PARENTING AND CONSTRAINING IN MAYA

INTRODUCTION

Hello everybody, my name is Adam Bradford. If you're reading this tutorial you probably have trouble with constraints or parenting in Maya, or you're one of many animators who do not consider themselves technical. Well, you're in the right place. In this tutorial I will guide you as we learn about constraints and parenting, the differences between the terminology, and move up to different ways to use constraints within your animation workflow. At the end of this tutorial you will have a working knowledge of constraints and will be able to apply the concepts to just about any issue you encounter. In other words my intention is to give you as much knowledge and understanding as possible, but it is up to you to experiment on your own after you've learned the basics. Constraints can be difficult and frustrating, even for me. I've been working with Maya for about 5 years now and even I still have trouble with them from time to time (mostly because of their slightly flawed workflow within Maya). But do not let this discourage you, as any problem always has a solution. I firmly believe that having a deeper understanding of the technical aspects within Maya will make you a better, more efficient animator.

Constraints are probably the biggest thing I see my fellow animators struggle with. I've almost become known as the go-to guy amongst my friends when someone is having trouble with a constraint setup and it's not quite working for them. I made a tutorial about a year and half ago as of writing this, but I wasn't satisfied with the explanations or the quality of the video itself. So I've set out to remake that tutorial, and I'm aiming for a much higher level of quality this time around. I realized, though, that it'll be hard to get it perfect, and different people learn in different ways. So this written portion of the tutorial is basically supplementary to the video, to suit different learning styles. All examples were done in Maya 2012 on 64-bit Windows 7, but, the concepts and setups are platform independent as far as I know.

We will cover the following concepts:

- Spaces
  - Parenting
  - Hierarchies
- Constraining
  - The differences between Parenting and Constraining
  - Space Switching
- the BlendNode
- Using the “Best of Both Worlds” with Parenting and Constraining
- Baking animation
- Cycles and why/how to avoid them
- Practical Examples of common constraint setups

Without further ado, let's dive right in!

The following conventions will be used:
Definitions are bold and red. Keywords are bold and green. Keywords that have already been defined are bold. Maya UI Element Names are italicized. X Y and Z attributes are their respective colors.

Please note that although it seems like there's quite a lot of pages to this tutorial, a lot of that space is taken up by often quite large images. I've tried to size them in the way that makes the most sense so that the information flows intuitively. I've also separated the “chapters” by a blank page for a logical separation of the concepts. Hopefully though, they still flow coherently and intertwined with each other.

I've also sprinkled a few topics here and there that don't necessarily directly pertain to parenting or constraining, but I find them useful and figured I'd at least pass on my knowledge, as they have helped me greatly in my own workflow.
First let's discuss the ever important topic – and quite frankly the topic that everything else is built on – of spaces. What is a “Space”? A Space in the context of a Maya scene, is a world that an object resides in. Any time you create an object in Maya, it is created in the world. This World is essentially a coordinate system that the object moves around in. If you create an object, say, a sphere primitive, it is created at the Origin of the world, or, 0,0,0 in Translate X Y and Z. If you move the sphere primitive around in space, all of its translate values are a relative offset to the Origin.

If you're more than a beginner in Maya or 3D in general, this shouldn't be new to you, but I promise it's an important concept to understand nonetheless, hence why we're covering it.

As you can see, the sphere is now translated in its X Y Z values, and these values are relative to the World Origin. If an object's coordinate system is relative to the world origin, it is in World Space.

Let's suppose we want to create a Parent-Child relationship. Create a second sphere by
PARENTING AND CONSTRAINING IN MAYA

pressing Ctrl + D or Cmd + D to duplicate the one you have selected. Move it out to the side and scale it down slightly so we can tell the difference between them. Rename the big sphere ParentSphere_01 and the small sphere ChildSphere_01 (it’s always a good idea to rename your objects and keep your scene tidy).

We can create a Parent-Child relationship in one of several ways:

- With the ChildSphere_01 selected, Shift-Select ParentSphere_01.
  - The easiest way is to press P on your keyboard.
  - You can also go to Edit > Parent.

Figure 2 – The Edit > Parent Menu Option

- Alternatively, you can open your Outliner by going to Window > Outliner, selecting
PARENTING AND CONSTRAINING IN MAYA

ChildSphere_01 and middle-click-dragging it on top of ParentSphere_01.

Since you have your Outliner open, let's discuss what we just did. If you click the plus sign next to ParentSphere_01, the ChildSphere_01 appears down and to the right of ParentSphere_01. This is a sort of graphical representation of a Hierarchy, which is a system for maintaining Parent-Child Relationships. If an object appears below and to the right of another object in the Outliner, it is a child of that object.

There are some distinct facts to know about a Parent-Child Relationship:

- It can't be turned on or off (toggled), it either exists or it doesn't.
- Any child object inherits all the attributes of its parent object (translate, rotate, scale – X, Y, and Z). The child's world is now the parent object itself; the parent is the origin of the child's coordinate system. All translation values are now a relative offset of the parent rather than the world.
- The child object is now in Parent Space.
- Maya still maintains the information of the child's offset to the world behind the scenes. This is known as its absolute transformations.

Let's find out how to demonstrate this. Move the ChildSphere_01 so it has values in all 3
translate channels. Pay close attention to these channel values and press **Shift + P** to unparent the sphere. Did you notice how they changed? The sphere’s translate values just shifted from **Parent Space** to **World Space**, and yet the child sphere didn't visibly move in the viewport because Maya handles the transition seamlessly.

Figure 4 – ChildSphere_01’s translate values **relative to its Parent** and **relative to the World** after being unparented
PARENTING AND CONSTRAINING IN MAYA

There's another set of terms that it is wise to be familiar with, and that pertain to the above concepts: Local Mode, Object Mode and World Mode. These modes are different ways of translating an object. With Object Mode, the translate manipulator is aligned to that objects space and orientation. With Local Mode, if it has a parent, it is aligned to the parents orientation axis. With World Mode, no matter the parent objects orientation, the translate manipulator is always aligned to the World Space.

We'll quickly demonstrate this. The easiest way to go into Local Mode is to select the object in question, in this case ChildSphere_01, hold the W key (the hotkey for switching to Translate Mode), left-click, and drag to “Object” in the Hotbox menu that appears:

By default, if the object is either in World Space or its Parent Object's rotation axis is aligned to the world, Object translations will be the exact same. If, however, the alignment of the Parent Object is different, you'll have to keep in mind that the translations are different.

- Take ParentSphere_01 and rotate it slightly, even in just one axis if you so choose.
- Now try moving the ChildSphere_01 around in Object Mode, but only in one axis.
- Notice it only transforms in that exact axis that you've chosen. If you were to switch back to World Mode using the same switching method, and then try translating in that single axis, it would move a combination of two or all three of the translations.
- The converse is true if it is in Object Mode but in World Space.
Now we'll play around with the object in **Local Mode**. With **Object Mode**, if the Parent Object is moved and/or rotated, as well as the child, the Child's orientation axis no longer represents the true axis it's being moved in (combination of axes rather than just one). We can change it so that we know we are translating exactly aligned to the Parent Object's orientation axis. In that same **Marking Menu**, go down to the bottom left and select **Axis**. A **sub-Marking Menu** appears, now select Local on the left side.

![Figure 5b – The Local Axis Translation Hotbox Menu Item](image.png)

- Play around with the spheres, in **Object Mode**, **Local Mode**, and in **World Mode**, Parented, not Parented, to understand the difference. We'll circle back and discuss more in the practical examples section.

It's important to understand the differences in these modes because they'll affect the way your curves look in the Graph Editor. We'll discuss this even more in depth in the Practical Examples section.
Now let's discuss Constraints! Ooh buddy, this is it. Constraints are basically the central topic of this entire tutorial.

What is a Constraint? A **Constraint** is a way to attach one object to another but with more flexibility than with parenting. Let's discuss several key differences between **Parenting** and **Constraining**, and then we'll learn how to set them up.

- Constraints can be used to affect only certain channels/attributes
  - You can set them to affect only Translate, Rotate, Scale, X, Y, and Z, or any combination, depending on the **constraint**

- Constraints can be “turned off”
  - Whereas an object is either Parented or not, **Constraints** have a “Weight” attribute that allows them to essentially be disabled.

- You must select the Driver and then the Driven object instead of the Child and then the Parent
  - We'll use the terminology as “constrain the Driven to the Driver” or “parent the Child to the Parent”

- **Constraints** override **Hierarchies**
- **Driven** objects can have more than one **Driver** object
  - Whereas Child Objects cannot have more than one direct **Parent**
- Driven objects cannot be animated freely (with one exception which we'll discuss)

It is important to re-iterate the differences between **Parenting** and **Constraining**, besides the list above. With parenting, you select the child object, then the parent, and press P to insert it into the hierarchy. If something is in a hierarchy, it is Parented. With constraints, you setup connections with special nodes that can be turned on and off, and they override Hierarchies. Also note that the Parent Constraint itself is still only a constraint that mimics a Hierarchy.

We'll refer to them as “Parenting” and “Constraining” as such for the rest of the tutorial.
Let's discuss each point one by one and demonstrate how to setup a constraint. First, make sure you're in your Animation module menu by pressing F2 on your keyboard. Then find your way to the Constrain Sub-Menu on the right side. Let's find out what the basic ones do:

- **Point**: constrains the driven objects translation attributes
- **Aim**: allows the driven object to “look at” the driver object (we won't necessarily go in depth with this one, it's quite advanced to setup)
- **Orient**: the **driver** controls the **driven's** rotation attributes
- **Scale**: the **driver** controls the **driven's** scale attributes
- **Parent**: directly mimics a Parent-Child relationship (translation and rotation)

For now we won't worry about the options below those 5 basic constraints. They're a bit more advanced.
PARENTING AND CONSTRAINING IN MAYA

Go ahead and click the Option Box on the Point Constraint menu item, and the Option Window opens. These are the settings you'll see by default:

Let's discuss what these do, shall we?

- **Maintain Offset** is just how it sounds: if there is an offset between the two objects, it is preserved when the constraint is added. By default this is unchecked, and there are three boxes to specify the offset manually. If you want the object's position in space to remain the same, you should check this option. Typically this will be the case.

- **Animation Layer**: we won't worry about this particular option for the time being

- **Constraint Axis**: pretty self-explanatory. You can specify to constrain all 3 axes or just 1 or 2 of your choosing. This is one of the differences with constraints as we discussed earlier.

- **Weight**: this allows you to control the amount of influence the constraint has. We'll look into this when we setup multiple driver objects. It also allows you to “turn off” the constraint, which we'll also discuss later.

If your ChildSphere_01 is still unparented, select ParentSphere_01, shift-Select ChildSphere_01, and click Apply or Add in the Option Window (for the Point constraint). You can now move the ParentSphere_01 around and the ChildSphere_01 will follow in it's translation.
We can tell the constraint is added in one of several ways:

- In Wireframe Mode, the driven object appears as a pinkish color when the driver is selected.

- In the Outliner, there is now a constraint node below and to the right of the driven object.

- In the Channel Box, the attributes that were specified are now highlighted in blue.
The options for Point, Orient and Scale are the exact same. The difference with the Parent options is that you are allowed to specify the particular translate and rotate attributes instead of one or the other.

Go ahead and setup the other constraints with these same options and see what they do. Try checking and unchecking maintain offset for each one to understand the difference, or specifying only certain attributes instead of all of them (this is the part where you experiment).

The perceptive reader will notice that the Point and Orient control the translation and rotation of the driven, respectively, and yet the Parent seems to do so as well. So, what's the difference? The key difference is that the rotation pivot point of the driven object in a Parent constraint is at the pivot point of the driver object, whereas if you were to setup a Point and Orient on the driven object, the driven object would rotate from it's own pivot point. So, depending on where you want the driven to rotate from, you can add a Point + Orient combo, or just a Parent.

Notice that I said to make sure that ChildSphere_01 was unparented from ParentSphere_01 before adding the constraint. This was to make sure that they were not in the same hierarchy. If you're going to constrain an object, you most likely want to constrain objects that are in a different hierarchy. If ChildSphere_01 were still parented under ParentSphere_01, there would be no point in constraining it (no pun intended). This leads us to the next difference between parenting and constraining: constraining overrides Hierarchies. Let's solidify our understanding of how this happens.

- Select the point constraint node of the driven sphere in the Outliner and delete it
- Create another sphere primitive and rename it ParentSphere_02. Parent the ChildSphere_01 under the new parent sphere. ParentSphere_02 now controls ChildSphere_01's attributes.
- Now point constrain ChildSphere_01 to ParentSphere_01

Figure 09 – The Child Sphere is Parented to one sphere and constrained to the other
PARENTING AND CONSTRAINING IN MAYA

If you try translating the ParentSphere_02 you'll notice it no longer has control of the child sphere's translate, but, you can still control it's rotation and scale. Now if you move ParentSphere_01 around it controls the child sphere's translation but nothing else. ParentSphere_01 and ParentSphere_02 are in completely different hierarchies, and ParentSphere_01 is still overriding ParentSphere_02. It is important to understand this concept out of anything else up to this point, so make sure you've got a handle on it.

I feel like I should take a bit of a side-track and discuss the underlying architecture of Maya in the simplest way possible. **Anything in Maya that has modifiable attributes and isn't a UI Element** is known as a **Node**. Nodes can have any number of connections to other nodes; these connections are essentially feeding data from one node to another, and the nodes themselves are doing computations of the data. There are many different types of nodes as well as many different types of connections. These can be constraints, direct connections, expressions, or animation data. The attributes in the Channel Box can be highlighted differently to denote different kinds of connections. Suffice it to say that **Maya at it's very core is comprised of nodes that feed data to each other**. The reason I think it's important to know this is because every time you add a constraint, the constraint node is feeding data into the translate node, and that is why the appropriate channels appear blue: to represent incoming connections. The following image is the **Hypergraph**, which shows the flow of data between nodes based on our current scene:

![Hypergraph](image.jpg)

Figure 10 – Hypergraph Nodes and Connections
There's another way we can learn how the flow of data works, and this is done through the Connection Editor. The Connection Editor is simply a more user friendly version of the Hypergraph, but both still accomplish the same thing: connecting nodes together.

Open up your Connection Editor by going to Window > General Editors > Connection Editor. If you have one of the Parent Spheres selected, this window will look something like this:

![Figure 11a – Connection Editor](image)

This window has 2 columns. The node on the left will make a connection with the node on the right. If you have an object selected as you open the window, it will automatically load that into the left column. If you have a second object selected it will load that into the right column. If you want to refresh the column with a newly selected object you can press Reload Left or Reload Right, depending on which node you want connected. We're not too worried about all these scary looking attribute names that are loaded by default, so go to Left Display and uncheck Show Non-Keyable. Do the same for the Right Display. Now only the attributes that show up in the Channel Box are what's displayed, which is all we care about. To make a direct connection, select the attribute or attribute(s) on the left first, then the attributes you want connected on the right; you must always select the attributes in this order. These are direct connections because there's no way to create an offset like with a constraint; everything is a 1:1 ratio. **Direct Connections** appear as yellow in the Channel Box.
We won't necessarily be using this technique as we'll just be concerned about constraints from here on, but I figured it's good to know a different way to make a similar connection. This is another part of the tutorial where you should take some time to experiment; try connecting the translate to the rotate or the scale to the translate. You can get some interesting results that may or may not be useful. Nevertheless, it's good to know how to do.

Let's briefly recap this: You can make a driver object control a driven object through a constraint, which allows for maintaining offsets. You can also create a direct connection through the Connection Editor but the offsets will not be maintained. Both situations will override a standard Hierarchy.

Up to this point we've only discussed the possibility of having one driver object with the possibility of the driven object also having a separate parent object. Now we'll discuss the next difference between parenting and constraining: driven objects can have more than one driver object, but child objects cannot have more than one parent.
If you're following along and you still have that same scene open, we'll set this up:

- Delete the ChildSphere_01's Point Constraint node from the Outliner.
- With ChildSphere_01 selected, press Shift + P to unparent it.
- We're left with ParentSphere_01, ParentSphere_02, and ChildSphere_01, all as children of the World.
- Select ParentSphere_01, Shift-Select ChildSphere_01, and add a Parent Constraint without Maintain Offset checked.
- Repeat the same for ParentSphere_02 and ChildSphere_01.

ChildSphere_01 is now being driven by two objects. We can see this most clearly by looking below the ChildSphere_01 in the ChannelBox, under Shapes, the parent constraint node should be visible.

![Figure 11 - The Point Constraint node in the ChannelBox](image_url)
Notice the naming convention that is used: The name of the object, followed by a W, followed by an iteration number starting with 0. The W stands for Weight, which in the context of constraints is the amount of influence a driver object has over a driven object.

If you have more than one driver object, the Weight attributes determine which one has more influence. If they both have a Weight of 1, they have an equal amount of influence. If you were to try and move either ParentSphere_01 or ParentSphere_02 around, you will notice that ChildSphere_01 will move exactly half-way between the two. If you turn the Weight to 1 on one, and 0 on the other, the one with full Weight will influence the object 100%. If you switch from one driver to another, you're switching into their respective coordinate spaces. In other words, fully switching from one driver object to another, or, switching from one coordinate space to another, is known as Space Switching. This is the same thing we did when ChildSphere_01 was simply parented to ParentSphere_01 and we unparented. ChildSphere_01 was switching from ParentSphere_01’s space to World Space. The difference with switching a constraint weight is that these very attributes can be animated; you can't animate the space switch of a Parent-Child relationship like you can with constrained objects.
PARENTING AND CONSTRAINING IN MAYA

The Blend Node

Up to this point we've only discussed having a static child or driven object; we haven't actually animated anything yet. The major difference with animating a child object vs a driven object is that child objects can be animated freely and still follow their parent object; whereas a driven object has to have an intermediate node called a Blend Node. The **Blend Node allows a driven object to switch from being controlled by a driver object to moving around on its own while being animated.** The reason this happens is because a driven object will have incoming connections to its attributes while it's constrained, but when animation data is added (i.e. a key is set) there will be different connections coming in. A node in Maya cannot have more than one incoming connection to a single attribute. Maya handles this with the intermediate **Blend Node.** The **Blend Node** controls the flow of data coming into the attributes of a driven object and switches between multiple incoming connections. If this is beginning to sound too wordy, allow me to demonstrate with pretty pictures and bullet lists:

- Begin by setting a key on **ChildSphere_01** while it's still constrained (In this case with a Parent Constraint)
- A new attribute appears called **BlendParent1**
- The attributes that are connected become green in the ChannelBox, this is to designate that they are controlled by a **Blend Node**

![Figure 13 – The Blend Attribute is added when you set a key on a constrained Object](image-url)
I've used the term Blend Node because it’s generic enough for the actual names that are used. In this case a BlendParent1 is used, but depending on which constraint is connected, it could be called a BlendPoint, BlendOrient, or BlendScale.

By default the BlendParent1 attribute will be set to 0. **If the Blend Node is set to 0, the animation curves control the driven object; if it's set to 1, the driver controls the driven object.** If the animation curves are controlling the driven object, it can essentially move around on its own. **This is the only way you can move around a driven object after it's been constrained.** With the Blend Node at 1, the Weights of the multiple driver objects still take effect as they normally do.

Try setting the full Weight influence of one of the driver spheres to 1 and moving it around to control the driven sphere, and of course having the BlendParent set to 1. Now try setting the Blend attribute back to 0 on a different frame. Do you notice what happens? Depending on your default key tangent type, it could be smoothly transitioning from 1 to 0 over several frames. I can tell you from experience that this is less than ideal in most situations. **It's best to have the Blend Node switch from 0 to 1 or 1 to 0 over a single frame.** The same goes for the Weight attributes.

The problem with this, as I've experienced recently in one of my own shots, is that it's a lot of work making sure the Blend Nodes and Weight attributes switch over a single frame. When you're in the zone, animating everything else, it basically becomes a nuisance, and you can still find the Blend Node switching smoothly over several frames. The reason this happens is because the Blend Node is a float attribute, meaning it can have several numbers to the right of the decimal point (floating point). This means that it can have any amount of values in between 0 and 1. This next setup will be optional, but highly recommended. We'll essentially change the Blend Node attribute into an integer, meaning only 0 or 1, and no blending in between.
What we'll do is create another attribute, just below the BlendParent1, and have that control the original.

- With the ChildSphere_01 still selected, go to Modify > Add Attribute
- In the Long Name text field, put something like BlendParentInt1, or whatever suits your tastes
- Under Data Type, select the Integer bubble option
- Under Numeric Attribute Properties, set the Minimum, Maximum, and Default to 0, 1, and 0, respectively. It's important that these are the correct values.

Figure 14 – The Add Attribute Window with the correct options added
Now we'll use the technique we learned earlier for connecting attributes:

- Open your Connection Editor with ChildSphere_01 still selected.
- It should load on the left column by default. Click Reload Right.
- That's right, we're going to connect two attributes of the same node. Select BlendInteger1 on the left column and BlendParent1 on the right column to connect them.

![Figure 15 – The Connection Editor with the Child Sphere attributes and their connections](image)

BlendInteger1 can now only go from 0 to 1, and because it's feeding its data to BlendParent1, it basically “constrains” it to these values, and doesn't allow it to blend in between. What we did is basically convert BlendParent1 from a float to an integer by way of a secondary attribute, to make our workflow a bit easier.

There's still one inherent flaw with this workflow: the driven object will “snap” or “pop” when it's being switched from one coordinate space to another; either by switching the Weight of the driver object, switching the Blend Node on or off, or both. But, with every problem, there's a solution, and this is no different.
PARENTING AND CONSTRAINING IN MAYA

Space Switching

The next thing we want to be able to do is get rid of this pop, or what we'll call **Seamless Space Switching**. You'll remember that I mentioned that Maya keeps the information of an objects world space coordinates behind the scenes. Well, it turns out it's pretty easy to access this information through a MEL script and use it to our advantage. Don't worry too much about it being a script if that's something that scares you. We won't go too in depth into MEL scripting as that's yet another thing that's quite outside the scope of this particular tutorial. I'll only show you the script itself and how to use it. But first, an overview of the steps themselves:

- “Get” the values of the world space coordinates of the driven object
- Switch it's coordinate space, either with the Blend Node, the driven Weight, or both
- “Set” the values back to the world space coordinates

This will create what appears to be a seamless transition from one space to another, hence the name.

With these next few steps we'll setup a basic shelf button

- Open the Script Editor
  - Go to Window > General Editors > Script Editor
- Copy the Get script into the bottom portion
  - It's not a bad idea to memorize, so go ahead and type it out manually

Here's the script in case you don't feel like typing it:

```mel
// get xform
float $t[] = `xform -q -ws -t`;  
float $r[] = `xform -q -ws -ro`;  
```
PARENTING AND CONSTRAINING IN MAYA

- Make a Shelf Button
  - Select the script as shown and left-click and drag it to your shelf
  - Rename the shelf button “GetX” from the Shelf Editor
- Repeat for the SetX shelf button

![Figure 16b – The SetX Shelf Button](image)

```plaintext
// set xform
xform -ws -t $t[0] $t[1] $t[2];
xform -ws -ro $r[0] $r[1] $r[2];
```

When you're done you should have GetX and SetX shelf buttons. Test it out on one of the spheres:

- Move the sphere in space
- Press the **GetX** shelf button
- Move it around again
- Press the **SetX** shelf button
- The sphere snaps back to where it was
Another technique at our disposal is what's called the Rest Position. I actually just found out about this as I was writing, so I thought it would be helpful to include in our bag of tricks. The Rest Position is the position an object moves to after the weight of its constraint influence has been turned off (set to 0). By default, the rest position is the position the object was in before being constrained. We can change the rest position just before we turn off the influence of an object to achieve basically the same effect as our space switch script: the object stays in the same world space position. You can change the Rest Position of a constrained object by going to Constrain > Set Rest Position with the constrained object selected.

- Rotate the Driver sphere around so the Driven sphere moves with it.
- Change the Weight of the Driver's constraint to 0. The Driven sphere will snap back to its initial rest position.
- Change the Weight back to 1
- With the Driven sphere selected, go to Constrain > Set Rest Position
- Change the Weight back to 0. The Driven sphere now stays in place.

This can come in very handy, and we'll use the Rest Position and our GetX/SetX scripts in tandem in the Practical Examples section.
Before we can apply a seamless space switch to our child object, we need to learn one more subject, and that is getting the best of both worlds with parenting and constraining.

What I mean by the best of both worlds is that we can combine the functionality of parenting and constraining to get the most flexibility. We'll do this by creating what we'll call a buffer object. This will be an intermediate node in between the driver and the driven (but quite different from a Blend Node). This will have the added benefit of not having to deal with the Blend Node in most cases (something I prefer). So if we imagine this like a flow chart it will look something like this:

So essentially we'll create a buffer object, constrain it to the **Driver**, then parent the **Driven** object to the buffer object. The **Driven** object will be able to move around freely because it's parented to the buffer node while the buffer node is constrained to the **Driver**. The idea is that we don't want to touch the buffer node once it's constrained, and therefore we don't have to deal with the Blend Node in most cases (one less attribute to switch).

We'll be using a Locator, which is just a simple crosshair object in your scene that doesn't show up during render time. Locators are a simple yet very useful tool to use in conjunction with Parenting and Constraining, and pretty much everything from here on out will involve Locators.

In this section we'll discover how to create a “Best of Both Worlds” setup, as well as setting up what's called a Set Driven Key to speed up our workflow for using the Best of Both Worlds setup.
PARENTING AND CONSTRAINING IN MAYA

• Go ahead and create a fresh scene (it's easier than undoing the exact number of steps)
• Create 3 spheres like before and name them accordingly, but don't constrain them yet
• Create a Locator by going to Create > Locator
• Parent the Locator under the child sphere
• Zero out the translate and rotates of the Locator
• Press Shift+P to Unparent the Locator
  ◦ Now the Locator is in exactly the same world space as the child sphere
• Now reverse the parent by parenting the child sphere under the Locator
• Parent Constrain the Locator to each Parent Sphere
• Give one of the Parent Spheres full Weight influence
• Test out the new setup by moving the Parent Sphere and then moving (and/or animating) the Child Sphere. The Child Sphere should be able to move around freely while still being controlled by the Parent Sphere
Much like the BlendInteger1, we'll setup a custom attribute to switch the Weight of the Driver objects so it's easier than switching the Weights to 1 and 0 or vice versa.

- Go to Modify > Add Attribute with the Child Sphere selected
  - We want to be able to switch the Parent “on-the-fly”, and adding the custom attribute to the child is the best way to do this (as opposed to one or both of the parents)
- Name the Attribute “Space” to represent the space switch attribute
- Select the Enum bubble option
  - This is one of those attributes that when you select opens a drop-down menu to select from
- In the lower portion of the Add Attribute Window, two options named Green and Blue have shown up. Select Green and rename it to Parent01 in the text field below
- Select Blue and rename it Parent02
- Click OK

![Add Attribute Window](image)

Figure 18 – The Space Attribute in the Add Attribute Window
Now we'll set up a Set Driven Key to switch these attributes (I know them as “Set” Driven Keys or SDK’s for short, rather than “Driven Keys” - either way it’s interchangeable). Essentially Set Driven Keys allow one Attribute to drive another attribute (or attributes). When animating regular objects you change values over time, or, a value-to-time ratio. **With SDK’s, we set up a value-to-value ratio** (it’s a special way to hook up animation curves). The idea is that we setup an Enum attribute to switch from one Parent Sphere to the other, and under the hood this attribute switch is changing the Weight values opposite each other. So in other words **we’re setting up a single attribute to switch multiple attributes**. Let’s set it up now.

- In the Animation Module Menu (F2), select Animate > Set Driven Key > Set. The SDK Window opens

![Figure 19 – The Set Driven Key Window with nothing selected](image)
PARENTING AND CONSTRAINING IN MAYA

- Select the Child Sphere, and click Load Driver in the SDK Window

Figure 20 – The SDK Window with the ChildSphere as the “Driver”
PARENTING AND CONSTRAINING IN MAYA

• Expand the Locator object from the Outliner (probably not a bad idea to rename it to something like “ChildSphere_Buffer” or something similar) and select the constraint node underneath it.

• Select Load Driven

• Select the “Space” Attribute that we added in the top right column

• Select the “ParentSphere01W0” and “ParentSphere02W1” attributes in the bottom right column. We've selected the driver and driven object in the left columns, and their respective attributes that we want controlled in the right column

Figure 21 – The SDK window with everything setup and selected
This is where it could potentially get confusing. You need to be sure of exactly which Parent Sphere is which, and which attribute is which in the constraint node. If you've done it right, the order of the Parents in the Enum Attribute that we added will match the order of the Weight Attributes in the constraint node. This is ideal for less confusion.

Note also that the naming might be a bit confusing. The “Driver” spheres are controlling the “Driven” spheres via the Parent Constraint setup, but in the case of the Set Driven Key, the “Driven” Sphere is also the “Driver” of the constraint node which is being “Driven”. The nomenclature can tend to get confusing, but hopefully this clears it up.

• With the Child Sphere selected, make sure the Space attribute is set to Parent01.
  ◦ It's important to always change the attribute of the driver object first. Note that I'm not talking about the driver object of the constraint setup, but rather of the SDK setup. In this case they're basically opposite.

Figure 22 – The SDK “Driver” object attribute
PARENTING AND CONSTRAINING IN MAYA

- Select the constraint node via the SDK Window in the bottom left corner (clicking the list items will select the object itself), and make sure the Weight of ParentSphere01 is set to 1 and the Weight of ParentSphere02 is set to 0.
- Hit the Key button in the SDK Window.

Figure 23 – The Weight Attributes selected and changed in the Channel Box
PARENTING AND CONSTRAINING IN MAYA

• Now switch the Space Attribute to Parent02. If you see where this is going you can probably guess the next step.
• Select the constraint node once again and switch the ParentW1 from 1 to 0 and the ParentW2 from 0 to 1
• Hit the Key button again

You can verify if you did it correctly by selecting the Child Sphere and switching the Space attribute. If the sphere “snaps” in space, then you did it correctly (unless of course your Locator buffer node hasn't been moved). You can also check the Weight attributes just to be double sure (they'll have red channels in the channel box, which represents that they have keys, but they won't have keys in the timeslider)

Now we can combine our GetX/SetX shelf buttons with our custom attributes for a perfectly seamless switch
• Set the Space attribute to Parent01
• With the Child Sphere selected, press the GetX shelf button
• Switch the space to Parent02
• Press the SetX shelf button
• The Child Sphere remains in the exact position, despite having “snapped” when we switched spaces

You can further test this by animating the child sphere, and doing the GetX > Space Switch > SetX steps over a single frame This our ultimate seamless space switch
• Move the Child Sphere from frames 1-12 with it in the Parent01 space
• On frame 13, press the GetX shelf button
• Switch it to Parent02's space
• Press the SetX shelf button
• Continue moving the sphere from frames 13-24

You now have an object:
• Being controlled by multiple driver objects
• Animating freely despite being constrained (via the Locator)
• Seamlessly switching from one space to another while being animated.

These things would not be possible without leveraging the “best of both worlds” of parenting and constraining and using our custom GetX and SetX buttons
PARENTING AND CONSTRAINING IN MAYA
Let's learn about a common pitfall when setting up constraints: cycles. Cycles are created when the flow of data loops on itself infinitely; this can and will create unpredictable behavior and is best to be avoided at all costs (unless you know exactly what you are doing, and even then it's not recommended). Common ways to setup a cycle are:

- Constrain a Parent to its Child
- Constrain a Driver to its Driven
- Create a Direct Connection from an attribute back into itself

Essentially you must ask yourself, “what is this objects parent?”, or “what is this objects driver?”, before adding a constraint or connection. If you have the Command Line open at the bottom of your Maya Window, a yellow warning will pop up saying you've created a cycle. Undo immediately and re-assess how to create the setup you were going for if this happens.

Figure 24 – A common Cycle error. To be avoided no matter what
Baking

We'll cover one final sub-topic that doesn't necessarily have to do with Parenting or Constraining itself, but when used together can be quite powerful. We'll talk about what is known as Baking, and of course I'm talking about the term Baking within the context of Maya (although I have been known to make some pretty awesome peanut butter chocolate chip cookies :)

Baking is actually something that I don't think most animators would be familiar with because it is something that is used for simulations. If you have a huge scene with lots of physics calculations going on, it can be quite CPU-Intensive to try and scrub your timeslider to see the effects of the simulation. Well, **baking** is the process of locking down the results of a simulation, essentially to make real time playback more feasible. Lucky for us, this is not the only thing it can be used for. We can basically “lock-down” the results of a constraint setup as well, and this provides a lot of flexibility in what can be accomplished.

If you still have that scene open, go ahead and use that, or create a new one. Either way we'll continue using our trusty spheres:

- Create a constraint setup with one sphere controlling another
- Animete the driver sphere so that the driven moves around in space a good amount
- Select the driven sphere
- Go to Edit > Keys > Bake Simulation > Option Box
For our purposes, the default settings work just fine. Go ahead and click *Bake*

The driven sphere now has a key on every single frame and the connections from the constraint are no longer there

Some people don't like having a key on every frame, but personally I don't mind it. It's pretty trivial to delete the keys that are unnecessary when you're done, and there's always the option of using the Smart Bake, or changing the Sample By rate. It's also wise to note that you should pretty much always bake down the buffer Locator node, instead of the driven object itself. We'll discuss why shortly.
PARENTING AND CONSTRAINING IN MAYA

The Real Deal: Practical Examples

I promised myself when making this second version of the tutorial that I would use better examples because in the last one I simply used parent and child spheres. Well, what I've done up to this point is explain all the underlying concepts....with spheres. I figure they're just simple objects to explain all the concepts with, so hopefully everything up to this point has made enough sense with those as the examples. Fear not, for we will now move on to real animation examples of how to use everything we've learned up to this point. But first I think it's worth recapping all the concepts in one fell swoop.

- A **space** is a coordinate system for a 3D object to move around in
- The center of the world coordinate system is the origin, or 0,0,0 in X Y and Z
- Everything is created at the **origin** by default
- A **Hierarchy** is a system for maintaining Parent-Child relationships
- Parent-Child relationships cannot be toggled, they either exist or they don't
- A **Child** object is in it's **Parents** Space. The **Parent** is the center of its coordinate system. If an object is not a **Child** it is in **World Space**
- A **Child** object can animate and move around freely in it's **Parent Space**
- **Constraints** can be used to drive other objects, but are not in the same hierarchy
- **Constraints** can affect only certain channels, and be “turned off” with the **Weight** attribute
- A **Child** can only have one parent but a driven object can have multiple driver objects
- Multiple Driver objects can share their influence with their respective weight attributes
- If you want to animate a driven object, you must use the Blend Node attribute
- The best way to animate a driven object is to parent it under a buffer object, or locator, which is constrained to the driver. The driven object can be animated freely, but the flexibility of constraints is still provided
- Maya is made up of **nodes** that feed data to each other
- You can control the flow of data by making **connections**, be it by direct connections or **Set Driven Keys**
- You can create a seamless space switch with a couple simple shelf buttons
- **Cycles** should be avoided at all costs
For the remainder of the tutorial I will try to discuss a few common examples of how to use the concepts that we've learned up to this point. However, I can't possibly expect to give you a “one size fits all” scenario; in other words I can't give you an example for every constraint issue you will encounter. The idea that I've had in writing everything up to now is to give you all the tools you need, but it's up to you to decide how and when to use them. I've also tried to give you those tools basically in isolation; so when we get into the practical examples and I say, “Constrain ObjectA to ObjectB”, or, “Create a custom attribute to switch the Weight of the constraint”, you should know exactly what that means and how to do it.

We'll cover the following examples:

- A Character holding an object and many “sub-scenarios”
- Moving a pose around in space
- Creating a “reverse hierarchy” for a swinging character

For these examples I'll be using the freely available Norman rig so you can follow along. Project files will be provided with the tutorial download link. So let's get to it, shall we?

Prepare for lots of images!
This section we will be using **01a_PF_NormanWithSword_Start.ma**. Go ahead and open it up and check out the file. You'll see Norman in his default pose and a simple sword that I modeled; this is what we'll be using for the prop object.

- This is the default setup that we'll assume you would encounter in a typical scene
- Create a NURBS Circle control by going to Create > NURBS Primitives > Circle
- Rename the circle sword_ctrl
- Move it so it's centered around the bottom of the blade of the sword
- Go to Modify > Freeze Transformations and Edit > Delete By Type > History once you have it centered on the sword.
  - Freeze transformations zeroes out the transforms of the object while still maintaining its position in World Space (in other words it doesn't snap to the origin). If you freeze transforms of an object in space, that's where it will be when you zero out its translate and rotate channels.
  - Deleting the history gets rid of all the incoming connections and keeps things clean

---

Figure 26 – The Sword Control NURBS Circle centered around the sword
PARENTING AND CONSTRAINING IN MAYA

- Constrain the *sword_geo* to the *sword_ctrl* with maintain offset; the *sword_ctrl* should drive the *sword_geo*
- Grab *ctrlX_Rt_ArmExtra* and *ctrlX_Lf_ArmExtra* and switch the FkIk attributes to 1 (switching the hands to IK) (I'll refer to them as right hand control and left hand control from here on, for simplicity)

![Channel Box / Layer Editor](image)

Figure 27 – The Hand FkIk control attributes switched to 1

- You don't necessarily have to worry about putting Norman into a cool pose, but I've taken the time to at least take him out of his boring default pose
- Move the sword up to near his chest and move the hands so that they're grabbing the sword
- Now that we have the sword and the hands in place, and a cool pose to work with, go ahead and save a new version of your file (keep in mind we haven't constrained anything yet other than the sword control)
Now constrain the left hand control to the sword control, and then the right hand control to the sword control, with Maintain Offset checked

- Note that it is important to do them one at a time. If you were to grab, for example, the sword control, the right hand control, and then the left hand control, and try to parent constrain, it would add a constraint to the left hand control (the last thing selected) and add the sword and the right hand control as drivers with equal Weight values. This is not the effect we want, so it's important to only have two objects selected at a time when constraining.

What we should have at this point is Norman holding the sword, and the sword controlling the hands. This is well and good but we'll come across some issues:

- First, it's not very animation friendly. If you were to select all of Normans controls and set a key, it would add a Blend Node on his hand controls because they're constrained. Then you have to go in and turn the Blend Attribute to 1 to make sure the hands still follow the sword (not to mention adding a custom Integer attribute to prevent blending).

- Secondly, what if we want to adjust the placement of the hands, but already have animation? This setup is not ideal to do so, so let's go ahead and back track a bit.

Figure 28 – Norman Holding the sword
PARENTING AND CONSTRAINING IN MAYA

- Open up the file that you saved before adding the constraints to the hands.
- Create a locator by going to Create > Locator.
- Select the `sword_geo`, then the locator, and Parent Constrain with Maintain Offset unchecked. This will snap the locator to the pivot of the sword.
- Select the locators constraint and delete it. This is a quick and easy way to align two objects.
- Rename the locator `R_Hand_Attach` or something that suits your naming conventions. The point is to be consistent if nothing else.
- Drag the locator straight down until it’s just below the hilt.
- Parent the Locator to the `sword_ctrl` NURBS circle.
- Repeat the locator steps with a second one and call it `L_Hand_Attach`. Move it slightly below the first one.

Figure 29 – The Hand Control Locators parented under the Sword Control NURBS Circle
PARENTING AND CONSTRAINING IN MAYA

- Do the same Locator align steps and align a locator to each hand control. Rename them something that makes sense to you. I'll call mine R- and L_Hand_Offset_Loc.
- Constrain each Hand Locator to their respective IK hand control and then delete the constraints.

Figure 30 – Hand Locators aligned to the Hand controls
PARENTING AND CONSTRAINING IN MAYA

• It wouldn't be a bad idea to color coordinate the Locator controls, as it could get confusing as to which one is which in their default green color

Figure 31 – The Right Hand Locator control changed to a Red color

• You can also increase the size by changing the LocalScale attributes in the Shape node of the Locator itself
PARENTING AND CONSTRAINING IN MAYA

• Create a null buffer node for each **Locator**
  ◦ The steps for creating a null buffer node are the exact same as for the Locator buffer: simply press Ctrl+G with nothing selected to create a null, then parent constrain that to the object with Maintain Offset unchecked, and delete the constraint.
  ◦ You may be wondering why we created a locator for each hand control and then added a null buffer for the Locator instead of just adding a buffer node to the hand control. The reason is twofold: 1) We don't want to mess up the hierarchy of the rig and more importantly 2) if you reference in a character rig, you cannot alter its hierarchy with group nodes. The in-between locator node is our work around for this. I also realize that I did not tell you to reference the rig in, but this was so you could open up the provided project files and follow along. It is always good practice to reference your rig, and keep these things in mind.
  ◦ I previously explained what a null node is, but then I went back and edited it so that a Locator was used instead. A null is simply a transform node without a shape, and, if it has nodes parented under it, it's known as a group node. We can use nulls/groups in tandem with Locators to get the best results (another “best of both worlds” if you will – the Locator is simply to have an object that you can select in the viewport)

• Open your **Outliner** and select the R_Hand_Attach Locator on the sword, then the null buffer node for the Right Hand Locator control, and Parent Constrain with Maintain Offset. Repeat for the left side.
PARENTING AND CONSTRAINTING IN MAYA

• What we should have here can by represented by a flow chart

• This seems like a heck of a setup just for a guy to wield a sword, but believe me when I say that it's worth the trouble.
  ◦ The sword locators can be used to adjust the position of the hand, by translating them up and down in local mode, or rotating them in the Y axis; when rotated the hand should pivot perfectly around the handle
  ◦ The hand locators can be used to further adjust the position of the hand if necessary
  ◦ Both sets of locators can be animated freely

• We'll still run into the issue of the Blend Node when trying to add animation to the hand controls. You can avoid this by having a selection set that doesn't include these hand controls when you key everything, or you can just key them before you get started, and make sure the Blend Attribute is set to 1 and not have to worry about it again.

• Optional Steps:
  ◦ You can add a Space Switch attribute to the sword_ctrl, and use the constraints for the hand locator buffers to switch with. This might be necessary if the character switches the sword from one hand to the other or takes one hand off, for example.
  ◦ Then its a matter of using the GetX/SetX shelf buttons we created earlier for our seamless space switch
  ◦ You can also add a group offset to the Sword Control, then constrain that to Normans Chest, Hips, or Global control (or all 3 if you're so inclined). From there you should know how to add a Space Switch attribute based on your current knowledge

• Check out 01b_PF_NormaWithSword_End.ma and compare your setup to mine to see if you did something differently or incorrectly. If so, go back and try re-doing the steps.

So that's our first and probably most common scenario of constraints within animation. Hopefully you understood the steps as you were doing them because of the previous explanations, instead of just following along for the sake of following along. It's really important to understand the concepts.

Let's move on to moving our character around in space.
This particular example might not be as common of a setup, but it’s good to know how to do either way, and it’s quite simple. Let’s run through a scenario: You have your character posed out in space, but you realized that he needs to be in a different position, and more importantly, rotated slightly, while still being in the same pose. For the sake of argument, let’s suppose you can’t move his world controller around, what do you do? If it was just the position that you needed to move you could grab his “mover controls”: the controls necessary to translate him in space and keep the same pose. But what if you need to rotate him as well, even just slightly? You can’t just rotate the mover controls, because they’ll each rotate in their own space. We need to fix it so that all the controls rotate from the same pivot point.

As we are using Norman, there are plenty of variables to think about. You need to first determine which controls are the mover controls. Depending on what space the controls are in, it could be tricky. You should be able to simply grab whichever ones are the mover controls, translate in space, and not see any “double transformations”. We can demonstrate this pretty easily:

- Open up a fresh scene with Norman imported and unhide his controls via the VIZ CTRLS icon
- Grab his root control and move him around (it’s not even necessary to pose him)
- By default the limbs are in FK, and the pose is maintained as you translate and rotate his root control. This is the effect we want.
- Now switch his limbs to IK mode
- Try moving the root around again. See what happens? The IK limbs stay pinned in space, as they should
- Now select each individual limb control and switch the space from World to Root changing their respective attributes from 1 to 0 and 0 to 1.

Figure 33 – The Space Switch attributes for most Norman controls
PARENTING AND CONSTRAINING IN MAYA

• The effect should be the same as what we demonstrated with the FK controls
• Now switch any limb(s) of your choice back to World Space (make sure only one of those attributes are set to 1 and the rest are set to 0)
  ◦ The idea here is that any control that is not in the space of another control (in World Space) should be considered a mover control. Mover controls are what need to be constrained in the next step.
• Create a locator and move it to where you want the mover controls to pivot from. A good place would probably be the root or somewhere in between where the feet are if they're planted
• Constrain the mover controls, one by one, to the newly created locator
  ◦ Assuming the movers have animation on them, a Blend Node will be created. Set the attributes to 1 accordingly.

Figure 34 – The Mover Locator setup in the pivot point we want it in
PARENTING AND CONSTRAINING IN MAYA

• Now as I see it, there are two possible sub-scenarios:
  ◦ You want to slightly adjust the position of a pose on a single frame without having to re-position and re-pose the character
  ◦ You actually want to use the locator to move and animate the character in space.
• Lets try the first scenario, as it's easier. Move the locator so that the character is in the desired position.
• At this point you'll have to go into the Outliner and manually delete the constraints, but not before doing the GetX/SetX shelf buttons. So, GetX, delete constraint, SetX.
  ◦ If you don't want to dig in the Outliner, select the driver object, then the driven object, and go to Constrain > Remove Target. That's basically the same thing, and the GetX/SetX steps are the same.
PARENTING AND CONSTRAINING IN MAYA

Admittedly, in the time it takes to do all that, you probably could have just re-posed the character in space. It really depends on the scenario you're dealing with. Give it a try either way so you have it in your toolbox for the future.

- The other scenario involves actually animating the character with the locator control
- Redo the constraint steps for the mover controls
- Animate the locator so it moves the character how you want (obviously for the sake of the example you could just move it randomly)
- Now select the mover controls and Bake them down

That's pretty much it. Again, this is not as common of a scenario that you might encounter, but it's still good to know, as you'll most likely need it at some point without even realizing it.

The results can be seen in **02b_PF_NormanMover_End_01.ma** and **02c_PF_NormanMover_End_02.ma** for each respective sub-scenario.

An alternative method could be to get the mlAutoIkFk script that comes with the Norman rig. You can select the IK controls, switch them to FK with the script, and then all you have to do is constrain the Root control to the Mover Locator, since the FK will follow automatically. You can then switch them back to IK and delete the constraint on the Root control. It's pretty convenient to have this script if you happen to be using Norman, but not all rigs come with an Ik/Fk switcher, so it's good to know the first method either way.
PARENTING AND CONSTRAINING IN MAYA

Now we'll take a look at creating what's called a reverse hierarchy. A standard hierarchy you would see in a character rig such as Norman, starts from the hips, works its way up the spine and to the neck and head. In other words, the hips control the spine, control the chest, control the neck, control the head, etc. What if we come across a situation where we have a character swinging from an object such as a bar or pole? I can assure you that it is – while possible – not ideal to use a standard hierarchy, as there will be a lot of counter-animating and counter-posing. The solution is to create our own hierarchy – the reverse hierarchy – on top of the existing hierarchy, that starts from the arms (for grabbing the pole), which control the chest, which control the spine, which control the hips, etc.

The idea is not to permanently make changes to the rig but rather create a temporary rig with locators and constraints. And that's precisely what we'll do.

Think of it another way: pretty much every animator is familiar with the pendulum exercise and knows the principle of overlap. Imagine if you had to control the entire pendulum, starting from the bottom, working in IK, and counter animating the entire thing. That is tantamount to doing this example without a reverse hierarchy; it's just a pain. We'll make it as natural as a real pendulum. With that we can assume that no matter the rig, the spine should be in FK for this to work.

We'll start with 05a_PF_NormanRevHi_Start.ma for this example. In the scene is Norman jumping towards a pole that he will swing from. We'll accomplish two things: Having locators control his hands, and space switching before and after he grabs the pole, and having those same locators be the start of the reverse hierarchy that controls the rest of his body.
PARENTING AND CONSTRAINING IN MAYA

• Create two locators that are aligned with the **pole_geo**, and roughly lined up to each hand
• Create a third Locator in the middle of the pole and parent the other two under it
• Create Locator controls for each hand, as well as offset groups for each Locator

Figure 36 – The pole Locator Controls
PARENTING AND CONSTRAINING IN MAYA

• Create a Locator control for each spine control and the hip control. Add the necessary constraints

![Locator Controls for Normans Body](image)

Figure 37 – The Locator Controls for Normans Body

• Each Locator should be Parented in the opposite direction of Normans Spine Hierarchy. Or in other words, the Locators should control the Locator below it – hence, “reverse hierarchy”

• Constrain the offset group for the Locator for each hand to the appropriate Pole Locator.

• Create an offset buffer node for the top chest control. Constrain this to the middle Pole Locator.

• You should be able to grab the middle Pole Locator and move everything now
  ◦ Ensure that the arm pole vector controls “Local World” attribute is set to 0, and the Head control is set to Chest Space

• Set a key on frame 1 on all the controls and locators. Ensure all Blend Attributes that are created are set to 1
PARENTING AND CONSTRAINING IN MAYA

- Move down about 4 frames, and make the hands grab the pole
- Set a key on all controls and locators on frame 4
- Move down to frame 24 and swing Norman under the pole with your new reverse hierarchy setup, especially taking advantage of the middle Pole Locator. This is the result I got:

![Figure 38 – Norman swinging through on the pole with the reverse hierarchy controls](image)

From here you could easily bake down the animation of the normal controls, as well as switch the space of hand locators so he's able to release from grabbing the pole and jump away, based on what you know already, of course. The finished animation in the project file 03b_PF_NormanRevHi_End.ma is nothing particularly exciting, but if it gets the idea across, that's all that really matters.
WOW! That is a lot to take in! Hopefully all or most of this made sense. I really, truly hope this is a helpful tutorial. I just see way too many animators struggle with constraints, or more simply, the basic structures of how Maya works. We've covered what I believe to be the most in depth tutorial on parenting and constraining in Maya ever. The written tutorial that you just read and hopefully followed along with has been in the works for awhile, so I cut out a few practical examples in the hopes that I could come back to them soon and update the material, in order to complete it sooner. As of now, the production of the supplementary video will resume as well. In the meantime, I'd love to get your feedback. You can reach me at adam.bradford.7@gmail.com. And please, please please spread this to your animation friends!

That's all folks!