Chapter 2 Creating Animations

Chapter Overview

Animation Models

Creating Animation Scripts for Static Data

Creating Animation Scripts for Dynamic Thresholds

Creating Animation Scripts for Dynamic Time-Dependent Data

Advanced Choreographing Using Third Party Software

Chapter Overview

This chapter gives a detailed description of how to create rendered animations using NCSA Isosurface Visualizer in part by providing several animation scenarios and example script files. It is assumed the user is already familiar with the meaning of all command line parameters as discussed in Chapter 1, “Getting Started.”

Animation Models

Before attempting to create an animation, you should be aware of the several modes of animation, or animation models, that ISOVIS is capable of producing. The animation model you use depends on the type of data you have and how you would like to view it. Three categories of animation models are discussed below.

The first category of animation relevant to ISOVIS is animation of static data and a constant threshold. In this case, only one dataset is viewed while using multiple orientations. This process corresponds to running ISOVIS multiple times while varying parameters like -r, -t, -s, etc (i.e. rotation, translation, scaling etc).

The second category of animation relevant to ISOVIS is animation of

static data and a varying threshold. In this case, only one dataset is viewed while using multiple thresholds and possibly multiple orientations.

The third category of animation relevant to ISOVIS is animation of dynamic time-dependent data. In this case, one dataset exists per time increment. Usually there is a one-to-one correspondence between the time increment and the frame increment (i.e., one rendered dataset per frame of animation). In this animation model, you run ISOVIS multiple times while varying the name of the file used for the three dimensional dataset which you derived from the original time-dependent computational simulation. You may choose to simultaneously vary other parameters and/or the threshold.

Creating Animation Scripts for Static Data

You can create animation scripts for static data in several ways. Some of these methods, as well as examples, are outlined below. It is intended that these examples serve as a template to allow you to create an ideal animation script for your particular requirements.

Because the data is static, you will only deal with one dataset. For sake of example, the dataset file is called “dfile.” Of course in practice, you would substitute the name of your file.
In this first example, the goal is to create an animation of the static data which rotates the object geometry about the z axis. For clarity, a unique file name is given to each rendered output file. The easiest way to create the animation is to develop a shell script using your favorite editor, such as vi editor, which looks like the following:

```
# Shell script to create an animation of a 360 degree rotation #
about the Z-axis using isovis. Images will be rendered
# in an ntsc-sized viewport.
#
isovis -r 0 0 0 -s 2.5 2.5 0.2 -ntsc -o f.0000.hdf\
   dfile 10.0
isovis -r 0 0 1 -s 2.5 2.5 0.2 -ntsc -o f.0001.hdf\
   dfile 10.0
isovis -r 0 0 2 -s 2.5 2.5 0.2 -ntsc -o f.0002.hdf\
   dfile 10.0
. (gap here represents script entries 3 through 358 not shown)
isovis -r 0 0 359 -s 2.5 2.5 0.2 -ntsc -o f.0359.hdf
   dfile 10.0
```

Once created, all that remains is to make the shell executable (e.g., with the UNIX command “chmod”) and execute it. Obviously, it can become quite tedious to create the above script manually with an editor, especially as the number of frames grows larger. Instead, you could write a program or use a UNIX utility which generates scripts substituting variables where appropriate. However, with a little practice programming shells, you can write the previous script in a shell loop as shown here:

```
# Same functional shell as the last example, except now the
# script is executed through shell programming. Use "sh" to
# execute this shell.
#
```

```
i=0
var2='.hdf dfile 10.0'
while [ "$i" -le 359 ]
do
   var1="isovis -r 0 0 $i -s 2.5 2.5 0.2 -ntsc -o f."
   if [ "$i" -le 9 ]
   then
        var3="000$i"
   elif [ "$i" -le 99 ]
   then
        var3="00$i"
   elif [ "$i" -le 999 ]
   then
        var3="0$i"
   else
        var3="$i"
   fi
   var4="$var1$var3$var2"
   $var4
   i=`expr $i + 1`
done
```

Once you execute the script, 360 files will exist in the current directory labeled f.0000.hdf through f.0359.hdf. These files may take up quite a bit of disk space, as animations usually do. To view the animation, transfer the images from disk to video tape (how one goes about accomplishing this is beyond the scope of this manual) or use a utility, such as NCSA showr24, which displays the images in succession.

Creating Animation Scripts for Dynamic Thresholds

In this example, an animation of the static data is created. This time the data will not be rotated, but the threshold will change with each frame. The thresholds used will depend on the values contained in your actual dataset. For this example, we will assume that the data values vary between 0.0 and 100.0. As before, a unique file name is given to each rendered output file, and the animation is created through the use of a shell script:

```
# Shell script to create an animation of varying threshold
# Images will be rendered in an ntsc-sized viewport.
#
isovis -s 2.5 2.5 2.5 -ntsc -o f.0000.hdf dfile 0.0
isovis -s 2.5 2.5 2.5 -ntsc -o f.0001.hdf dfile 1.0
isovis -s 2.5 2.5 2.5 -ntsc -o f.0002.hdf dfile 2.0
  . (gap here represents script entries 3 through 99 not shown)
isovis -s 2.5 2.5 2.5 -ntsc -o f.0100.hdf dfile 100.0
```

Creating Animation Scripts for Dynamic Time-Dependent Data

Creating animation scripts for dynamic time-dependent datasets is slightly more complex than for static datasets in that there are multiple input datasets as opposed to one. For sake of example, the input files are called “d.0000.hdf” where the 0000 field of the file names is replaced with the appropriate time slice reference.

Below is the same script as shown in the first example of the section, “Creating Animation Scripts for Static Data”; however, now it is set up for dynamic time-dependent data:

```
# Shell script to create an animation of a 360 degree rotation
# about the Z-axis of time-dependent data using isovis.
#
isovis -r 0 0 0 -s 2.5 2.5 0.2 -ntsc -o f.0000.hdf d.0000.hdf \
   10.0
isovis -r 0 0 1 -s 2.5 2.5 0.2 -ntsc -o f.0001.hdf d.0001.hdf \
   10.0
isovis -r 0 0 2 -s 2.5 2.5 0.2 -ntsc -o f.0002.hdf d.0002.hdf \
   10.0
.
isovis -r 0 0 359 -s 2.5 2.5 0.2 -ntsc -o f.0359.hdf \
   d.0359.hdf 10.0
```

You can also create a similar script for dynamic data from the script in the section, “Creating Animation Scripts for Static Data.” Using shell programming, rewrite that script in the following way:

```
# Same functional shell as the last example, except now the
# script is executed through shell programming. Use "sh" to
# execute this shell.
#
i=0
while [ "$i" -le 359 ]
do
   var1="isovis -r 0 0 $i -s 2.5 2.5 0.2 -ntsc -o f."
   var2='.hdf d.'
   var3='.hdf 10.0'
   if [ "$i" -le 9 ]
   then
        var4="000$i"
   elif [ "$i" -le 99 ]
   then
        var4="00$i"
   elif [ "$i" -le 999 ]
   then
        var4="0$i"
   else
        var4="$i"
   fi
   var5="$var1$var4$var2$var4$var3"
   $var5
   i=`expr $i + 1`
done
```

Advanced Choreographing Using Third Party Software

Now that you are familiar with what ISOVIS can do, you may wonder if you can create dramatic fly-by animations with complex motion paths. It can be done, but requires some work to accomplish and preferably some third party animation software. The third party software must have the ability to create a user-defined perspective and the ability to write out motion information in a form accessible to the user. The ability to create

the former is necessary since the perspective view transformation of the animation software must duplicate that used in ISOVIS. The motion information written out by the animation software is used as input to a script file you create for the ISOVIS animation. A systematic description of the steps required to initiate advanced choreographing using third party animation software is explained below. Be forewarned that the process is time-consuming and requires a thorough understanding of how ISOVIS and the third party software works. Advanced choreographing should only be attempted when you think it will greatly add to the understanding of observed simulation dynamics.

To use third Party software you first must :

1. Match the perspective of the animation package to that used in ISOVIS. This can be done empirically by “eye-balling it” or by somehow plugging in the actual perspective transformation numbers (these numbers are given in Chapter 3, “Technical Notes”). Since animation packages vary, how one actually goes about accomplishing this is beyond the scope of this manual and is left as an exercise for the user.

2. Create a dummy object in the animation package approximately the same size as the geometry in ISOVIS. This dummy object is used to create the desired motion path. It is recommended that a cube be used as the dummy object where the boundries of the cube are roughly coincident with the boundries of the geometry in ISOVIS. The viewed size of the dummy object should have the same rotation, scaling, and translation parameters as the viewed object in ISOVIS. Verify that rotations (cw vs. ccw) and coordinate system orientations (left-handed vs. right-handed) match up between ISOVIS and the animation package, and make appropriate adjustments if they do not. The dummy object should rotate about its geometric center and be centered at the world origin.

3. Create the desired motion of the object using the dummy object in the animation package. Do this by varying only the rotation, translation, and scaling parameters of the dummy object and nothing else. Orient the viewer position coincident with the
z axis. The exact viewer location is derived from the perspective transformation being used. Once the motion choreographing is completed and the desired motion is obtained, write out the x, y, z rotation, translation, and scaling information for the dummy object.

4. Write a program which creates a script file using the x, y, z rotation, translation, and scaling information obtained from the animation package, and substitute it in the appropriate command line parameters for ISOVIS. When run, the script combines the motion derived from the animation package with the features and rendering capabilities of ISOVIS.