David J. Sushil davidjsushil@gmail.com

Creating Doors in Torque

Introduction

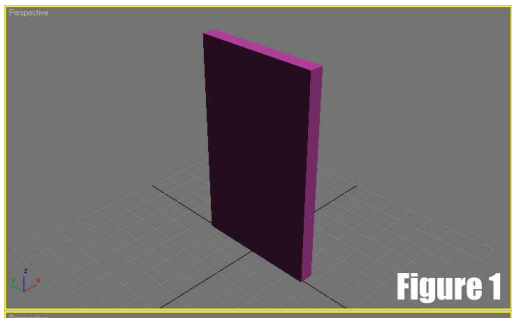
One truffle conspicuously missing from Torque's assorted box of datablocks is the ability to create a door. This tutorial will address some basic techniques for making triggered entranceways. We'll start by building a simple animated object in 3D Studio Max, and then explore two programming methods for interfacing the models in Torque.

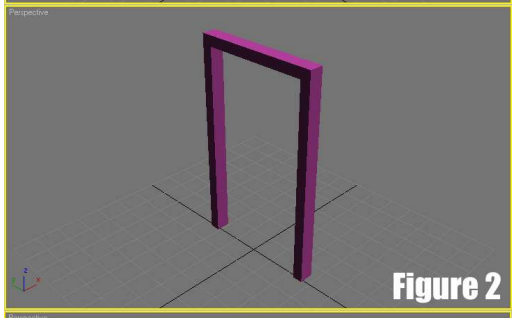
Building the Model

The first thing we need is a door model. Of course, you can create whatever kind of door you'd like – a Star Trek style sliding door, a boulder that rolls away from a cave entrance, or a drawbridge that lowers. In this tutorial however, we will create a free-standing door and frame with a handle. We are preparing this model for the purpose of animating it swinging inward. If you're pretty comfortable with your modeling skills in 3D Studio Max, then you can probably skip this section.

The Basic Model

We begin by using the Box tool in the Create panel. Our intention is to make a door frame, and we will do so by creating a compound object.





First, make a box that is similar to the one pictured in Figure 1. Be sure that it has only one length width and height segment each, and name it Frame. Make a copy of frame using Edit > Clone, and name the copy Subtract. Modify Subtract to be somewhat shorter and less wide than the original box. Finally, in anticipation of later steps, let's clone Subtract and name the copy Door. We'll use this copy in just a moment. For now, go ahead and hide it so it's not in the way.

Next, select Frame, and from the Create panel, select Compound Objects instead of Standard Primitives. Click the Boolean button. This will allow us to use Subtract to cut away the inside of Frame. In order to do so, we must identify Subtract as our Operand B. Click the Pick Operand B button, and then select Subtract. You should be left with an object that looks like the one in Figure 2. If so, convert it to an editable poly so Torque will understand it later.

Unhide Door, and modify its dimensions so it is not quite as thick as the surrounding frame (most doors aren't flush with their frames, after all).

We also need to create a doorknob for each side of the door. This can be done by placing a couple of boxes, cylinders, and spheres in the proper positions. You can see the finished door (with textures applied) in Figure 3. Name the additional objects Panel, Shaft, and Knob. To designate which side they belong on, append the names with Front or Back. Remember that adding trailing numbers (like 2, 3, 4, etc.) designates levels of detail, and may result in objects missing in your game.

To make things easier for animation, go ahead and group Door and your doorknob objects by selecting them all and selecting Group > Group from the menu bar. Name the selection Doorswing.

We've now completed our door model. In the next section, we'll look at how to animate it properly.

Animation

Now that we have our door object constructed, we need to create two animation sequences - one where the door swings open, and the other where the door swings closed.

Before we begin keyframing these animations, we need to edit the center of gravity for Doorswing. At the moment, if we rotate the door, it will rotate around the center, much like a revolving door would. This is because the Pivot point of our door is aligned to its center. To align the Pivot point to the doorframe, select Doorswing, and from the Hierarchy panel, select Affect Pivot Only. The Pivot will appear in the scene as a thick red, blue, and green tripod. You can then use your Select and Move tool to align the Pivot to the edge of Frame. With the Pivot repositioned, turn off Affect Pivot Only. We're now ready to animate the door.

As with any animation, we begin by turning on Auto Key mode, located below the timeline in 3D Studio Max. Select Doorswing. We will set four keyframes on the timeline, representing the door at four stages:

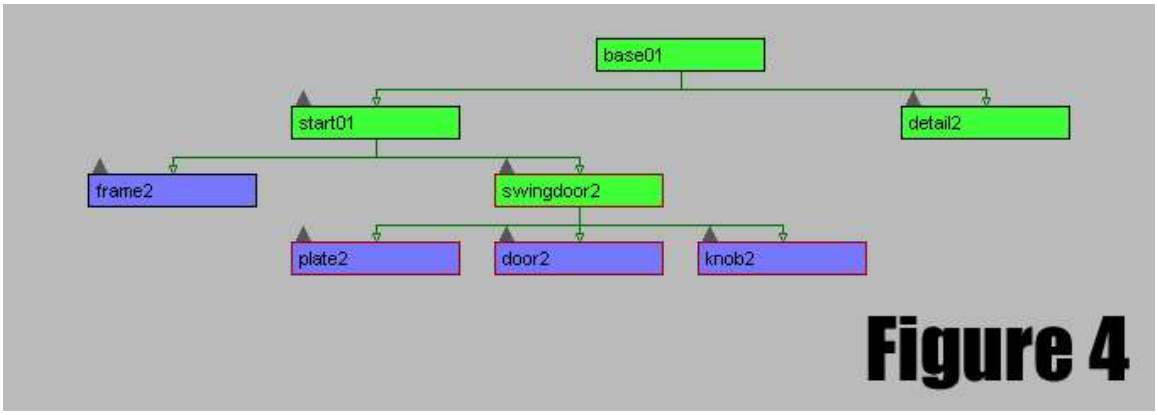
Frame	Physical Description	Sequence Position
0	The door completely closed	Start of the Open sequence
35	The door completely opened	End of the Open sequence
36	The door completely opened	Start of the Close sequence
71	The door completely closed	End of the Close sequence

Once you have finished animating these sequences, be sure to turn Auto Key off - otherwise additional changes to your scene might be contingent on the timeline, which is never good. By the way, if you are unfamiliar with how to keyframe animation, please see Animating Objects in Torque for additional instructions.

Interfacing with Torque

With our model built and our basic animation complete, we now focus our attention on preparing our door for use in Torque. This consists of three steps: first, embedding our shape like any other object; second, creating an animated collision mesh; third, creating sequence objects that indicate where our animations begin and end.

The initial step - embedding the shape - should be familiar to you by now. By way of a refresher, we will briefly recall the steps necessary for a proper export. First, select all the objects which will appear in your scene (Frame and Doorswing, and anything else), and click the Renumber Selection button in the DTS Exporter Utility rollout of the Utilities panel. Assign the selection the number 2. Next, click the Embed Shape button, which assigns helper objects to your scene. Check your Schematic View window to make sure all the normal helper objects are present and that your objects are connected to them properly. Your schematic view window should look similar to Figure 4.



In addition, we need to create a bounding box (bounds), which encompasses the all visible objects in our scene. Bounds is not linked to anything in our Schematic View, and it may be hidden from sight once created.

The next step is to create our collision meshes - Collision-1 and Col-1 - and animate them to match the motion of Doorswing. From the Create panel, select the Box tool and create a primitive that roughly matches the size of Doorswing. Name it Collision-1, and convert it to an Editable Mesh. Clone it, and name the copy Col-1. Animate both shapes using the same techniques outlined in the Animation section above - including moving the Pivot point.

When you have finished, be sure to link Collision-1 and Col-1 to their parents in the Schematic View - base01 and start01, respectively.

Finally, we can create our sequence objects. These objects will be named Open and Close, and they correspond with the animations we created previously. Use the Sequence button - located in the General DTS Objects dropdown section of the Helpers panel. It is very important that you set both sequences to be non-cyclic - that is, uncheck the Cyclic Sequence box in the Modify panel for both sequences.

You also need to use the Curve Editor window to designate where each sequence begins and ends on the timeline. Do so by clicking the Curve Editor (Open) button on the toolbar. Find the Open sequence, and drill down through its menus until you find the Sequence Begin/End controller. Set keys on the timeline at 0 and 35. Repeat the process for the Close sequence, but set the keys to 36 and 71.

At long last, we are ready to export our model. Of course, it won't function properly yet, but we can at least check it out in the Torque Game Engine and see if it looks nice. Export the shape by click the Whole Shape button in the Utility panel. Save the model in your Torque file hierarchy, preferably in demo/data/shapes/door/door.dts.

Next, we'll explore how to code door behavior in TorqueScript.

A Whole System of Doors

Create a new .cs file in demo/data/shapes/door/ and call it door.cs. This file will contain all of the code necessary for a functioning door. Add the following lines of code:

```
datablock StaticShapeData(door){
     shapeFile = "demo/data/shapes/door/door.dts";
};

datablock TriggerData(doorTrigger) {
     tickPeriodMS = 1000;
};
```

These statements create two datablocks - a StaticShape called door, which references our door model, and a Trigger called doorTrigger, which we will use to execute our animations in the engine.

Next, we need a function that will create instances of these datablocks. We'll call this function addDoor(), and it looks like this:

```
function addDoor(){
      %doorObj = new StaticShape(){
            position = "233.3 -206.45 191";
            scale = "0.05 0.0575 0.0505";
            rotation = "0 0 1 -90";
            datablock = door;
      };

      %obj = new Trigger(){
            position = "232.3 -205.25 191";
            scale = "1 1 1";
            rotation = "0 0 0 0";
            dataBlock = doorTrigger;
            polyhedron = "0.0000000 0.0000000 0.0000000 2.0000000
                        0.0000000 0.0000000 0.0000000 -2.0000000
0.0000000                     0.0000000 0.0000000 2.0000000";
            doorID = %doorObj;
      };
}
```

These two statements basically instantiate copies of our datablocks, assigning them to a particular location in the level. You will of course need to alter the position, scale, and rotation of both objects to correspond with their location in your level.

Notice our new Trigger stores the ID number of the StaticShape door we just created. If you wanted, you could create a whole network of doors in your level, simply by copying these two statements and altering the transform data in each (position, scale, rotation). So long as your Trigger stores the ID number of its accompanying StaticShape, the system will work.

Now that we have a door and trigger in our level, we need code to animate them. The code is as follows:

```
function openDoor(%id){
      %id.playThread(0, "open");
}

function closeDoor(%id){
      %id.playThread(0, "close");
}
```

Each function takes one argument - the ID number of the door - which it uses to call either the Open or Close animation sequence we created in 3D Studio Max.

Now all we need are the trigger events which call these two functions. They are:

```
function doorTrigger::onEnterTrigger(%trigger, %this, %obj){
      openDoor(%this.doorID);
}

function doorTrigger::onLeaveTrigger(%trigger, %this, %obj){
      closeDoor(%this.doorID);
}
```

This file is finished. You can test it in Torque by loading your mission, pulling up the console (~), and running the following two lines of code:

```
exec("demo/data/shapes/door/door.cs");
addDoor();
```

By walking up to the door, it should open. When you walk away, it should close. Mission accomplished!