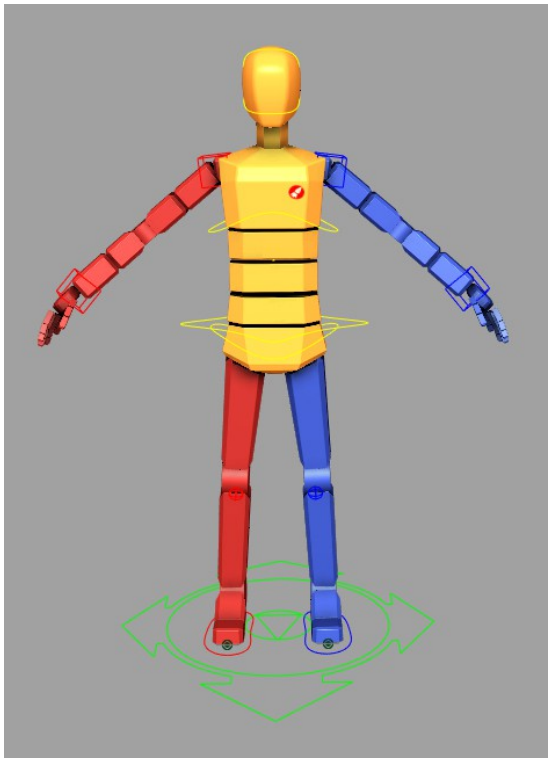
Welcome to our Studio!

Red9 Puppet Rig Overview:

Demo systems : March 2017
Red9Consultancy.com
info@red9consultancy.com



Base Rig System:

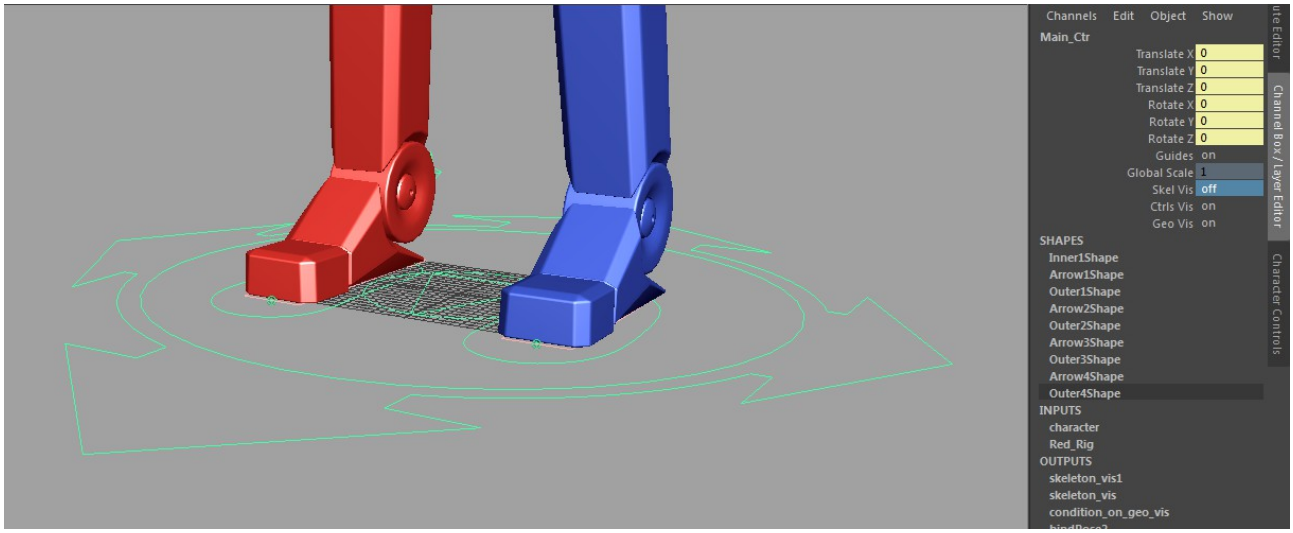
The Red9 Puppet Rig is a bespoke, procedural rigging system, designed from the ground up to be fast, robust and feature rich. The build system is designed not just for bipedal characters, it also natively supports quadrupeds and any other bespoke character requirements in it modular design. Being the internal Red9 puppet it seamlessly integrates to all the Red9 StudioTools, giving you instant access not just to a rig system, but to an entire open source animation pipeline. What's more the pipeline is mature and fully production tested in many high profile game and film studios around the world.

Internally all our rigs are built to the same scale (unless requested otherwise), this allows you to take animation from a character that's 8ft tall and simply load it onto one who's 2ft tall and the data will still map correctly. Scale is bound to the 'MASTER_NODE' as an attribute rather than directly being exposed to animators.

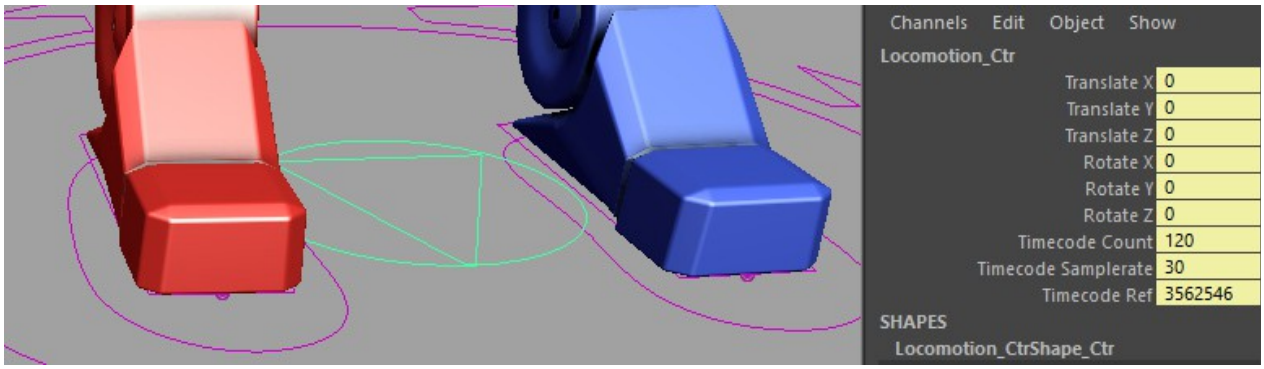
At the root of the Red9 rig is our internal MetaData system, exposed through the Red9 StudioPack API. This powerful base architecture allows you to interface with the rig via a simple Python object that exposes most of the management and animation functions directly. Things like pose handlers, animation library, mirror functions, pose compare and many other vital functions are all wrapped for you.

Main_Ctrl:

The Main controller is the root of the hierarchy and the node designed to move the entire rig, unaffected in world space. It's also the controller that controls the base visibility of many of the systems such as switching the skeleton visibility, turning off the controllers and selecting the geo / lod to display.



Locomotion Control:

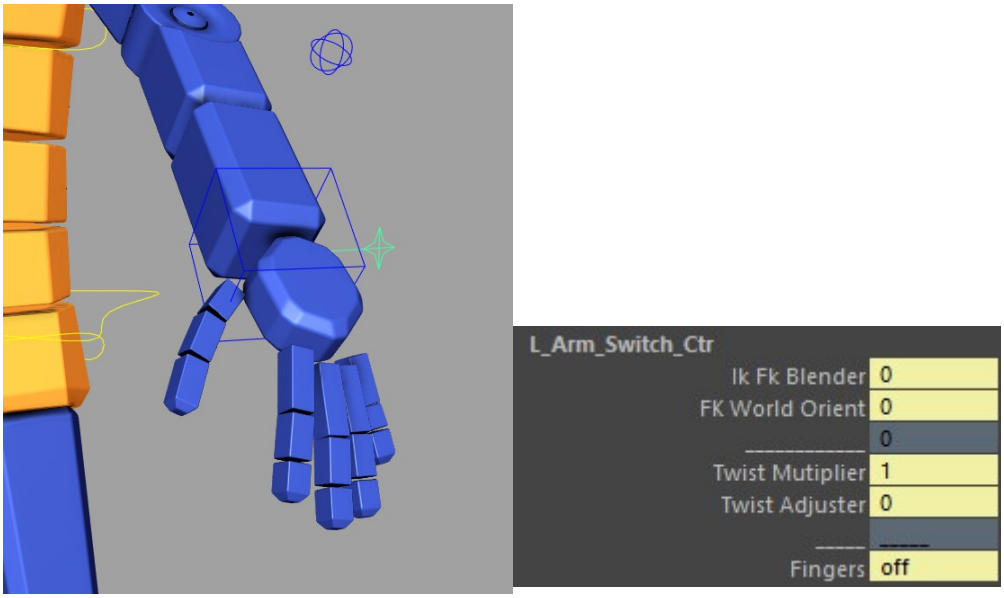


The Locomotion Controller is designed to be used to represent the general motion of the character, ideal for games where this often becomes the characters root animation for AI etc.

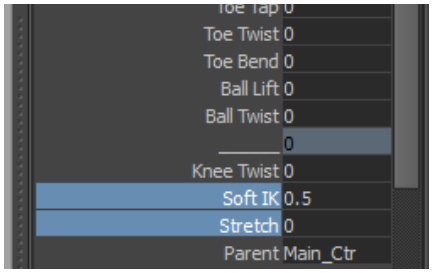
This is also the node that takes our “Timecode” system data when exposed, used to track SMPTE timecode from a moCap shoot right the way through production, allowing the syncing of data inside of Maya with our proprietary tools. When used in conjunction with our audio tools this also allows us to sync audio tracks in the Bwav format, to the motion data of the rig.

Generic IK/FK Limbs, SoftIK and Stretch system:

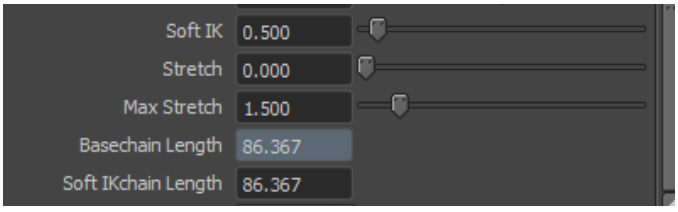
All core systems support full IK FK blending which is controlled by the pin stars at the back of the Wrist, Feet and Head controllers, it's also on the COG_Ctr for the spine solution. This switch also controls the orientation of the FK chains root node and whether it follows it's parent nodes rotations or world space. For example when the FK_World_orient is set to zero on the legs, the FK chains follow the orientation of the hips, if this is set to 1 the they follow in world space and the hip rotates do not propagate to the FK chains.



The arm and leg systems also support a SoftIK system with damped stretch. This is a massive benefit especially when binding to moCap data as it prevents that slightly 'stooped' look where people tend to back-off from fully extending limbs to avoid IK pop as the limb straightens. The system works by moving the ikHandle in the opposite direction of the controller once the softIK distance is reached. The motion is exponential so comes in smoothly, causing a damped effect.



The stretch system works alongside the softIK, running the same math to determine at which point the stretch begins. The higher the softIK value, the sooner the stretch begins. Unlike a lot of stretch systems this means that the joints start stretching before they lock out.

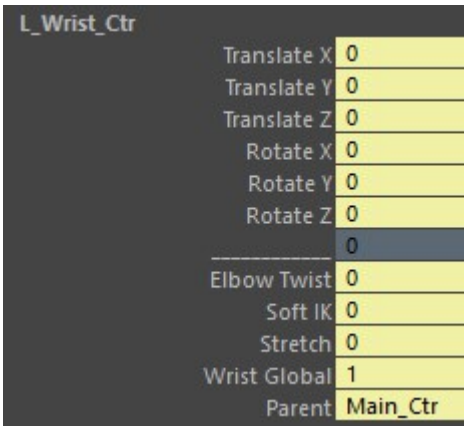


On the controllers as hidden attributes we also have some additional data for the system:

- Max stretch is a clamp on the amount of stretch possible.
- Basechain Length is the original length of the joint chain.
- SoftIK chain length is the length the soft system uses as the chain length. If this is reduced then the limb will stay slightly bent and never reach full lock out.

Arms Setups:

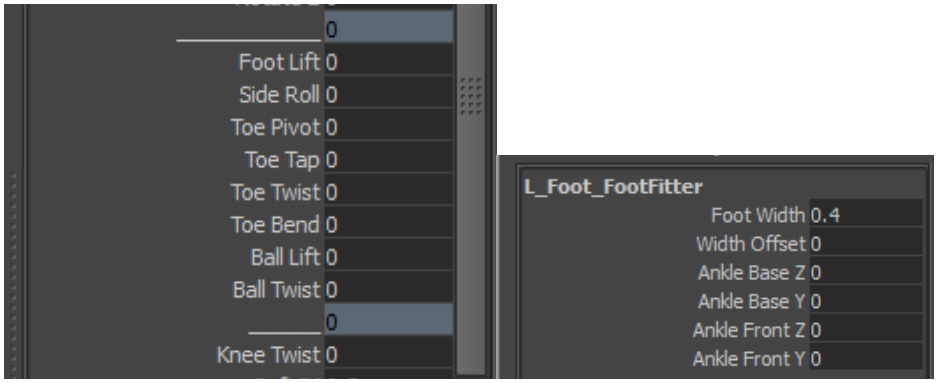
The Wrists have a 'global mode' where the wrist translates come from the IK solve, but the base rotates come from the forearm itself. There's a paddle controller that becomes available to control the rotate over and above that of the arm. Image waving and just controlling that with the IK controllers translate and not having to worry about the rotates tracking.



The Arm includes twisters for upper, lower and wrist deformation, but these are optional. There are attrs on the Switch Controller pins for controlling how much drive these get, and for manually over-driving them where needed. These are on the Switches so that both FK and IK as influenced by the results.

The Clavicle includes an 'Auto-Clavicle' attr that calculates the clavicle motion automatically based on the arm rotations. This is one of those things some animators love, some hate, but in terms of it's solve it's pretty accurate. There are various ramps controlling how this reacts that are exposed if needed. The clavicle also has the same twist attributes as the pins, these control the should twist systems.

Fully configurable Inverse Foot System:



The foot controller has all the usual channels for managing ball_lift, toe_tap etc, but what makes this system more powerful is the underlying structures.

The feet have a system we call “Foot-Fitter” on them, when you select the foot you'll see a pink rectangle underneath, this is actually what we use to set the width, ball placement and offsets to match the mesh geo. This means that ball roll / side lift data is consistent even if one character is wearing snow shoes, the data will still transfer over correctly. It's set as un-selectable but is just a child of the foot control, grab it and have a play, you'll soon see the power of this for transferring data.

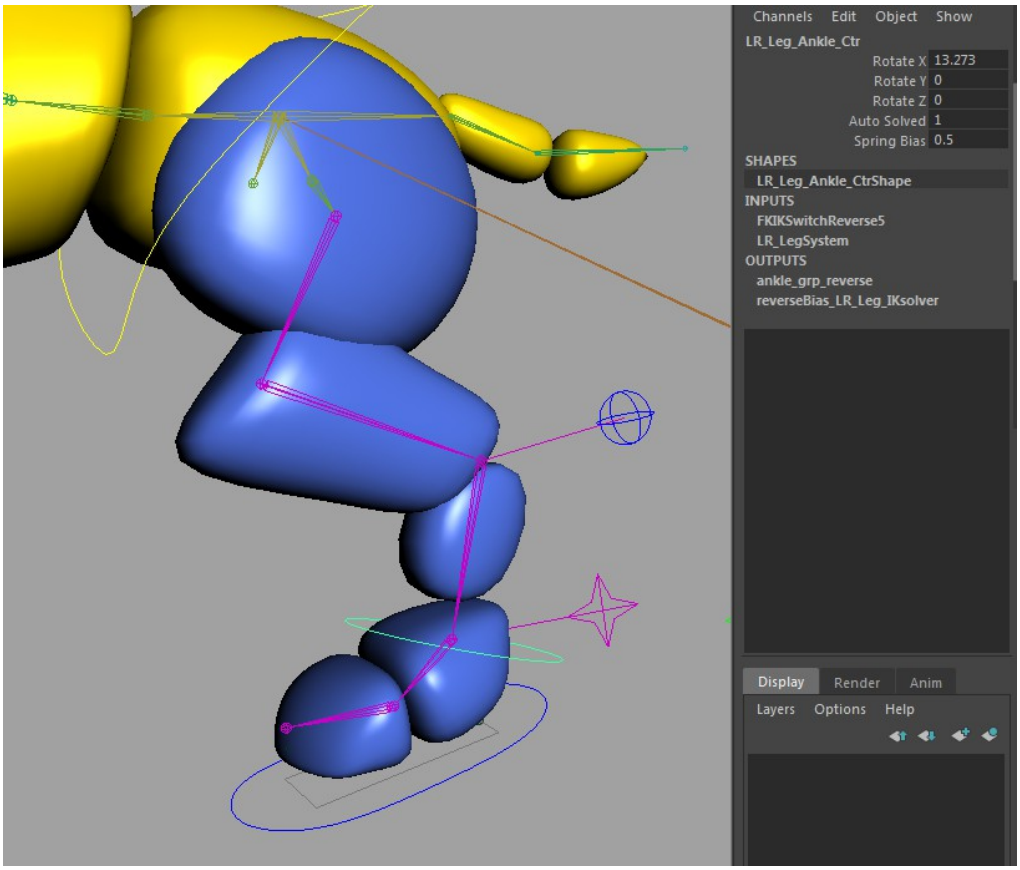
Auto-Solved Pole Controllers:

This is a new feature in the rig and is default from v1.06 build onwards. On the Pole Controllers you'll find the attribute “Auto-Solved” which does just that, blends on a tracking system that tries to resolve the best position for the pole vector based on the limb's controllers.

It's actually remarkably accurate and we've run this in preference to solved MoCap data in the past for certain clients as the results were preferable. It's also designed not to flip or pop!

Quadruped Leg solver:

For Quadrupeds the leg system builds a hybrid spring based setup with the capability of solving the 3 joints automatically. However unlike standard spring based setups this also allows you full control of the limb via an additional ankle controller.

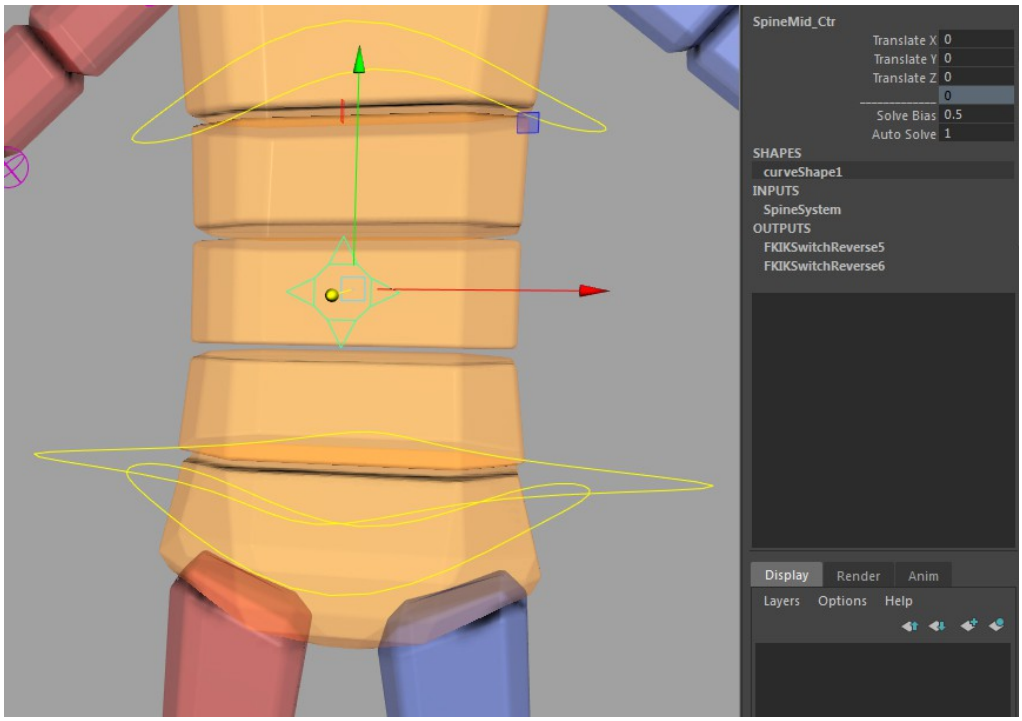


The Ankle_Ctr controls how the tension in the limb is distributed, allowing you to shift that tension up or down the limb via the “Spring Bias” control. If “Auto-solved” is off then the limb becomes a standard 2 joint solver with the ankle bound to controller.

However the core feature is that with the limb auto-solving, and the bias / tension set, you can still rotate this controller to fine tweak the angle of the ankle joint. This mixed with the Auto-solved pole trackers makes for an incredibly solid and intuitive limb system, and one that's had a lot of development work on live projects recently.

Spine Systems:

The spine systems has fully blendable IK / FK switching which is controlled by the “Ik fk blender” attribute on the COG control.



The spine solve also includes a stretch system but currently this is purely a linear stretch, the plan is to integrate the softIK stretch to this for a more realistic 'tensioned' deformation in the future.

In the centre of the spine there's a small star, this is the “SpineMidCtrl” controller that effects the middle of the IKspline itself, allowing you to fine tweak the solve. This changes how the tension along the spine is handled, whether it more closely follows the hips or the chest. The default for this is 0.5 which evenly distributes it 50/50 along the spine. This tracking behaviour runs by default but can be turned off with the “Auto-Solve” attribute on this star.

Head System

The head system built differs depending on the number of neck joints passed into the build and whether the neck is deemed to be vertical (biped) or horizontal (quadruped).

The default setup has both IK and FK systems, managed by switch pin controllers at the back of the head like the arms and legs. The default IK setup for the head manages the neck automatically depending on what the head control is doing. If you have 2 or more neck joints then the solution is effectively the same as the spline with a midNeck controller that manages how the centre of the solve is controlled.

The head also has “Head_Aim” and “Pin_Root” channels, the pin locks the rotates of the head to the world, allowing the chest to rotate whilst the head remains at the same angle. Head_Aim switches on another controller in root space that's used as the aim target / look-at for the head control.

Fingers:

The finger controls are hidden by default, their visibility is controlled by the Switch Pin controller as per the ik/fk blending etc. By default the fingers are simple fk controllers but we're working on expanding that to integrate a poseMixer on channels from a new controller. The Fingers also have their own poseHandler so that poses can be stored and loaded from either side.

Additional Patches available:

The Red9 Puppet Rig has 2 stages in it's creation before it's passed over to clients, a build and a publish cycle. Within the Publisher there are a number of additional patches that are available on request, some of which may become default features in future. These patches are tracked on the Rig's mNode for reference.

The Publisher system also allows clients to continue to update the core of the character, modifying skinning and adding additional controllers for deformation after we've delivered a rig. We simply take updated source files from the client and integrate them at the publisher stage to the rig.

patch_add_fkik_matchpoints:
This adds in additional nodes in the rig used by the IK>FK matching solution in the ProPack and is run by default. However it also allows for custom rigs to benefit from the codebase.

patch_pole_tracker:
This is a new system where the pole controllers move relative to the system they control, automatically tracking and solving the position of the pole vector controller for you. This is still in development but already proving very popular and table. This is activated by an attribute on the pole controllers themselves “autosolved” which blends between the basic pole behaviour and the autosolved tracking setups.

patch_unlock_finger_root_translates:
By default we only expose the rotates of the finger controller, this patch unlocks and exposes the root jnts of each fingers transform channels and re-hooks the rig accordingly.

patch_finger_poses:
This is an incredibly powerful addition to the setups. It allows clients to produce their own finger pose library for core poses and have those consumed by the rig at publish time. This turns the poses into attributes on the Arm's switch controllers allowing the attrs to drive the hands to each pose with simple attributes. The true benefit is that should the client need a master hand pose changing AFTER they've already animated a motion set then they can just redeliver a new pose, we re-publish the rig with that update, and all animations that drive that pose will get updated automatically.

patch_toe_poses:
As the finger poses but for toes ;)

Red9 Animation Binder:

Along with our client rig we usually ship an AnimationBinder file (*my_filename*_**BND**), designed to accept animation data from any source and ideally suited to transfer MoCap data to the systems.

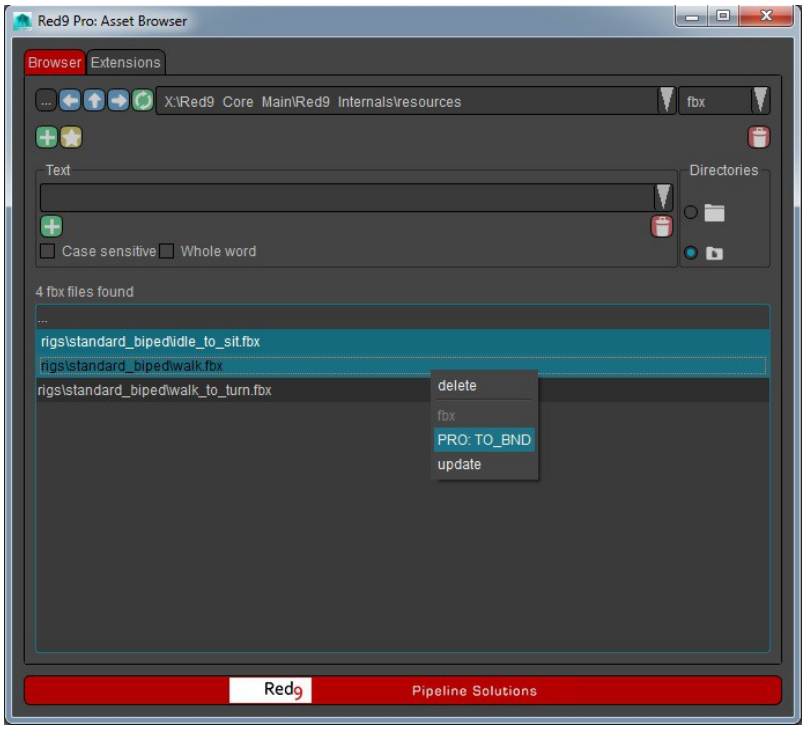
The binder runs HumanIK under the hood as an initial layer of character remapping, but we also expose custom manipulators for each control to allow you to modify and tweak the character mapping on the fly. This is a very powerful and fast way to do initial corrections to moCap data before you commit and bake the data down to the rig controllers.

If your source data does not contain a valid HIK definition the binder will still work, in fact the set-up is a lot easier just not as flexible this way.

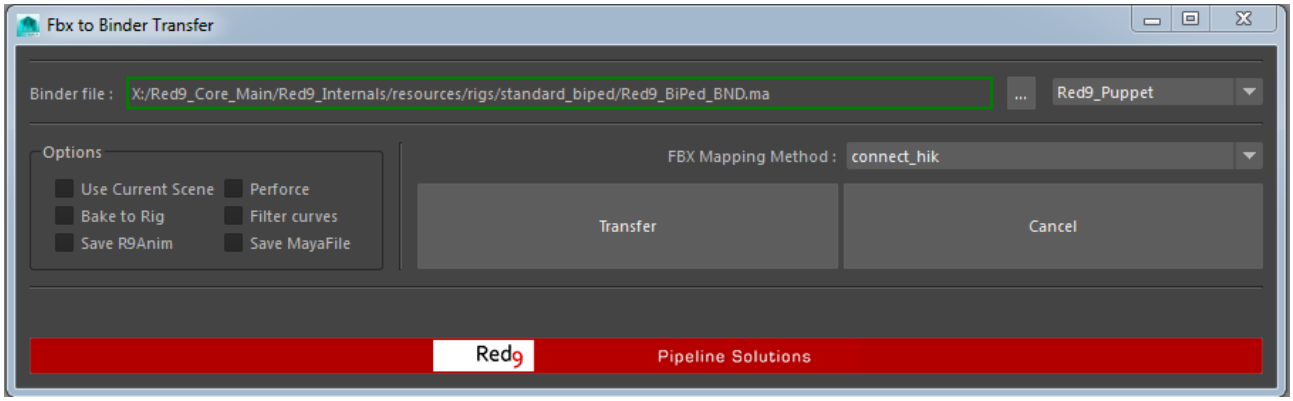
Connecting MoCap data to the Binder via the Browser function:

This is a new feature that's been in production for a while now and we've finally exposed it in the ProPack Browser for clients to use. This is something that we expand upon and modify on a client basis depending on your requirements, maybe hooking directly into a game exporter or perforce logic.

Open the Browser , point it to a folder containing fbx data and select one, or multiple fbx files. Now RMB click to open the options menu.



Open the **PRO: TO_BND** window and point the UI to the Binder file supplied with your Red9 Rig, this will be called *my_filename*_**BND**



FBX Mapping Method:

This controls how the fbx data is mapped onto the Binder's source skeleton. There are currently 3 options for this, all of which control how the fbx is dealt with by the animation transfer codebase.

- **update_anim**: this is the simplest method and relies on the incoming fbx's skeleton being identical to the binders source. This method simply uses the fbx update call to push the animation onto the source skeleton to then drive the binder. This would be the default for moCap deliveries.
- **reference_copy**: uses an internal method to transfer the fbx animation to the source skeleton but this method by-passes fbx's inability to deal with namespaces!
- **connect_hik**: this is the most powerful method as it uses the power of HIK under the hood to connect the incoming fbx's data to your binder, allowing you to accept moCap from ANY hik enabled source data directly to the Red9 Puppet!

Available Binders:

The binder files you see in the option menu are extracted from the clients project file and as such are dynamic to each client, allowing you to setup and expose all available rigs to the animators without them having to dig around into the depths of your source control structures.

Checkbox Options:

- **Use current scene**: assumes that the binder file is already loaded in the Maya session, default is off
- **Perforce**: If Perforce was found and your client project file has perforce set then this adds all files generated to your default P4 changelist
- **Bake to Rig**: this will bake the transferred data directly to the rig, if you're planning on editing the motion on the binder itself, which we highly recommend, then tick this off.
- **FilterCurves**: only run alongside the bake call, this deletes static animation curves then runs a Euler filter on the remaining data.
- **Save R9Anim**: this exports an r9Anim file alongside the Maya file.
- **Save MayaFile**: saves the resulting data to a Maya file

With the added power of the Binder systems you have not only a feature rich, fast and production tested character rig, but also a full animation transfer and mapping system integrated with it!