Advanced Character Tools and Systems in Autodesk® 3ds Max®

Paul Neale – PEN Productions

DG2442-P

Autodesk® 3ds Max® software provides a comprehensive and flexible set of tools for developing character animation rigs and systems. This class will focus on taking it to the next level by exploring advanced features for developing character rigs and systems. We will use MAXScript to develop tool sets that help you create custom reusable character systems. The class will introduce you to new ways of thinking that enable character technical directors to provide character rigs with depth. Attendees should have prior knowledge of rigging in 3ds Max.

Learning Objectives

At the end of this class, you will be able to:

- Describe the tools and systems used in 3ds Max for character setup
- Develop character systems using MAXScript
- Construct expandable and customizable character systems
- Explain controller stacks and list dynamic methods of use

About the Speaker

Paul Neale has been internationally known in the 3D Animation industry for almost two decades. His extensive involvement as Senior Director of Research and Development and Art Director of 3D has encompassed areas in TV series, feature film, special effects and high-profile games. Paul specializes in character rigging and modeling as well as writing plug-ins and scripted tools for system, software and production needs. In addition to his industry experience, Paul has been an Ontario College Professor for fifteen years where he brings his knowledge, professionalism and passion of 3D to his students. In 2008, Paul received Autodesk Masters Award for Contributions to CG Artistry. Paul Neale has been a Presenter for multiple Siggraph Master Classes as well as a Master Class at GDC. He has represented Autodesk as a regular Guest Speaker at trade shows and special events. Paul Neale has trained numerous companies over the years including Walt Disney Studios, UBIsoft.

paul@penproductions.ca

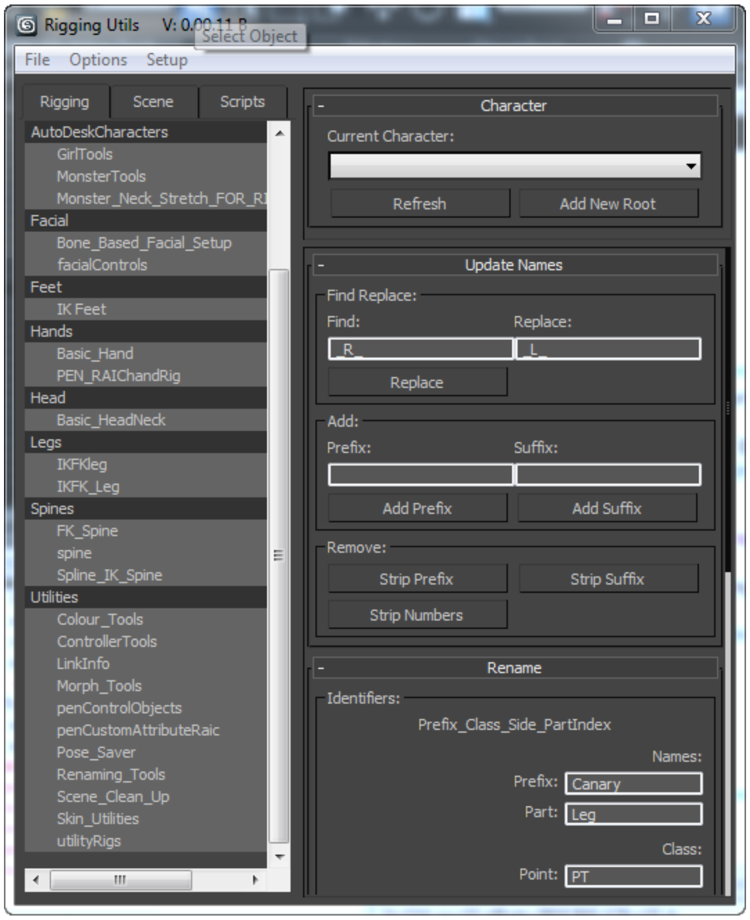
Describe the tools and systems used in 3ds Max for character setup

There are many tools and systems used for creating believable character rigs. Knowing which to use when and how can make all the difference. Games also don't allow many of them to be used because they would cause to much over head. Here is a short list of tools and technologies that are often used in character systems.

- Bones
  - Use for building hierarchies and skinning
- IK Solvers (Inverse Kinematics)
  - Use when limbs need to be locked down to other objects in the scene like the floor
- Spline IK Systems
  - Used often in spines when more complex motion is needed.
- FK (Forward Kinematics)
  - Preffered by animators because it creators natural arcs of motion.
- Control objects
  - Setup for animators to be able to manipulate the character rig.
- Skin
  - Use for binding meshes to bones and other nodes.
- Skin Morph
  - Use for correcting joints and other problem areas after skinning. Can also be used for mimicking muscles.
- Skin Wrap
  - Binds meshes to meshes. Often use for binding a shirt to a skinned body. Skin Wrap can be converted to skin.
- Muscle Bones
  - Usually refers to stretchy bones that are setup to mimic muscles under the surface of the mesh. Skin is used to bind to them.
- Skin Sliding
  - Real skin slides over the surface of the muscles and bones. This requires more advanced plugins to be created. It also has a lot of over head.
- Morpher
  - Often used in facial animation where the original mesh is copied and reshaped to create morph targets.
- Bone based facial rigging
  - Games usually use this method for creating facial shapes since moprhing is very expensive to compute during game play.
- Blended Morph based facial rigging

- ○ Refers to using morph targets that don't have specific facial shapes like a smile but instead use many muscle based shapes blended together to achieve more realistic movement.
- Max Script
    - ○ Max Script is the most important feature of all of them. Essentially Max Script is all the tools above and more since it can access them all and allow for repetitive operations to be preformed in a more productive way. Any good technical director needs to be fluent in at least the language that is native to their software to be effective in the production environment.

The list can go on and on but understanding the basics is the most important part. When used correctly the tools above will produce excellent results.



Find: _R_ Replace: _L_

Prefix: Canary

Part: Leg

Point: PT

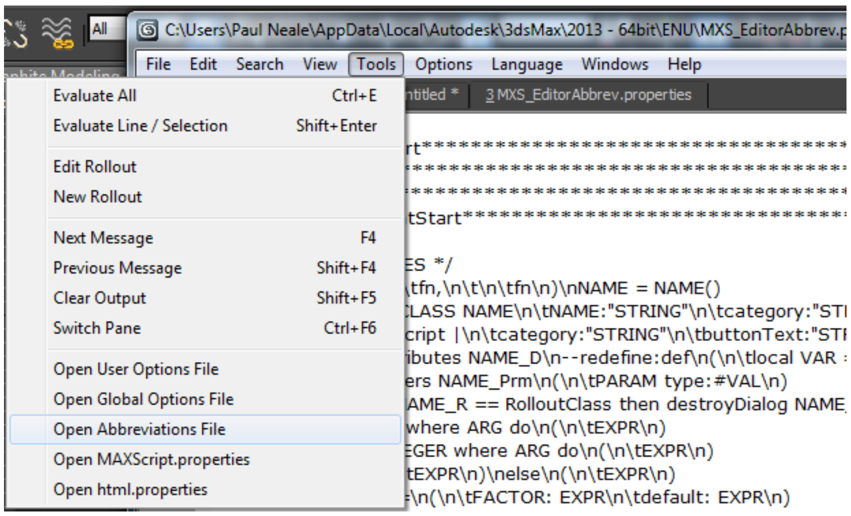
Develop character systems using MAX Script

A character system is a setup of tools and methods that allow technical directors and animators to build, manage, work with characters in a more efficient and effective manner. They can come in be built in an infinite number of ways and no one way is correct. Deciding if the system that you choose will fit your needs comes down to answering a few simple questions.

1. Does the system meet the needs of a current and future productions?
2. Will it be easy for others to understand?
3. Does it provide technical directors the flexibility to create more then one kind of character?
4. Does it provide animators with the flexibility that they need?
5. Is it expandable?

6. Is it portable to other applications?

One of the first steps of writing cleaner and more readable code in any language is to define a set of parameters for syntax and naming conventions as well as comments. To help with this in Max Script it is best to use the Abbreviations file to create presets for Headers, Comments, Functions and other often needed code snippets. This will ensure that you and others follow the same code specifications on this and other projects. It also greatly speeds up the writing of code.



```
rt*************************************
***************************************
***************************************
tStart*********************************
ES */
\tfn,\n\t\n\tfn\n)\nNAME = NAME()
CLASS NAME\n\tNAME:"STRING"\n\tcategory:"STI
cript |\n\tcategory:"STRING"\n\tbuttonText:"STI
ibutes NAME_D\n--redefine:def\n(\n\tlocal VAR :
ers NAME_Prm\n(\n\tPARAM type:#VAL\n)
AME_R == RolloutClass then destroyDialog NAME
where ARG do\n(\n\tEXPR\n)
EGER where ARG do\n(\n\tEXPR\n)
tEXPR\n)\nelse\n(\n\tEXPR\n)
=\n(\n\tFACTOR: EXPR\n\tdefault: EXPR\n)
```

To create an abbreviation select the Tools / Open Abbreviations File option in the Max Script Editor and on single lines add the name of the abbreviation with an = sign after it and then type out the code. It must be all on one line.

Here is the abbreviation for the comments that I use in all my code.

[code]

comment=/*CommentStart************************\n|\n*************************CommentEnd*/**

[/code]

When the word **comment** is typed in the Max Script editor and then the user presses Shift+CTRL+A the work comment will be replaced with the line associated with it. The abbreviation above will look like this when it is entered.

[code]

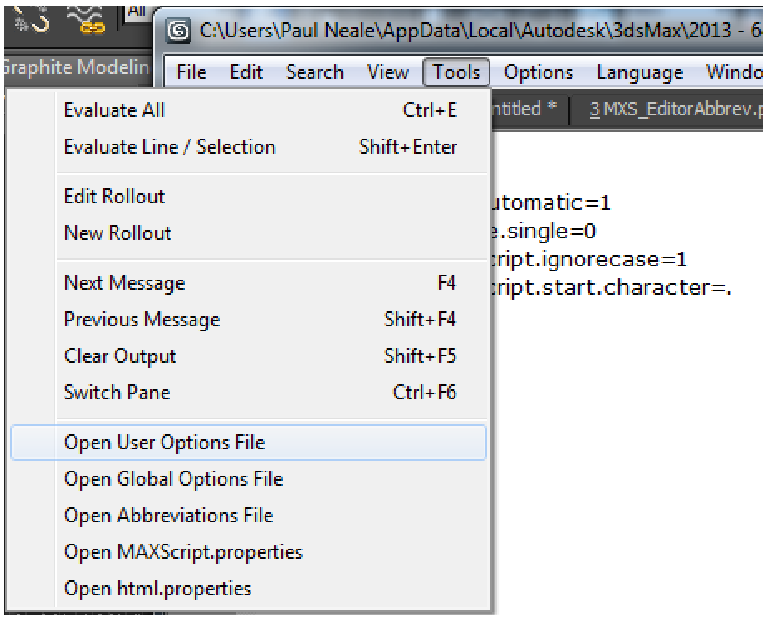
/*CommentStart**************************

[Cursor will be flashing here]

*************************CommentEnd*/

[/code]

\n inserts a new line as well all other standard escape sequences will work. The **|** can be added any where in the line and this is where the cursor will be placed.



Another useful feature that can reduce the amount of time that it takes to write code and also ensure that less mistakes are made is code completion.

Open the Tools / Open User Options File from the Max Script Editor and add the lines below.

[code]

buffers=20

tabbar.multiline=5

autocompleteword.automatic=1

autocomplete.choose.single=0

autocomplete.MAXScript.ignorecase=1

autocomplete.MAXScript.start.character=.

[/code]

Here is what each line achieves **:**

buffers=20 – The number of tabs that are allowed.

tabbar.multiline=5 – Allows for multiple lines of tabs in the editor.

autocompleteword.automatic=1 – If this setting is 1 then when typing a word, if only one word in the document starts with that string then an auto completion list is displayed with that word so it can be chosen by pressing Tab.

autocomplete.choose.single=0 – When set to 1 and an auto-completion list is invoked and there is only one element in that list then that element is automatically chosen.

autocomplete.MAXScript.ignorecase=1 – When set to 1 the API file is searched in a case insensitive way to find the function which will have its signature displayed as a calltip.

autocomplete.MAXScript.start.character=. – Allows you to specify characters which start, end and separate parameters.

Creating consistent and readable variable names is also a great way to make sure that code is working and readable. I personally use Suffixes but many use a prefix to denote different types of variables and what the variable might contain. Here is a short list of what I'm currently using out of habit more then anything else.

Possible variable names:

myObjectsAr – Ar describes an array

myButtonBt – Bt is a button

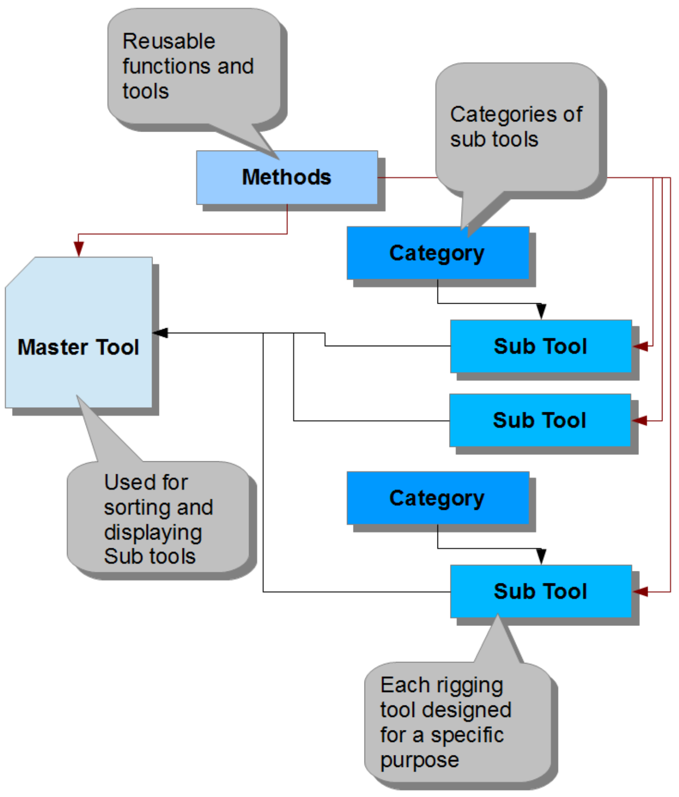
mYInt – Int is an integer value

myTm – Tm denotes a transform matrix value.

The above is an illustration of suffixes that I use. Make sure that the variable name is also readable and describes what it is for. For instance, **collectedObjectDataAr**, For any one else reading the code names like this can make it far easier to follow. When you return to your own code after a year it will also help you follow what you did.

Construct expandable and customizable character systems

Building expandable systems isn't as hard as it sounds as long as some thought is put into the structure of the tools. For the system that I use at PEN Production I have a master tool that is designed to sort and display all the sub tools and also scripts for storing common functions.

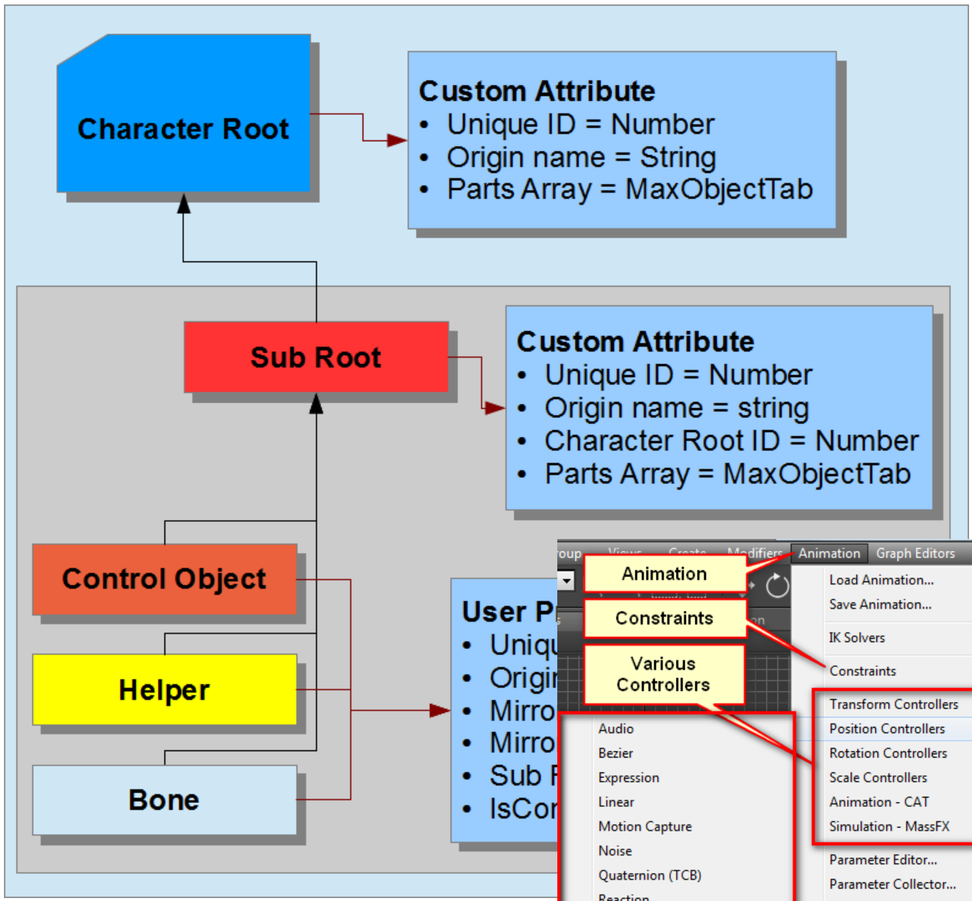

Reusable functions and tools
Categories of sub tools
Methods
Category
Master Tool
Sub Tool
Sub Tool
Used for sorting and displaying Sub tools
Category
Sub Tool
Each rigging tool designed for a specific purpose

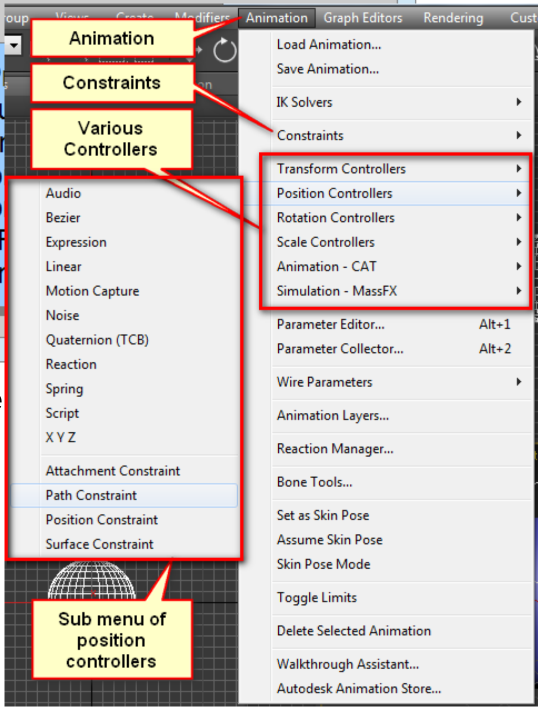
The break down will look something like the image to the left. A master tool is created that reads a preset directory and returns all the scripts (Sub Tools) found in it. Each script with in will have one main function called getRollouts that returns an array of rollouts used for that tool. Those rollouts are then displayed in the Master Tools panel ready for use. One or more Methods scripts are run when the Master Tool or Sub Tools run and contains all the generic and reusable functions. Some of these functions can be for building bone hierarchies, calculating the mirroring of objects or what ever is needed.

Sub Tools

Each Sub Tool is responsible for creating a part of a rig or a tool used for working with the rigs. If the tool is used for building a rig part it needs to be aware of what parts of a rig have already been built in the scene using that tool. This way it can also delete and rebuild parts of that rig if there are changes that are needed. There are several ways to keep track of parts of rigs, one is to use specific naming conventions that can be parsed to know what is an arm and what is a leg. This can be problematic at best as names can be changed easily and naming conflicts can happen. There are three other ways of identifying parts of rigs and that is by using User Properties, App Data or Custom Attribute definitions.

For the systems used at PEN Productions a mix of User Properties and Custom Attributes are





used to help organize and store information about the rig for later use in animation tools and the rigging tools.

Custom Attributes are added to the Characters Root node as well as the Rig Part Node that contain several properties. Unique ID's are generated using genClassId returnValue:true and that value is stored as a two part array and can be searched for in the scene if needed. Origin Name is the base name given to the part before the name of the character is prefixed to the beginning of the objects name. If the name is ever changed on purpose or by mistake it is easy to return it to the original as it is store with the node. This name can also be used for transfer animation from one character to another by temporarily swapping the object name with the origin name before saving and loading animation. It can then be swapped back to return the name to how it was set. Parts Array is a MaxObjectTab type value and stores weak references to all the Rig Parts in the rig. I will explain weak references later in the document.

The Sub Root is a root node for part of the character rig that was built by one of the sub tools. An example might be the right arm. Once again a custom attribute definition is added that contains similar properties as the Character Roots definition. The only real difference is the inclusion of a Character Root ID, this allows you to know which character root the Sub Root belongs to. The Parts Array parameter in this case stores an array of weak references to all the parts of the Sub Root.

Each object in the Sub Part, for instance the right arm, then use User Properties to store information about the part as well as other parts in the rig. The object that is it's mirror counter part can be listed as well as a pivot object to mirror across and a flag that can be set to identify if the object is a control object or not. Far more data can be represented this way and this is just a small sub set of what I'm currently using at PEN.

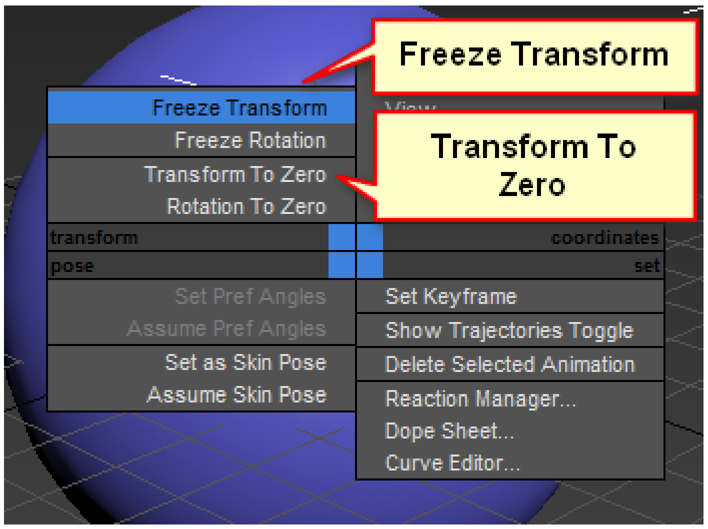
User Properties are a good place to store data but note that it is all stored as a string and the values need to be converted to strings for storing and from string to what ever their class is when reading. Another location that this sort of data can be stored is in the App Data channels. App Data is invisible to the end user so it has the advantage that it can be hidden. The disadvantage is that it is harder to manipulate manually. User Properties are quick and easy to access manually via the Object Properties dialog so change can be made to them without using any script. Also there are management functions already built in instead of having to write your own.

Explain controller stacks and list dynamic methods of use

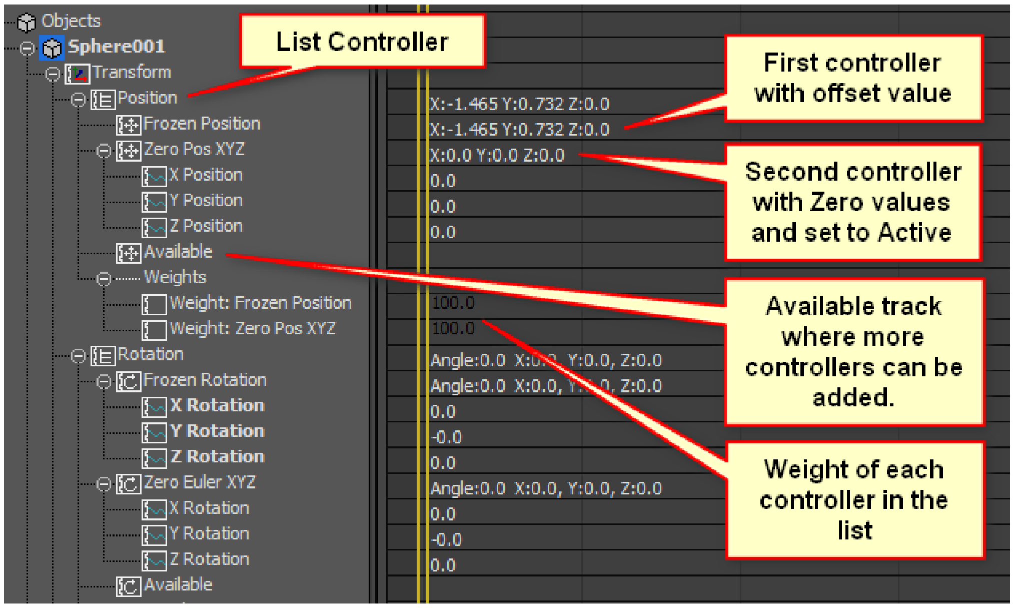
Controller stacks or one of the most underused areas of Max that I see and should be used for all the power that they can provide.

The way they are mostly used is when a controller is added via the Animation menu in the Main Menu Bar.

When controllers are added in this way a list controllers is automatically created and the new controllers is added to the list. The Animation menu only works for transform controllers and can't be applied to any controller stack on other parameters, for instance the Angle of a Bend modifier.



Another way that list controllers are added is via the Freeze Transform dialog that is found when you ALT+ Right Click. The Freeze Transforms works by creating a list controller on the Position and Rotation tracks of the object. The way that Freeze Transform works is based on how list controllers work. When the list is added to the position for instance the first controller in the list stores the current offset value from the parent object of the node or the scene root if there is no parent. The second controller is applied and has a value of [0,0,0] as it is additive to the first. Then the second controller is set as Active. The active controller is the one that gets the value change when a user selects and moves the object. All

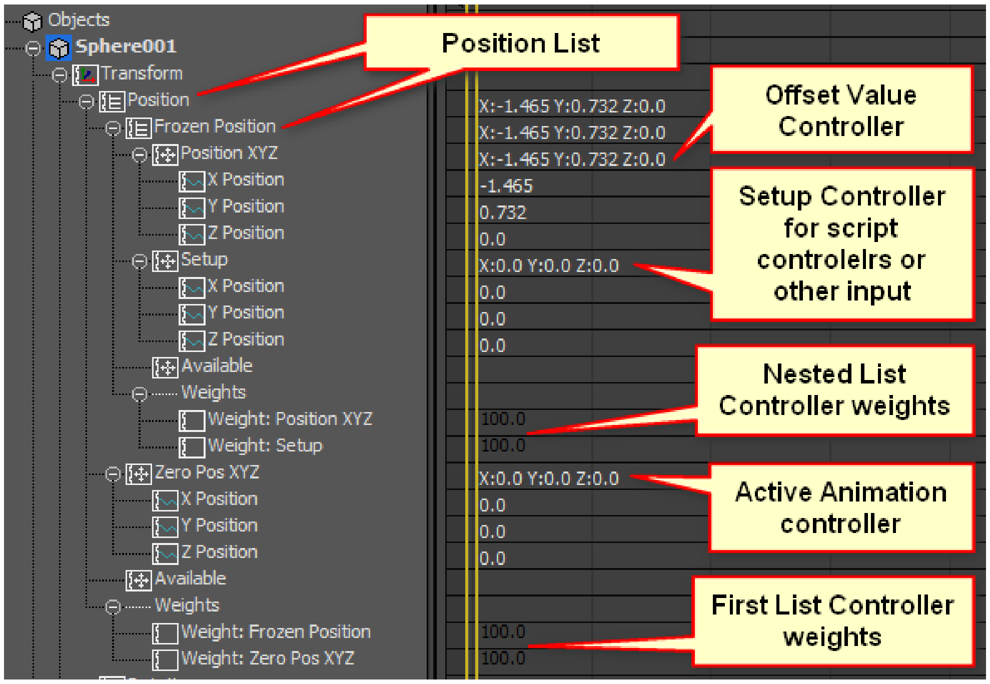

List Controller
First controller with offset value
Second controller with Zero values and set to Active
Available track where more controllers can be added.
Weight of each controller in the list

```
Objects
Sphere001
  Transform
    Position
      Frozen Position          X:-1.465 Y:0.732 Z:0.0
      Zero Pos XYZ             X:-1.465 Y:0.732 Z:0.0
        X Position             X:0.0 Y:0.0 Z:0.0
        Y Position             0.0
        Z Position             0.0
      Available                0.0
      Weights
        Weight: Frozen Position   100.0
        Weight: Zero Pos XYZ      100.0
    Rotation                   Angle:0.0  X:0.0, Y:0.0, Z:0.0
      Frozen Rotation          Angle:0.0  X:0.0, Y:0.0, Z:0.0
        X Rotation             0.0
        Y Rotation             -0.0
        Z Rotation             0.0
      Zero Euler XYZ           Angle:0.0  X:0.0, Y:0.0, Z:0.0
        X Rotation             0.0
        Y Rotation             -0.0
        Z Rotation             0.0
      Available
```

other controllers in the list still can input and change the objects position but they don't get input directly from the user moving the object.

The Available track is where more controllers can be added to the list. List controllers can also be nested one inside another and can make for a very powerful setup.

At PEN I use a nested list controller in the first controller of the main list. This is where the offset value is stored for the object. The idea is that within that list new controllers can be added that will control the position or rotation of the object that are not needed by the animator. It helps clean up the track view as it doesn't need to be opened and seen by the animator.



Position List

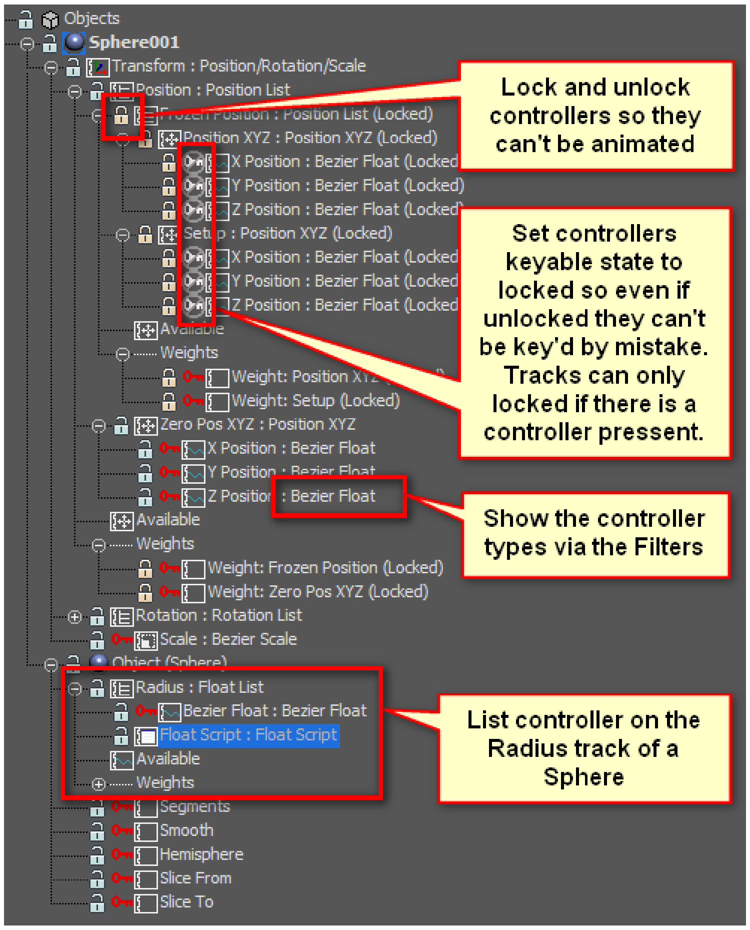
Offset Value Controller

Setup Controller for script controlelrs or other input

Nested List Controller weights

Active Animation controller

First List Controller weights


Objects
Sphere001
Transform : Position/Rotation/Scale
Position : Position List
Frozen Position : Position List (Locked)
Position XYZ : Position XYZ (Locked)
X Position : Bezier Float (Locked)
Y Position : Bezier Float (Locked)
Z Position : Bezier Float (Locked)
Setup : Position XYZ (Locked)
X Position : Bezier Float (Locked)
Y Position : Bezier Float (Locked)
Z Position : Bezier Float (Locked)
Available
Weights
Weight: Position XYZ
Weight: Setup (Locked)
Zero Pos XYZ : Position XYZ
X Position : Bezier Float
Y Position : Bezier Float
Z Position : Bezier Float
Available
Weights
Weight: Frozen Position (Locked)
Weight: Zero Pos XYZ (Locked)
Rotation : Rotation List
Scale : Bezier Scale
Object (Sphere)
Radius : Float List
Bezier Float : Bezier Float
Float Script : Float Script
Available
Weights
Segments
Smooth
Hemisphere
Slice From
Slice To

Lock and unlock controllers so they can't be animated

Set controllers keyable state to locked so even if unlocked they can't be key'd by mistake. Tracks can only locked if there is a controller pressent.

Show the controller types via the Filters

List controller on the Radius track of a Sphere

The Track View / Dope Sheet also allows for other options that can help with locking out artists from trying to make changes to the characters setup .

Locks can be setup on tracks so that controllers can't be changed. Any controller that already exists will still have input into the list controllers values.

A Keyable state can also locked and will prevent any new keys from being applied to the tracks. This can be useful if animators are manually adding keys in the dope sheet or curve editors and accidentally click on the wrong track.