David J. Sushil davidjsushil@gmail.com

Animating Objects in Torque

This document picks up where the Importing Assets From 3D Studio Max tutorial leaves off. If you haven't done so, I highly recommend reading that paper and familiarizing yourself with the process before proceeding. Once you're comfortable with importing static assets, animated assets will be a cinch. Of course, consult with your doctor if you experience any dizziness, painful or difficult swallowing, or bouts of inexplicable terror. We will address three aspects of animating objects – first, how to create animation sequences in 3D Studio Max; second, how to properly configure your files for Torque; and third, how to trigger the animations within the engine. Let's begin!

Animating in 3D Studio Max

At this point, I'm going to make an assumption that you're fairly comfortable with using 3D Studio Max. As a result, I'll only cover areas of the program that deal specifically with animation, as this might be a new concept for some readers.

The Basic Process

To begin, open an existing model, or create a new one. Once all of the model architecture is in place, we're ready to begin animating. Animating is simply the process of setting keyframes on the timeline. The timeline is located at the bottom of the screen, just below all four viewports. Notice the time slider on the left hand side? You can drag the time slider left and right to move forwards and backwards through your animation. Likewise, the animation controls are located below the timeline on the right. There are buttons for playing and pausing the animation. These will be useful later.

For now, we want to focus on two important buttons. The first is the Toggle Auto Key Mode button, indicated by the text "Auto Key"; the second, the Set Keys button, indicated by a picture of a key.

When you are ready to begin animating, click on the Toggle Auto Key Mode button. When Auto Key mode is activated, the timeline will become red. Please be careful! It is painfully easy to forget that you are working in Auto Key mode, so it's best to get in the habit of turning off Auto Key mode whenever you are finished animating. If you don't, it could have serious consequences!

With Auto Key mode enabled, click on an object within your scene. When an object is selected, it is possible to set keyframes for that object on the timeline. Each object in the scene has its own private timeline, where only its own keys appear. To access this timeline, you must select the object.

By way of example, let's imagine we're animating an electric fan. The blades of the fan are a single object, which we want to rotate 360° from the start of the animation to the end. To do so, we'd select the blade object. With Auto Key mode enabled, we would need to set a keyframe at the start of the animation, on frame 0, by clicking on the Set Keys button (the one with the picture of a key). Notice how a red and green block is added to the timeline at frame 0. This is how 3D Studio Max indicates a frame has been set. Next, we would reposition the timeline slider to the end of the animation sequence, maybe on frame 50. With the timeline slider positioned, we could then rotate the fan blades 360°. When we are happy with the rotation, we would set a keyframe at frame 50, again by clicking the Set Keys button.

The effect of this process is that if we move the timeline slider back and forth between frames 0 and 50, 3D Studio Max interpolates the position of the blades accordingly. It's all mathematics – at frame 25, for example, the blades should be rotated approximately 180°.

We could also test this by playing the animation using the animation controls. However, since the default length of an animation is 100 frames, and our animation is only 50 frames long, it would appear as though our animation has stopped midway through. We can remedy this by clicking on the Time Configuration button, which is located in the bottom-right of the screen, and looks like a window with a clock in front of it.

From this menu, we can set the Start and End times of the animation. In our case, we would want the End time to be 50, or the last frame in our blade spinning animation.

In either case, once we are satisfied with our animation, we should turn Auto Key mode off, by click on the Toggle Auto Key Mode button.

The most basic forms of animation are transformations – that is, animations that involve moving, rotating, or scaling objects. Torque supports transformation animation, as well as a few other types. We'll explore those additional animation types in a moment.

Handling Multiple Animations

In case you were wondering, every animation for an object can be contained on the same timeline. In the case of our electric fan, we might have two different animations – one where the blades are spinning normally, a second where the blades spin crooked (as if the fan has been damaged), and a third animation for when our player gets bored and shoots the fan into smithereens.

We've already animated the normal spinning sequence. The second "damaged" animation would begin on frame 51, and end on frame 101. For this animation, we would rotate the blades 360° from start to finish – just like our normal animation, however, we would also rotate the blades forward and back slightly, to create the look that the blades are wobbling.

In the third animation, we'd deform the entire model into a broken heap. This animation would begin on frame 102, and end on frame 110. It's a quick animation.

The secret to keeping all of the animations "pure" is to make sure that their keyframes butt up against each other. That is, if our first animation ends with a keyframe at 50, then our next animation begins with a keyframe at 51. Keyframe 51 should reset our model to precisely the way we want it to look at the start of our second animation.

Otherwise – and this is the critical part – 3D Studio Max will interpolate from keyframe 50 onward. This could possibly result in a poor animation.

The question is, how do we tell Torque where one animation begins and another ends? We do so by creating helpers objects called Sequences.

Indicating Your Sequences

Sequences are invisible objects created in 3D Studio Max which Torque reads like an instruction manual. Sequences give each animation a unique name, indicate where an animation begins and ends on the timeline, and designate whether an animation is cyclic (looping) or non-cyclic (non-looping). Sequences have a few other properties as well, but in general these are the three most important properties.

Creating a Sequence

To create a sequence object, click on the Helpers button from the Create panel. In the dropdown listbox that appears, select General DTS Objects. This option will only appear, by the way, if you have loaded the DTS Exporter Utility from the Utilities Panel. See the Importing Assets From 3D Studio Max tutorial for instructions.

From the Object Type rollout, click the Sequence button. Go ahead and create a sequence by clicking and dragging in a viewport. A simple box will appear. Don't worry if this sequence isn't within the bounding box of your model – it's strictly a helper object that communicates between Max and Torque. It will not be rendered in the engine.

From the Modify panel, give this sequence a name. Also, if you want your animation to repeat, be sure that the Cyclic Sequence box is checked. Otherwise, uncheck the box, and the animation will only play one time. Rotating fan blades should be cyclic. On the other hand, the animation for opening a door should be non-cyclic.

Create a sequence object for each animation on your timeline. Our electric fan has three, which we'll call *spin*, *wobble*, and *die*. Spin and wobble are cyclic; die, non-cyclic (it only has to die once).

Begin/End Controllers

The final step is to indicate where each sequence begins and ends on the timeline. To do so, we open the Curve Editor. Do so by clicking the Curve Editor (Open) button, located on the toolbar, and represented by a line chart icon. A curve editor (sometimes referred to as a *dope sheet* by animators), is normally used to control the speed and smoothness of an animation. We will use it to set special keys for each sequence, which Torque will use to partition our animations in the engine.

To set these keys, locate the first sequence in the panel on the left. In the case of our electric fan, the sequence is located in the Objects section, and is title *spin*. Open the spin object by clicking on the plus sign to its left. From within this object, open up the Object (Sequence II) sub-object. Finally, click on the Sequence Begin/End controller. Now, set two keys on the timeline to the right – one at the start of the animation (frame 0), and one at the end of the animation (frame 50). You can set keys by selecting the Add Keys button in the toolbar, and clicking directly onto the timeline.

Repeat this process for the next animation. That is, in the panel on the left, locate the next sequence (in the electric fan example, it's *wobble*), drill down until you find its Sequence Begin/End controller, and set keys at the start and end of the animation.

Once each animation's beginning and ending frames have been set, we're pretty much done. Sequences, unlike collision meshes, don't need to be linked to anything in our scene, so we need not bother with the Schematic View, except to make sure that our sequences are unlinked, and that everything else is embedded and arranged properly.

Handling Other Forms of Animation

By the way, I did mention that animations can take different forms. Just for the sake of argument, select a sequence and click on the Modify panel. Expand the Export Control rollout panel, and observe the various forms of animation that Torque supports. As stated earlier, transforms are the most basic, and export by default – note the checked box next to Transform Animation.

Most of these animations are self-explanatory. For example, the Morph Animation option allows you to deform your model vertex by vertex. This is good for creating a melting effect. Likewise, the Visibility Animation option allows your model to fade in-or-out of the scene.

If your animation falls into any of these categories, simply check the corresponding box, and export your shape as you otherwise would for Transform animations.

Playing Animations in Torque

Once your sequence objects have been set, it's time to export your model. Even though we've added animations, exporting an animated shape works exactly the same way as for a static shape. Again, consult the Importing Assets From 3D Studio tutorial (seriously).

To test out your animation in Torque, we need to import the asset via code. It's a fairly simple process, really. Like any other object, the code looks like this:

```
datablock StaticShapeData(Fan) {
      shapeFile = "~/data/shapes/low-poly-fan/low-poly-fan.dts";
};

function addFan() {
      $fanObj = new StaticShape(Fan){
             datablock = Fan;
             position = "234.95 -206.45 194.5";
             scale = "1 1 1";
             rotation = "1 0 0 0";
      };
}
```

This code should be placed within a .CS file – say, fan.cs, located in demo/data/shapes/fan/, for example. Notice the datablock called fan, which has one property – the location of our .DTS fan model.

We instantiate the fan using our custom addFan() function. Of course, you should change the position, scale, and rotation properties of the fan object to reflect its transform data in your level.

See how we store our fan object in a global variable ($fanObj)? We'll need that variable to control our animations later.

In the meantime, to test this code, load Torque, and pull up your console window by pressing the tilde key (~). Enter the following code:

```
exec("demo/data/shapes/low-poly-fan/fan.cs");
addFan();
```

If everything worked correctly, your fan should appear. You'll notice, however, that it is not animated! Animations are controlled via a method called playThread. It looks like this:

```
$fanObj.playThread(0, "spin");
```

We call the method from the global variable $fanObj. It takes two arguments. The first we can ignore (it's usually 0), but the second argument is the name of the animation we want to play. Remember that *spin* is the sequence in which the blades rotate normally. It's a cyclic sequence, so once it begins, it continues indefinitely. We can call any animation sequence we want simply by calling playThread and passing the name of the animation as an argument.

To make our fan spin when it's loaded, insert the playThread code within our addFan() function. Simple enough. Have fun animating objects!