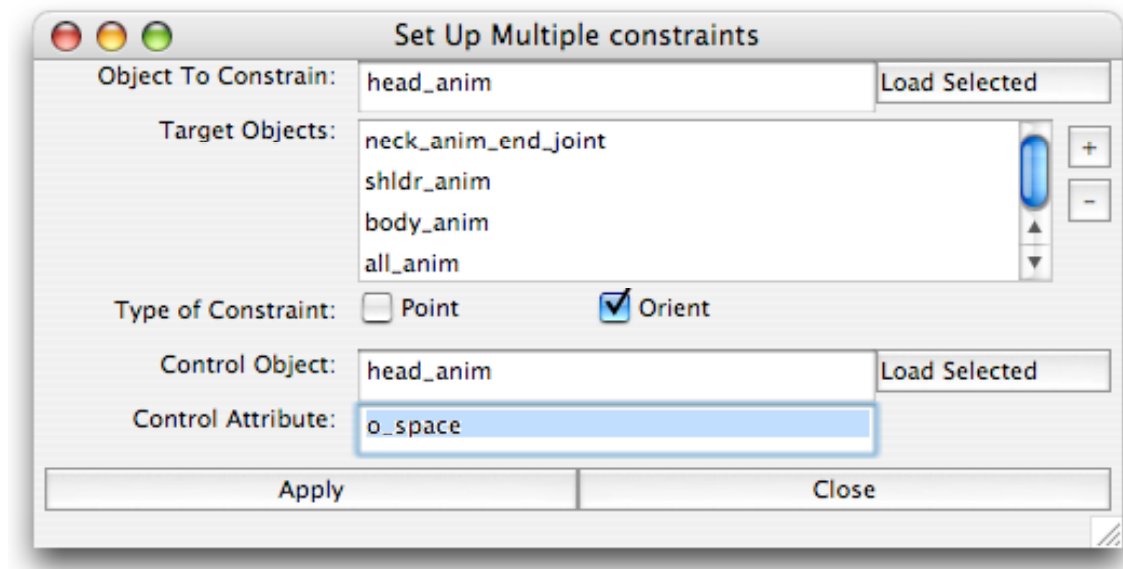


Figure 260 - Settings for o\_space for Multiple Constraints

- Click **Apply**

head_anim	
TranslateX	0
TranslateY	0
TranslateZ	0
RotateX	0
RotateY	0
RotateZ	0
ScaleX	1
ScaleY	1
ScaleZ	1
Visibility	on
T_space	neck_an ▼
O_space	neck_an ▼

Figure 261- Head Spaces

You'll notice that by default the names of the spaces are based on the names of the objects they're constrained to. While that's helpful, the animator probably





## Master Classes

# Instructor Notes – Animator Friendly Rigging Part II

Jason Schleifer

doesn't need to know that the neck space is actually neck\_anim\_end\_joint. Instead, let's change the names of the pull-downs to make sure that they are easier to read.

### 70. Modify the names in t\_space

- Select **T\_space** in the **Channel Box**
- Choose **RMB > Edit Attribute**



## Master Classes

### Instructor Notes – Animator Friendly Rigging Part II

Jason Schleifer

**Attributes**  

constGrp  
const\_o\_space  
const\_t\_space  
o\_space  
t\_space  
type\_o\_space  
type\_t\_space

New name: t\_space

**Numeric Attribute Properties**  
☒ Keyable  
☐ Has minimum ☐ Has maximum  
Min/Max: 0.0 0.0

**Enum Names**  

neck\_anim\_end\_joint  
shldr\_anim  
body\_anim  
all\_anim

New name:

Close

Figure 262 - Editing `t_space` attribute

- Select **t\_space**
- Down in **Enum Names** you can see the list of items that we have to work with.
- Select **neck\_anim\_endJoint**
- In the **new name** field, enter **neck** and hit **return**.

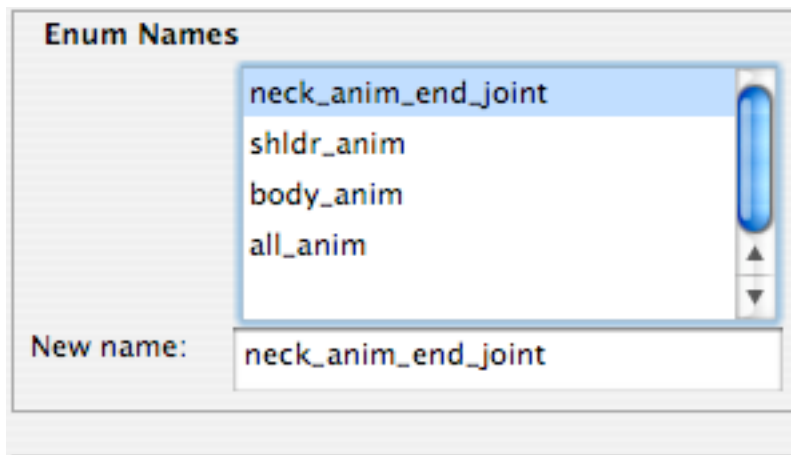


Figure 263 - selecting `neck_anim_end_joint`

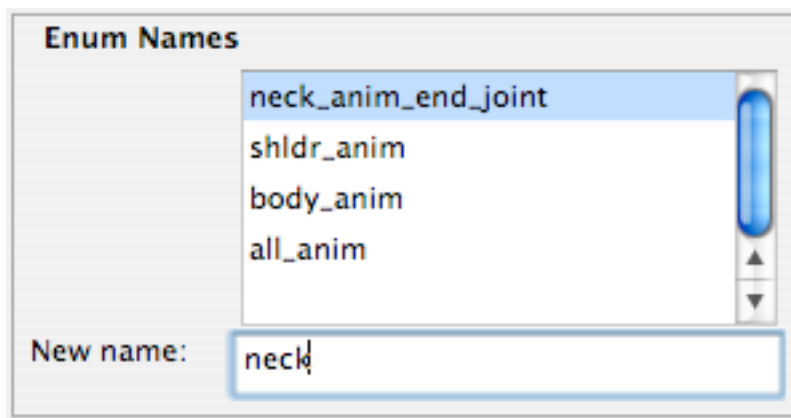


Figure 264 - Typing a new name

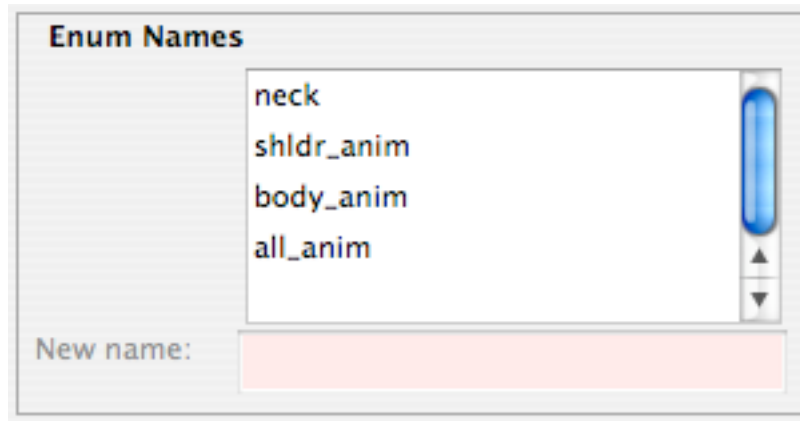


Figure 265 - Validating the new name

- Now change the names for the rest of the **Enum** names.

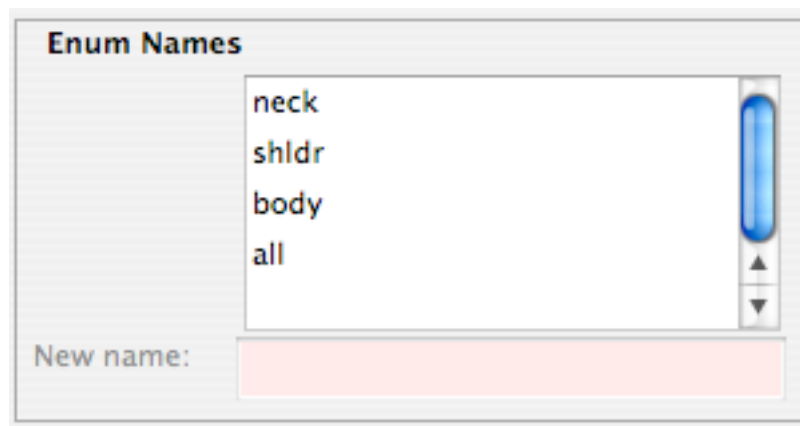


Figure 266 - The rest of *enum names* changed for **t\_space**

#### 71. Change the Enum Names list for **o\_space**

- Click on **o\_space**
- Change the names to **neck**, **shldr**, **body**, and **all**

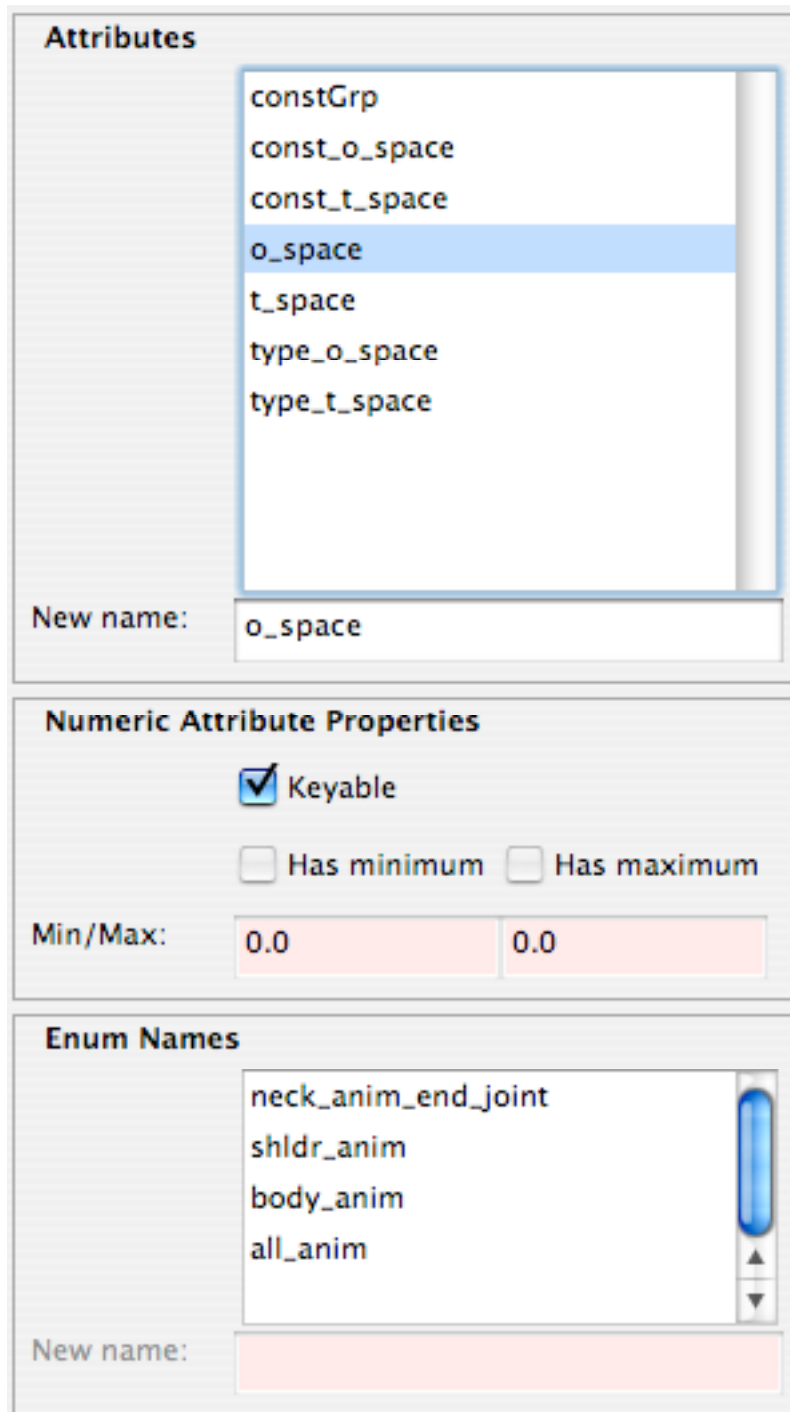


Figure 267- o\_space default enum names



## Master Classes

### Instructor Notes – Animator Friendly Rigging Part II

Jason Schleifer

**Attributes**

- constGrp
- const\_o\_space
- const\_t\_space
- o\_space**
- t\_space
- type\_o\_space
- type\_t\_space

New name: o\_space

**Numeric Attribute Properties**

☒ Keyable

☐ Has minimum ☐ Has maximum

Min/Max: 0.0 0.0

**Enum Names**

- neck
- shldr
- body
- all

New name:

Close

Figure 268 - o\_space modified

#### 72. Set the Default Spaces

The default animation space for the animator will now be set so the head's **translation** follows the neck, but the **rotation** follows the **all** control. This is one of the easiest ways to work, as the head will be very easy to control.

- Set **head\_anim.o\_space** to **all**
- Set **head\_anim.t\_space** to **neck**

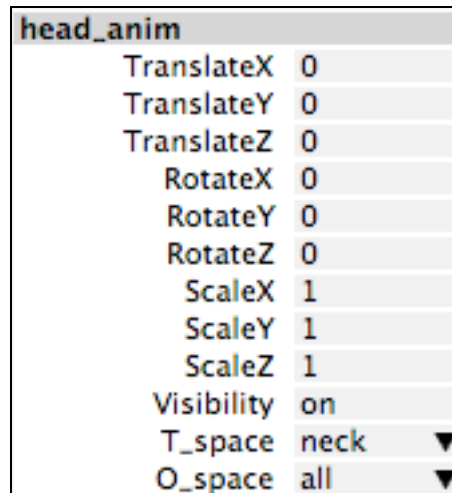


Figure 269 - head\_anim default spaces set

Example File: *jj\_neck\_rig\_wip\_v8.ma*

#### Check with our Animation Requirements.

##### General head motion requirements

- Head needs to be able to orbit side to side
- Head needs to be able to look up and down
- Head needs to be able to lean side to side
- Head should be able to move forward and back
- Head can compress and extend
- Head should be able to move side to side.

#### Additional head motion considerations

- Head should be able to orient *independent* of shoulders and body (i.e. twist shoulders, head should stay at same orientation), or *with* the shoulders and body (twist shoulders & it orients the head).
- Head translation should be able to be done in shoulder space, body space, or all space.

It looks like we have solved the motion requirements of our head! Fantastic!  
What about our animation control requirements?

- **Simple controls**  
Yes, we have simple controls for the head and neck.. as simple as you can get!
- **Animation should be easily transferable.**  
We have the head control set up in a way that will make it easy to copy animation.
- **Controls should be unique and make immediate sense.**  
Sweet, got that sorted.
- **Controls should have the correct rotation orders.**  
We checked our rotation orders on the head, it is the same as for the body, so that should be good. The neck control has the correct joint orient and rotation orders.
- **Controls should be named correctly.**  
We named our head and neck control: head\_anim and neck\_anim. Easy to find, easy to use!
- **Only be able to set keyframes on controls we want animators using.**  
Ah, this we have not done yet!

#### 73. Set the keyable attributes on head\_anim

- Select **head\_anim**
- In the Channel Box, select the **scale** attributes and the **visibility**.
- Choose **RMB > Lock and Hide Selected**



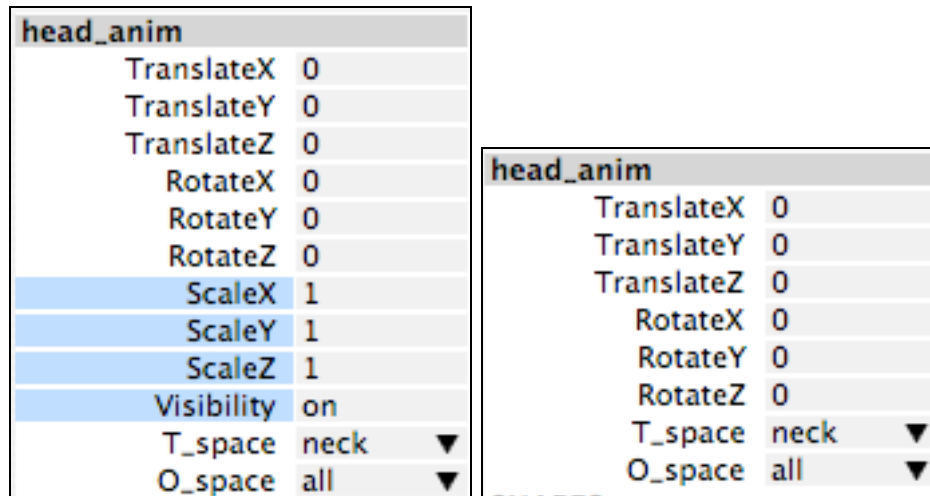


Figure 270 - Locking and hiding the scale and visibility attributes on head\_anim

#### 74. Set the Keyable Attributes on neck\_anim

- Select **neck\_anim**
- In the Channel Box, select the **translate**, **scale**, and **visibility** attributes.
- Choose **RMB > Lock and Hide Selected**

#### Clean the Scene!

As you can probably tell, the outliner and display are a bit of a mess right now. We have a lot of control, but there are nodes all over the place, things visible that we don't need the animator to see, and attributes haven't been limited yet.

#### 75. Parent head\_anim\_grp

- Parent **head\_anim\_grp** under **anim\_grp**

#### 76. Parent DoNotTouch Nodes

- Select **neck\_base\_joint**, **neck\_ikHandle**, **neck\_curve**, **neck\_top\_infBase**, **neck-bot\_infBase**, and **neck\_distance**
- Hit **ctrl+g** to group them together.
- Rename the group **neck\_grp**
- Parent **neck\_grp** under **doNotTouch\_grp**



## Master Classes

# Instructor Notes – Animator Friendly Rigging Part II

Jason Schleifer

- Parent **head\_geo\_grp** under **doNotTouch\_grp**
- Rename the **head\_geo** node above **neck\_geo** as **neck\_geo\_grp**
- Parent **neck\_geo\_grp** under **doNotTouch\_grp**

### 77. Set Non-Inherit Transforms

- Select **neck\_curve**
- Open the Attribute Editor
- Turn off Inherits Transform
- Select **neck\_geo\_grp**
- Open the Attribute Editor
- Turn off Inherits Transform

### 78. Hide Items

- Hide **neck\_base\_joint**
- Hide **neck\_ikHandle**
- Hide **neck\_curve**
- Hide **neck\_distance**
- Hide **neck\_length\_top**
- Hide **neck\_length\_bottom**
- Hide **neck\_curve\_base\_bind**
- Hide **neck\_curve\_top\_bind**
- Hide **neck\_anim\_end\_joint**

### 79. Set Display Layers

- Add **neck\_geo** to **doNotTouch\_layer**
- Add **head\_geo\_grp** to **doNotTouch\_layer**
- Create a new layer called **fk\_neck\_layer**
- Add **neck\_anim** to **fk\_neck\_layer**
- Color **fk\_neck\_layer** yellow



## Master Classes

# Instructor Notes – Animator Friendly Rigging Part II

Jason Schleifer

- Add **head\_anim** to **mid\_anim\_layer**

At this point the head is just about ready for testing! The animator should be able to key the head's position and orientation, use the FK neck if they so wish, and control the head's space!

*Example File: jj\_neck\_rig\_wip\_v9.ma*

However, there's one last control that we may want to add.. and that's a head pivot control.

I've dealt with a number of situations where I had to have a character rest their head against something, and it would have been really great to be able to have an easy way to pivot around that object.

Let's give the animator the tools to move that head pivot!

### Add Head Pivot Control

#### 80.Add the head pivot

- Select **head\_anim**
- Click on the **Pivot: Create** button in the **Animator Friendly Rigging** shelf.



You'll notice that there's now a pivot available for the head! But what happens if we move the pivot?

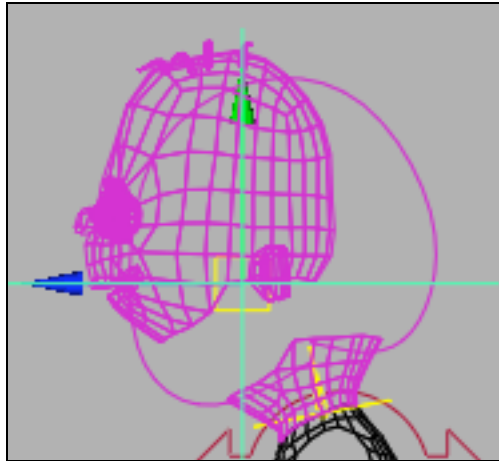
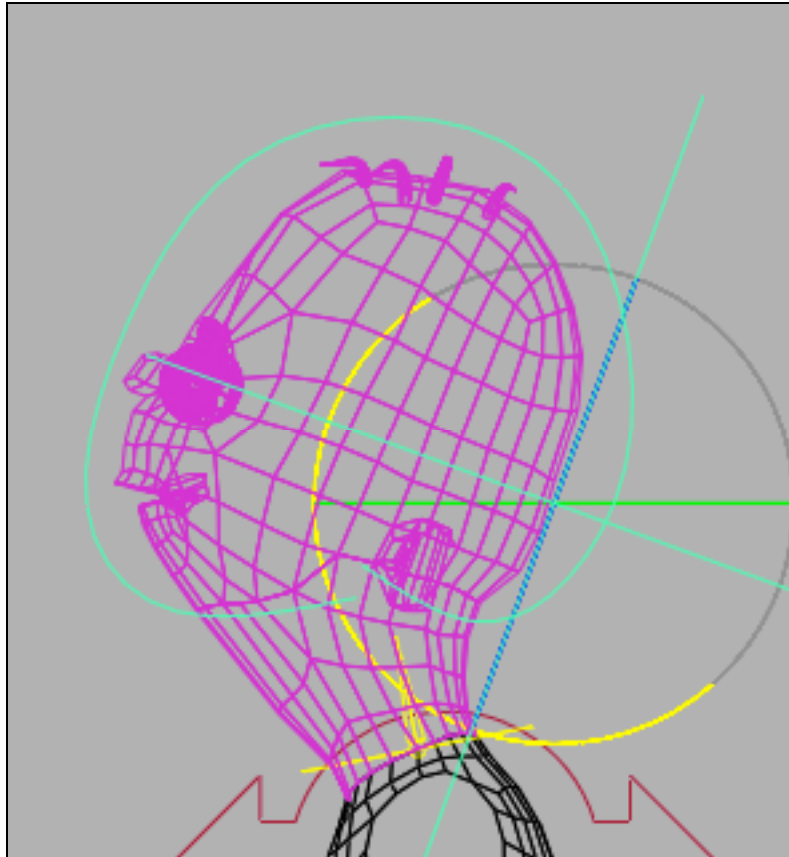


Figure 271 - Pivot moving

The reason for the head separating is because we currently have **head\_geo\_grp** parent constrained to **head\_anim**. By changing **head\_anim**'s pivot, it changes the position of **head\_geo\_grp**. So we just need to create a different group to parent constrain **head\_geo\_grp** to, and parent it underneath **head\_anim**.

#### 81. Create a new **head\_geo\_grp** constraint

- Select **head\_geo\_grp**
- **Duplicate** **head\_geo\_grp** by hitting **Ctrl+d**
- **Delete** all the children of **head\_geo\_grp1**
- **Rename** **head\_geo\_grp1** to **head\_geo\_const\_grp**
- **Delete** **head\_geo\_grp\_parentConstraint1**
- Select **head\_geo\_const\_grp** and then **head\_geo\_grp**
- Choose **Constrain > Parent**
- Parent **head\_geo\_const\_grp** under **head\_anim**



*Figure 272 - Ability to move head pivot!*

We can now give the animator the ability to move the head pivot.

Now that we have a rig that matches our needs, we'll hand it off to the animator or testing.

#### **82. Save the scene**

- Save your scene in the WIP directory for JJ as *jj\_rig\_v2.ma*



Master Classes

## Instructor Notes – Animator Friendly Rigging Part II

Jason Schleifer

### Author Biography

Jason Schleifer is a character animator at the Redwood Shores campus of Dreamworks Animation, currently animating on "Shrek 3". Most recently, he animated on "Over the Hedge", "Penguins Christmas Caper", and "Madagascar". Previously, Jason was Animation Lead at Weta Digital on "The Lord of the Rings, Return of the King." While at Weta Digital, he also animated on "The Lord of the Rings, The Two Towers" and "The Lord of the Rings, Fellowship of the Ring." As well as animating on the three films, he also wrote the animation pipeline that was used throughout the trilogy. In addition to animating for features films, Jason has taught two Maya Master Classes on Rigging for Animation, co-taught a 2002 Siggraph course on character rigging, and is currently a Mentor/Lecturer for the online animation school AnimationMentor.com. Read more of his rambling and check up on the progress of his short film "Jonh and His Dog" at <http://jonhandhisdog.com/>.



## Master Classes

# Instructor Notes – Animator Friendly Rigging Part II

Jason Schleifer

## Included Mel Scripts

### js\_attrDraggerSingle.mel

This script can be used to replace the use of the Channel Box for manipulating attributes on the selected object. With the use of a hotkey to toggle the script, the user can pick the keyable attribute they want to manipulate on the selected object. Then clicking and dragging with the middle mouse button will manipulate that attribute. If they select another object, the middle mouse button will automatically begin manipulating the attribute on the new object.

To set up the script for use, it's necessary to create a hotkey to toggle script on and off. To do that, you will need to set up a **press** and **release** command.

This is an example of setting the script to work with the “c” hotkey:

```
// Create the nameCommands. These will allow you to assign
// hotkeys to the script.
nameCommand
    -ann "Attribute Dragger Context Press"
    -c "source
        js_attrDraggerSingle;js_attrDraggerSingle;"
        js_attrDraggerPress;
nameCommand
    -ann "Attribute Dragger Context Release"
    -c "MarkingMenuPopDown"
        js_attrDraggerRelease;

// Assign the hotkeys
hotkey -k "c" -name "js_attrDraggerPress";
hotkey -k "c" -releaseName "js_attrDraggerRelease";
```

### js\_hashRename.mel

js\_hashRename will rename the selected objects and will replace any # marks with numbers.

For example, if given the command:

Autodesk® Maya® Master Classes – Instructor Notes  
SIGGRAPH™ 2006



## Master Classes

# Instructor Notes – Animator Friendly Rigging Part II

Jason Schleifer

```
js_hashRename "torso_####_control";
```

It will rename selected objects as:

```
torso_0001_control  
torso_0002_control  
etc.
```

### **js\_hashRenameUI.mel**

A user interface to js\_hashRename.mel.

### **js\_replaceHash.mel**

A script to replace a series of #### in a given string with the given number.  
For example:

```
js_replaceHash "my_###_string" 10;
```

Results in:

```
my_010_string
```

---

<sup>i</sup> <http://video.google.com>

<sup>ii</sup> <http://youtube.com>

<sup>iii</sup> <http://www.bbcmotiongallery.com/Customer/index.aspx>

<sup>iv</sup> Parkour Video Players – Bryce McGovern, Kevan Shorey, Jason Osipa, Carlos Puertolas, Jason Schleifer. Filmed by Rachel Ito. Shot on location in San Francisco. Music by Jason Schleifer.

<sup>v</sup> There are many fantastic skeleton references on this web page:

<http://staticfiles.sabc.co.za/VCMStaticProdStage/EDUCATION/Schools/Beyond%20The%20Classroom/Ideas%20Library/Theme%20Pictures/Human%20Body%20-%20Skeleton/Skeletal.htm>

<sup>vi</sup> **js\_splitSelJoint.mel** can be used to segment a skeleton segment into an even number of sections. You can also use the **js\_splitSelJointUI.mel** script to launch an interface to js\_splitSelJoint.





## Master Classes

# Instructor Notes – Animator Friendly Rigging Part II

Jason Schleifer

---

<sup>vii</sup> **js\_createSkelGeo.mel** will create geometry for the selected joints, or for the joints passed into the script.

<sup>viii</sup> **js\_createCurveControl.mel** - This script will create a single curve which can be used to control the given attribute on the selected objects. For example, if you want to drive the ty attribute on 10 objects with a "height" curve on another object, you would select the 10 objects, and then enter:

```
js_createControlCurve controlObj height ty
```

<sup>ix</sup> **js\_pivot\_create** is part of the **js\_pivot.mel** script.

<sup>x</sup> **js\_pivot.mel** is a series of procedures used to handle animating pivots on your characters.

<sup>xi</sup> **js\_cutPlane.mel**

<sup>xii</sup> **js\_rotationOrderWin.mel**

<sup>xiii</sup> **js\_createStretchSpline.mel**