

Continuing Character Setup in Softimage|3D

To be transformed into a convincing animated character, a model must be prepared for all the kinds of motion you intend to create. This preparation—basically everything between modeling and animating—is called character setup.

In the first part of this series on character setup in Softimage, we looked at constructing a character's skeleton (see "Character Setup in Softimage," May 1999, p. 49). A web-based supplement discussed attaching the character to the skeleton using Softimage's enveloping process (visit "Web Only Content" at www.3d-design.com). If you've followed along this far, your character should be able to bend and move with its skeleton. In this installment, we'll make a set of animation controls so the character will be easy to manipulate.

As I mentioned in part one, lack of foresight during the setup phase will mean a limited performance during animation. Character setup should allow for a complete range of movements, poses, and emotions. Your ability to pose and animate a character quickly will be aided greatly through the use of constraints.

A constraint is simply a way to attach or relate one object to another object. Constraints free you from the traditional limitations of the hierarchy. Think of a simple skeleton, designed in a linear fashion from the torso out to the limbs. When you move or rotate a parent in the hierarchy, all the chil-



dren follow along. But what if you wanted the hands or feet to stay put or the head to sit still while the shoulders rotate? That's where constraints come in.

Softimage|3D certainly isn't the only program that allows for setups using constraints and expressions (a mathematical relationship between two objects similar to a constraint), but it has been the standard in character setup for years. Despite its slow devel-

Now that your Softimage|3D character has developed some backbone, the setup process continues with a look at controls, constraints, and expressions.

ANIMATORS ANONYMOUS

opment and the long wait for Sumatra (the next generation of Softimage|3D), it remains one of the strongest 3D programs available for creating animated characters. Softimage|3D allows you to take a freeform approach to character setups; it's an open workflow that doesn't feel like an add-on or a plug-in. Softimage|3D's toolset lets you build a working virtual machine for manipulating your character.

To demonstrate the power of constraints, we'll continue using Stuart, the star of part one, who was designed originally for an in-house project at my company, Cineframe Animation.

Without using constraints, you could certainly pose and keyframe the end effectors themselves. However, there's a good reason to use constraints instead. A keyframed end effector can't have a parent relationship—its position can relate only to global coordinates. If you were to need to move a character globally after animating it, any end effectors you had keyframed would drag behind, pointing toward the last keyframed position. You can avoid this problem by constraining end effectors to other control objects, which, unlike end effectors, can have parent relationships.

Using constraints, you can create a hierarchy of control objects separate from the skeleton that have their parent-child relationships. For example, some of these relationships might rotate the hips while the torso stays still, while others might rotate the hips and the torso together. The skeleton follows the constraint objects, which in turn drive deformations and movements in the body. This type of character setup is three-tiered: consisting of body geometry, skeleton, and constraints.

In Softimage|3D's Schematic view, the skeleton and body appear as one hierarchy and the constraints are connected with yellow lines to the individual constraint objects. The three-tiered system looks like a traditional marionette in the schematic window; the character animation controls (also called constraint objects) look like wood crossbars, and the constraint lines resemble mari-

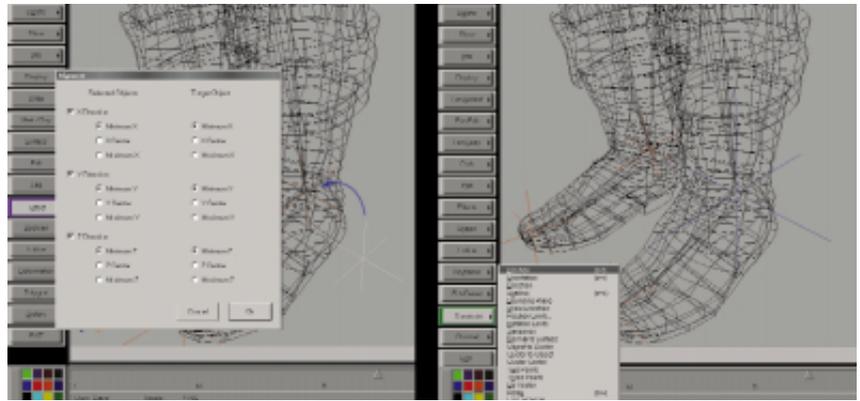


FIGURE 1. You can use Effect→Alignment to get the null into position, then use Constraint→Position to attach the leg end effector to the null (now colored blue and scaled bigger for easy selection).

onette strings. As a result, this system is sometimes referred to as the “marionette” approach.

In Softimage|3D, a constraint object isn't a special kind of object. It's simply any object to which another object (or objects) has been constrained. Normally, you'd choose something generic, like a null or a spline—a simple object that won't render and that's easy to see and select.

Leg Constraints

Let's start with the legs. We'll use a null as a constraint object for each leg. To create the first null, choose Get→Primitive→Null. We need to place this null in the same position as the left leg end effector. With the null selected, choose Effect→Alignment. In the dialog, select the X, Y, and Z Direction check boxes. Click OK, then select one of the leg end effectors. The null jumps to the position of the end effector. To complete the process, select the end effector and choose Constraint→Position, then select the null. Once the constraint is active, you can no longer move the end effector—its position is determined entirely by the position of the null (Figure 1). Depending on how large your character is in the workspace, you may want to scale the null up to make it easier to see (even in shaded mode with Show Icons active).

To control the plane of the 2D chain created in part one, you can apply an Up Vector constraint object. (This object controls the orientation of the chain, for

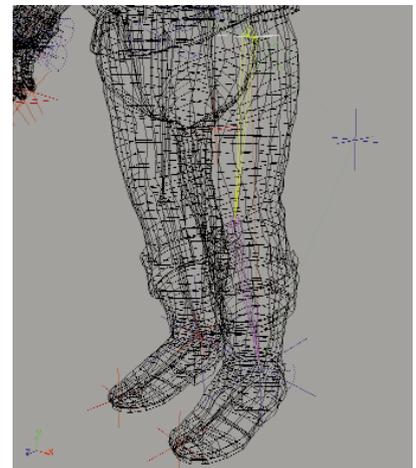


FIGURE 2. One way to control the plane of a 2D chain is an Up Vector constraint. Here, the first joint of the leg points along its Y axis to the null. Moving the null from side to side twists the leg chain, making the knee point in or out.

instance, the positioning of the knees and arms in relation to the body.) Position a null behind the body near the legs. It's best if you position it exactly on the current plane of the legs to maintain the default pose. To do so, open an ortho window with the upper leg joint selected. Choose Y directly above the window to align the view with the Y axis of the selected object. Position the null exactly behind the middle line in the joint of the upper leg (Figure 2). Select the joint again, choose Constraint→Up Vector, then select the null. The leg chain should point directly away from the null without moving from its cur-

rent orientation. Now, add the null into the constraint hierarchy. Repeat the process for the other leg, and finally parent both constraint nulls to a parent null.

Saving a translation keyframe (SaveKey→Object→Explicit Translation→All) on each of the leg nulls lets you experiment freely with the positions of the legs without losing the original positions of the nulls. Once the constraints are active, you can no longer reliably use Skeleton→Reset Actor to return the character to its default position; constrained parts will always snap back to their constrained positions after a refresh. If you place the constraints just right for the default position and save keyframes, you can always return to the original pose when you need to.

The constraint for the hips chain (with its root near the belly) should be the main parent of the rest of the body. Instead of using a null this time, it might be easier to use a spline (in this case a primitive square, which is a linear spline drawn in a square shape). Spline objects, like nulls, won't show up in renders and they aren't solid-shaded in OpenGL mode. They appear to hover over your character's geometry, so they're easier to select. And, unlike nulls, spline points can be scaled to whatever size is convenient, while the center remains at the default scale of 1.

Choose Get→Primitive→Square. Rotate 90° in X, then choose Effect→Freeze→Transformations. To position the square at the hips, choose Constraint→Position (instead of Effect→Alignment) and select the root of the hips chain. We also want this control to rotate the hips, so choose Constraint→Orientation and then select the hips' root. The square moves and rotates into position. Next, you need to undo the constraints while keeping them in place. Choose Constraint→Relax and hit the middle mouse button to relax both constraints. Now repeat the process the other way, constraining the hips' root to the square's position and orientation. Select the root of the hips chain, choose Constraint→Position, and select the square.

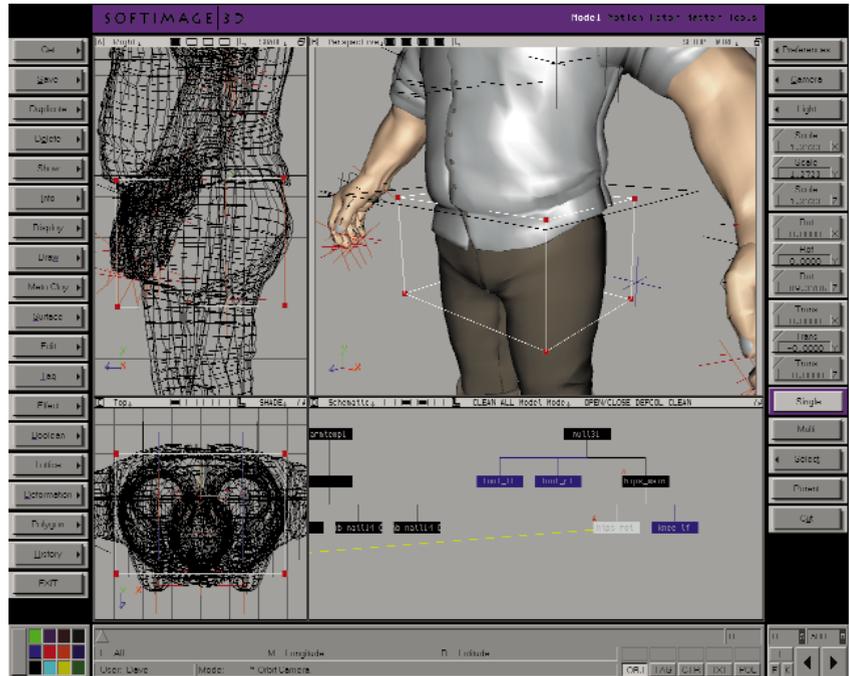


FIGURE 3. The points of a spline cube can hover conveniently over the hips area, while its center (the red and green arrows in the upper left window) remains in position at the root of the hips chain.

Repeat this process with Constraint→Orientation, and parent your new hips control to the parent of the two leg controls. To keep the default pose (as I mentioned earlier), save an Explicit Translation and Rotation keyframe on the square. This control now rotates and translates the whole body, except the legs.

Independent Hip Rotation To control hip rotation independently of the upper body, you can simply rotate the hip joint, which is just beneath the root that was constrained earlier. Because the two leg chains are parented to this joint (or the end effector), the upper legs will always move with the hips. But selecting joints in shaded mode is problematic. Unfortunately, Softimage3D doesn't have an X-ray mode that allows you to see joints within the shaded geometry, so grabbing the hip joint can be tricky if you aren't in wireframe mode. You could use the selectability feature to freeze the character's geometry and make it unselectable. To select the geometry you want to freeze, choose Select→Selectability→Toggle Selection. Still, without a wireframe window visible, it's hard to be sure what you've selected. Of course, you could select the

joints in the schematic window. Instead, I find it simpler to keep applying constraint objects even to parts like this that don't need constraint objects but can benefit from the easier selection and visualization. This approach has the added benefit of maintaining all the animation information in one hierarchy and one animation file.

For areas like this, I tend to use a spline cube as a constraint object. (Again, any object can be used as a control.) Similar to a spline square, a spline cube is made from a single linear spline, drawn in the shape of the cube. In shaded view, the cube doesn't appear solid-shaded; only the lines show, hovering over the character's geometry. To make a spline cube, create a default primitive cube, then use Magnet→On Point (located in the window's Layout Options menu) to draw a linear curve over each point and segment of the real cube. When you're finished, discard the default cube and your spline cube is ready.

Use Constraint→Orientation and Constraint→Position to move the spline cube into position over the hip joint. In this situation, you'll never need the constraint to translate the joint, only to rotate it, so after relaxing the translation constraints, use

Constraint→Orientation on the hips joint to the spline cube. Note that constraints can be used to aid in forward kinematics as well as inverse kinematics (IK). In this situation, one constraint is used for driving rotations only (forward kinematics), while others affect translation (as with the legs) and still others affect both rotation and translation (as with the hips' root).

Parent the spline cube to the spline square hip control, then save a translation and rotation keyframe. You can also tag the points of the spline cube and translate them to the best position for visualization purposes. This is fine, since you're not changing the object's center. The spline cube will function the same way as before, but it will appear to hover over the entire area affected by it rather than over the root only (Figure 3).

Inverse and Forward Kinematics A useful arm setup will allow you to switch between inverse and forward kinematics. If a character is leaning on a desk, for example, you'll want the arm to stay positionally independent from the body. As you make changes in the spine and hips, the hand can stay locked to the desk through IK. But when the position of the hand isn't the top priority, many animators prefer the look and control of forward kinematics. It's easier to overlap the rotations of the arm joints—normally, you'd want the lower arm to rotate into place slightly after the upper arm. This procedure is difficult to simulate using IK.

The simplest way to allow for both inverse and forward kinematics is to use the dopesheet to deactivate a portion of the time line for the relevant constraint. To do this, apply a constraint null to an arm the way you did earlier for the legs. Select the arm end effector and open the dopesheet. Click once to the left of the green bar. You should see "CnsPos" in the green bar, which indicates that a position constraint has been applied to the end effector. Drag a box selection over part of the time line. Click DEACT just above in the dopesheet window. That portion of the time line will turn gray,

and the end effector will no longer be constrained to the null in those frames.

A more reliable and interactive way to allow for both inverse and forward kinematics is to apply an expression to the arm end effector telling it to attach to a null only if a certain condition is met. For instance, you can get a third null to act as a switch: When it translates to or above a value of 1 in Y, the IK constraints will become active. Otherwise, you can use direct forward kinematics on the joints.

To make this setup, get a null to act as the switch (called a slider), and name it `ikswitch` (in Info→Selection). Name the arm control `arm_if`. Select the end effector and choose Motion→Expressions→Edit. Type "etrnx" in the Affected Element portion after the object name or select `etrnx` from `Fcurves`. `Etrnx` is the function curve that describes the explicit translation in X.

For the part of the box labeled Expression, put your cursor in the field, then click on Condition in the Functions box. Click Insert, which inserts the condition "template" into the Expression area. Select the `<cond>` part and type over it with the condition part of the equation. Since you want the expression to apply only when the slider null is at or above a value of 1 in Y, type in `ikswitch.etrny>=1`, which means "if `ikswitch` is equal to or greater than 1 in Y translation..." Now replace the `<true_expr>` part with `arm_if.etrnx`. This means "then make my value the same as `arm_if`." For the `<false_expr>`, type in `this.etrnx`, which means "otherwise, let my value remain unchanged (do nothing)." Hit Validate to validate the script and make sure you have no syntax errors (Figure 4).

Select the Expression part and click Copy. Click on Next, then paste it into the expression part. Change the Xs to Ys, including the one in the affected element. Hit Validate, then repeat the process for Z. This will give you three expressions, one for each axis.

Try out the expressions by translating `ikswitch` up to 1, then moving `arm_if` around. The arm end effector follows. Move

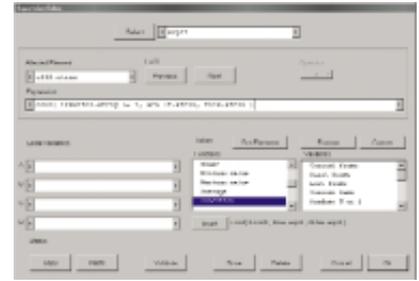


FIGURE 4. An expression can be used to control whether an end effector will be constrained to a null at any given time. This expression tells the end effector to follow its null, `arm_if`, only when the null `ikswitch` is greater than or equal to 1.

`ikswitch` to 0, and the arm will no longer be constrained to the null. For constraint switching during animation, simply animate the Y position of the `ikswitch` null.

If you attach an Up Vector constraint to the arm chain (as you did previously with the legs), you'll need to add expressions deactivating that constraint as well.

Alternatively, you might apply constraints to the joints for the forward kinematic motions, the way you did with the hips rotation control. Joint rotation controls are easier to select, and they maintain all the body animation under one null. However, they're more difficult to set up because you have to turn some constraints off and turn others on at the same time.

Whichever method you use, be sure to parent any constraint objects to the appropriate parent in the constraint hierarchy (in this case, either the main null or the main hips constraint). And don't forget to save an explicit translation and/or rotation keyframe.

Spine and Chest A common approach to constructing a spine is to make a three-joint chain from the hips to the base of the neck. Since every node in a spine must be able to twist in X as well as rotate in Y and Z, a 3D chain should be used. (Note: Drawing 3D chains in the Right window is different than with 2D chains—the Z axis always points along the global Z axis. You may want to draw a 2D chain and switch it to a

ANIMATORS ANONYMOUS

3D chain using Skeleton→Toggle 2D/3D for more consistent rotation values.) When animating, pose the spine and chest by beginning at the bottom of the chain and rotate each joint successively to the top.

An easier approach might be to control the motion from the top down using what could be called an artificial chain. That is, instead of building a multi-joint chain, you'll link together several single-joint chains. In the Right window, draw three single-joint chains up the spine, aligning the bottom joint's end effector with the next joint's chain root. Constrain each end effector to the next chain's root, up the line from bottom to top. Position several nulls between each chain, and constrain each chain root to a null.

For the top chain, constrain the root both positionally (Constraint→Position) and orientationally (Constraint→Orientation). Now you can control the top of the spine via one null and each link using the other nulls. The top chain or joint can be positioned and rotated quickly and the other links translated quickly to form a pose. Using this method, the joints detach when the nulls are pulled out of reach, creating a stretching effect useful for spines. Note that your body's spine can stretch a bit as well. But the biggest advantage here is the ability to do top-down posing.

While each node can translate and the top node can translate and rotate in any direction, all twisting in X down the chain is controlled by the top node. Since the joints are not a single IK chain but a series of distinct chains linked artificially, you can drive their rotations through expressions. Otherwise, the expressions would conflict with the chain's built-in IK.

Because the top joint is controlled by a null oriented like the chain root, you need to link the X rotation (twist) of each lower joint to the Y rotation of this chest null (roots are oriented differently than joints). Select the first joint below the top one and choose Expressions→Edit. Type "rotx" where the cursor is, in Affected Element. In Expression, type or select the name of the

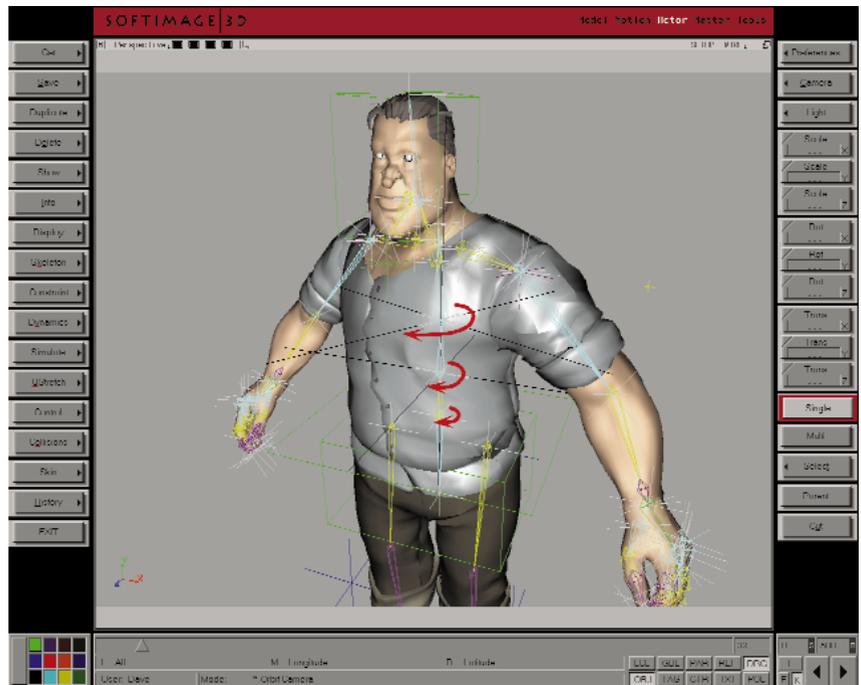


FIGURE 5. In this top-down approach to spine control, each node up the spine can be translated for positioning while the gradual twisting down the spine is controlled through expressions based on the rotation of the top (chest) joint. (Note: This screen shot is a composite. Softimage3D doesn't have a display mode that allows a skeleton to show through shaded surfaces.)

top null (from the Scn Elements box). After the name and period, type "roty" and then "/2" or 3 or 4, depending on how many joints you've made to represent the spine. This procedure tells the joint to rotate half as much in X as the chest null rotates in Y. Hit OK and repeat this process with each spine joint (you can use Expressions→Copy), decreasing the value of the rotation by increasing the division number. Last, you need to zero the rotation values of the chest null to make it work properly—make the value 0,0,0 with an effective value of 0,90,0. To do this, parent the chest null to a new null with a rotation of 0,90,0 (or parent to the hips square which is oriented this way) and set the chest null to 0,0,0. When the controlling null rotates, it will do so in local space, yielding more predictable rotation values. Its rotation will now twist the spine in decreasing amounts from the top down (Figure 5).

Many other methods can be employed to set up spines and spine-like structures such as tails and long necks. You can con-

strain a series of joints to clusters along a spline and animate the spline. You can deform the spline to another spline with fewer points for even finer control. This approach allows you to control many joints via very few points. You can drive rotations of a series of joints by rotating a single null through expressions.

Remain Flexible Once you've set up constraints, it's easier to manipulate all of your controls using the Preferred Transformation feature. For instance, select the chest control and choose Info→Transformation Setup. Click on the Rotate button and choose Y (check off X and Z). Whenever you select the chest control, the transformation mode will switch automatically to the preset choice. This feature is especially helpful for rotations, because you don't need to remember which axis you're most likely to use whenever you go to rotate the object. You can still use the other axes, of course, but this method can speed up the animation workflow quite a bit.

ANIMATORS ANONYMOUS

It's important to remember that a character setup can be changed as the need arises. Constraints can be deactivated and temporary constraints can be added. The whole thing can be thrown out for a particular shot and replaced by a setup that better fits the needs of the movement. What if your character has to lean its elbow on a

counter, chin resting in hand, while moving its torso slightly and gesturing with the other hand? With the current setup, which is designed to accommodate hand positions only, this could be a very difficult shot. But if you design a setup to accommodate the specific motion you need, your animation will go much more smoothly.

We've looked at only a few options and essentials for character setups. With an engineer's attention and an eye open to the numerous possibilities, you'll find character setup a fascinating study and an amazing help during character animation. ●