Chapter 2 Creating Animations

Chapter Overview

Animation Models

Creating Animation Scripts for Static Data

Creating Animation Scripts for Dynamic Time-Dependent Data

Advanced Choreographing Using Third Party Software

Chapter Overview

This chapter provides a detailed description of how to create rendered animations using NCSA Height-Color Visualizer. Several animation scenerios are discussed and example script files are provided. 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

You must understand the several modes of animation, or animation models, contained within HCVIS before you attempt to create an animation. The animation model used depends on the type of data you have and how you would like to view it. Two categories of animation models are discussed below.

The first category of animation relevant to HCVIS is animation of static data. In this case, only one data set needs to be viewed while using multiple orientations. This process corresponds to running HCVIS multiple times while varying parameters like -r -t -s etc (i.e. rotation, translation, scaling etc) but keeping the height field file and color field file the same.

The second category of animation relevant to HCVIS is animation of dynamic time-dependent data. In this case, one data set exists per time increment. Usually there is a one-to-one correspondence between the time increment and the frame increment (i.e. one rendered data set per frame of animation). In this animation model, you run HCVIS multiple times

while varying the name of the files used for the height and color field which you derived from the original time-dependent computational simulation. You may choose to simultaneously vary other parameters as well.

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 data set. The term "data set" here refers to one set of data consisting of one height field file and one color field file, both of which are specified as command line arguments as described in Chapter 1, "Getting Started." For sake of example, the height field file is called "hfile" and the color field file is called "cfile." Of course in practice you would substitute the names of your files.

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 hcvis. Images will be rendered as
# wireframe in an ntsc-sized viewport.
#
hcvis -r 0 0 0 -s 2.5 2.5 0.2 -ntsc -W 4 4 2 -o f.0000.hdf\
   hfile cfile
hcvis -r 0 0 1 -s 2.5 2.5 0.2 -ntsc -W 4 4 2 -o f.0001.hdf\
   hfile cfile
hcvis -r 0 0 2 -s 2.5 2.5 0.2 -ntsc -W 4 4 2 -o f.0002.hdf\
   hfile cfile
. (gap here represents script entries 3 through 358 not shown)
hcvis -r 0 0 359 -s 2.5 2.5 0.2 -ntsc -W 4 4 2 -o f.0359.hdf
   hfile cfile
```

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. A better way might be to write a program or use a UNIX utility which generates scripts substituting variables where appropriate. However, with a little practice programming shells, the previous script can be written 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 hfile cfile'
while [ "$i" -le 359 ]
do
  var1="hcvis -r 0 0 $i -s 2.5 2.5 0.2 -ntsc -W 4 4 2 -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 the script is executed, 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 tool, such as NCSA Image, which views the images in succession.

Creating Animation Scripts for Dynamic Time-Dependent Data

Creating animation scripts for dynamic time-dependent data sets is slightly more complex than for static data sets in that there are multiple input data sets as opposed to one. For sake of example, the input height field files are called "h.0000.hdf" and the input color field files are called "c.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 previous section except 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 hcvis. Images
# will be rendered as shaded polygons using 24 bit/pixel
# output in an ntsc-sized viewport.
#
hcvis -r 0 0 0 -s 2.5 2.5 0.2 -ntsc -o f.0000.hdf h.0000.hdf \
   c.0000.hdf
hcvis -r 0 0 1 -s 2.5 2.5 0.2 -ntsc -o f.0001.hdf h.0001.hdf \
   c.0001.hdf
hcvis -r 0 0 2 -s 2.5 2.5 0.2 -ntsc -o f.0002.hdf h.0002.hdf \
   c.0002.hdf
.
hcvis -r 0 0 359 -s 2.5 2.5 0.2 -ntsc -o f.0359.hdf h.0359.hdf\
   c.0359.hdf
```

You may create the script as described in the previous section. Using shell programming, the same script can be written as follows:

```
# 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="hcvis -r 0 0 $i -s 2.5 2.5 0.2 -ntsc -o f."
```

```
  var2='.hdf h.'
  var3='.hdf c.'
  var4='.hdf'
  if [ "$i" -le 9 ]
  then
       var5="000$i"
  elif [ "$i" -le 99 ]
  then
       var5="00$i"
  elif [ "$i" -le 999 ]
  then
       var5="0$i"
  else
       var5="$i"
  fi
  var6="$var1$var5$var2$var5$var3$var5$var4"
  $var6
  i=`expr $i + 1`
done
```

Advanced Choreographing Using Third Party Software

Now that you are familiar with what HCVIS can do, you may wonder if you can create dramatic fly-by animations with complex motion paths. It has been 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 HCVIS. The motion information written out by the animation software is used as input to a script file you create for the HCVIS 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 HCVIS 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 HCVIS. This can be done emperically 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 to the user.

Since geometry in HCVIS is normalized to allow different data sizes to fit within the boundries of HCVIS window, next :

2. Create a dummy object in the animation package approximately the same size as the geometry in HCVIS. This dummy object is used to create the desired motion path. It is recommended that a rectangle be used as the dummy object where the boundries of the rectangle are roughly coincident with the boundries of the geometry in HCVIS. The viewed size of the dummy object should have the same rotation, scaling, and translation parameters as the viewed object in HCVIS. Verify that rotations (cw vs. ccw) and coordinate system orientations

(left-handed vs. right-handed) match up between HCVIS and the animation package, and make appropriate adjustments if they do not. The dummy object should rotate about its geometric center and be located in the Z=0 plane.

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 XYZ rotation, translation, and scaling information for the dummy object.

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