LSU Hydro Group Image Processing Guide & Documentation

July 24, 2001

1 Outline

This guide serves two purposes. First, it sets down in words the process of visualization used by this group currently. Second, it will hopefully act as a template for others to expand upon. The remainder of this guide contains the following information; a brief general introduction, an brief explanation of polyr, an introduction to Maya, and an explanation of the current system including relevant scripts.

2 General Intro

The computational fluid dynamics code used by the LSU Hydro group produces large (on the order of 10 MB) output files, usually containing density or velocity values. These files are produced on machines that are different than the machines that will do visualization. There are 2 ways to deal with this. One, create a "Heterogenous Computing Environment" (HCE). That is, as the hydrocode is creating density files, ship them back to the imaging machine (IM) and perform visualization tasks at the same time. Two, wait until the hydrocode has run and then ship all of the density files to the IM. The HCE is obviously the more efficient of the 2 scenarios, but it is also more complex (see John Cazes' dissertation and
`http://www.phys.lsu.edu/astro/cazes/dodconf/hce.html`
for a description). For the remainder of this guide, we will take the stupid, inefficient, and simple path and assume that all files to be imaged exist on the IM.

3 Polyr

In the end, we want an image of an object that contains 4 density levels, each denoted by a different color. The first step toward this is to create 4 separate files that contain vertices and polygons defining each of the density levels. This is done with a shell script called make_obj_cyl.sh. This shell script reads in the values of the density levels from four files; .den1, .den2, .den3, .den4. The leading 4 byte word is stripped off the unformatted Fortran, raw density file by the strip4 executable. The result of the stripping is a file den.cyl. The den.cyl file is then fed to the polyr executable. Polyr takes several options:

- -r number – sets the maximum number of polygons to create
- -cyl$isym numr numz numphi – set the geometry to cylindrical with isym=$isym (isym is a hydrocode variable that you have set in fort.7) and set the number of radial, vertical, and azimuthal zones.
- -O $HEREDIR/den.# – sets the output to OBJECT format, suffix .obj, to a file called den.#.
- $HEREDIR/den.cyl – sets the input file.
- $den1 – sets the density level value.

The .obj files are 'object' files that can be read by Maya.

4 Intro to Maya

Maya is Alias|Wavefront software that performs a truckload of functions, but we will be concerned only with it's volume rendering aspects. Maya is accessible on the following machines; immeroctane.phys.lsu.edu & immeronyx.phys.lsu.edu (these are the "imaging machines" from the previous section). However, you can, most likely, only access immeroctane (unless Joel lobbies for us to get accounts on immeronyx).

Once you are logged into an IM, you start Maya by entering,

```
maya -lic=complete
```

The first time you start Maya, several directories will be setup in your home directory. The most important of these are:

/maya/
This is the base Maya subdirectory.
/maya/4.0/
This is the version 4.0 subdirectory.
/maya/4.0/scripts/
This is the directory where all MEL scripts have to be kept.
/maya/projects/default/scenes/
This is where scenes are saved in Maya.
/maya/projects/default/images/
This is where the images produced by rendering are saved by default.

Now, the Maya user interface (UI) should be loading. This is the interactive part of Maya that should be used for the development of new imaging routines. We now discuss the most important features of the UI.

4.1 The Script Editor

All of the commands executed in the UI have associated MEL commands, where MEL is the Maya Embedded Language. Since these are the commands that are going to be used in our visualization scripts, it is important to be aware of their function and syntax. The Script Editor (SE) is a window that displays these MEL commands. From the UI the SE can be opened by choosing *Window* $\rightarrow$*General Editors* $\rightarrow$*Script Editor*. Once opened, the SE has 2 panes. The top is the history pane and shows the MEL commands that have been performed in the UI. The bottom pane is where you can enter MEL commands to be performed. To execute MEL commands that you have entered, choose *Script* $\rightarrow$*Execute*. Some notes of caution. The command history that is displayed by default in the history pane does not contain all commands, only the ones Maya deems important. To show all commands, choose *Script* $\rightarrow$*Show All Commands*. !!WARNING!! You will now see EVERY command that is executed, including window creation/destruction commands. If you perform too many commands, it may become difficult to untangle the information in the history pane.

4.2 Importing Files

When Maya begins, you will see what is referred to as a scene. The axes are at the center and your viewpoint is out of the plane. Note that the z axis does not point vertically. You now want to put objects (the wireframe

models that were created by polyr) into the scene. These objects should be in the ~/maya/myOBJfiles directory and named den.1-4.obj. (You need to create this directory and populate it with the following files: the raw density file, the polyr executable, an executable name strip4, 4 files called .den1-4, and the make_obj_cyl.sh shell script. You can copy these from /usr/people/barnes/maya/myOBJfiles/) In the UI, choose *File* →*Import* and a chooser window will appear. Select the myOBJfiles directory, highlight the den.1-4.obj files, and click on the 'Import' button. Alternatively, you could enter the following commands in the SE bottom pane,

```
file -import -type "OBJ" -rpr "den" "~/maya/myOBJfiles/den.1.obj";
file -import -type "OBJ" -rpr "den" "~/maya/myOBJfiles/den.2.obj";
file -import -type "OBJ" -rpr "den" "~/maya/myOBJfiles/den.3.obj";
file -import -type "OBJ" -rpr "den" "~/maya/myOBJfiles/den.4.obj";
```

(Note that all MEL commands must be followed by a ';'). You should now see the wireframe models in the UI (and they should be very big, you shouldn't see the edges). These wireframes are now Maya objects called polySurface1-4.

4.3 Selecting Objects

Now that you have multiple objects in your scene, you must be able to select specific objects. Position the mouse at the left edge of the scene and hold down the left button. Slide the mouse across the wireframes (at least through the center) and release the button. The blue lines of the wireframes should turn green. You have selected all surfaces. In the SE, you also could have entered the following in the bottom pane,

```
select -r polySurface1 polySurface2 polySurface3 polySurface4;
```

However you have chosen the surfaces, you can change the selected surface by going to the right side of the UI (called the Channel Box) and choosing *Object* →*polySurface1-4*. When all of the surfaces are selected the one that is current is denoted with green lines, the others have white lines. (As an aside, selecting can be undone by choosing *Edit* →*Undo*, by clicking in a part of the scene that does not contain an object, or by entering and executing

```
select -cl;
```

in the SE.) To select one and only one surface, we now turn to the Attribute Editor.

4.4 Attribute Editor

With all of the surfaces selected, go back to the top of the UI and choose *Window →Attribute Editor*. This brings up the Attribute Editor (AE) window. If you click on the 'Select' button at the bottom of the window, you should see the white-lined surfaces change back to blue-lined surfaces and the green remain green. The Channel Box now displays the properties of the selected surface. These properties or attributes are also shown in the AE. Let's now discuss how the attributes can be changed.

4.5 Changing Attributes of Selected Objects

There are 3 ways to change the attributes of any selected object. First, you can go to the Channel Box, click in any of the displayed number boxes, and manually change the number. Second, you can use the mouse to change some attributes. On the left side of the UI you should see 6 boxes arranged vertically with pictures of arrows, cones, spheres, and cubes. The arrow box is highlighted; this means that the cursor is in 'Select' mode. Clicking on the cone box allows you to translate a selected object. Clicking on the sphere box allows you to rotate a selected object. The cube box is used to change the scale of a selected object. For example, make sure you are in 'Select' mode and select all 4 surfaces. Click on the cube box. You should see colored boxes appear along the axes at the center of the objects. Put the mouse on the origin and hold down the left button. Moving the mouse to the right scales everything up, moving left scales down. However, with this method it is difficult to control the scale precisely. For better control, select all surfaces and then change the scale listed in the Channel Box to a specified value. The third way to change attributes is with MEL commands. In the SE, you could enter and execute,

```
select -r polySurface1 polySurface2 polySurface3 polySurface4;
setAttr "polySurface1.scaleX" 0.2;
setAttr "polySurface2.scaleX" 0.2;
setAttr "polySurface3.scaleX" 0.2;
```

```
setAttr "polySurface4.scaleX" 0.2;
```

We will return to attributes a little later when we discuss lights and cameras.

4.6 Giving Surfaces Material Attributes

In order to see all 4 nested surfaces in a rendered image, we must control the color and transparency properties of the surfaces. To do this, we need to assign materials to each surface. As before, there is more than one way to do this. In the UI, select a surface. Then go to the top of the UI and choose *Lighting/Shading* $\rightarrow$*Assign New Material* $\rightarrow$*Blinn*. (You could choose another material, but in our application we always use blinn.) This opens a 'blinn1' window. In this window, you can choose the color of the surface (click in the 'Color' box to bring up another window with a colorwheel), the transparency of the surface, and various other quantities. When you have changed the necessary attributes click the 'Close' button at the bottom of the window. If you bring up the AE for that surface you should now see three tabs; polySurface#, polySurfaceShape#, and blinn#. Alternatively, you could accomplish this by opening the SE and entering and executing the following,

```
select -r polySurface1;
createAndAssignShader blinn "";
setAttr "blinn1.color" -type double3 0 1 0;
setAttr "blinn1.ambientColor" -type double3 0.145098 0.145098 0.145098 ;
```

This set of MEL commands will make the innermost surface green and give it an ambient color. Examples of changing transparency can be found in the eq_view.mel script in /usr/people/barnes/maya/4.0/scripts/. !!Note that the script createAndAssignShader.mel must be in your ~/maya/4.0/scripts/ directory for this to work. You can get that script from /usr/people/barnes/maya/4.0/scripts/createAndAssignShader.mel. This is not a standard MEL command, it is a MEL script that has been written by developers and modified by Eric Barnes to run properly in batch mode. There are a lot of these scripts that Maya uses. They can be found (with some difficulty) in one of the subdirectories under /usr/aw/maya4.0/scripts/.

4.7 Lights, Camera, Action!

There are default lights and cameras that are used when a scene is rendered in Maya. If you want to create new cameras and lights, there are again 2 ways to do it. First, in the UI choose *Create* →{*Lights* or *Cameras*} →{*Light Type* or *Camera*}. You will then see a selected (green) icon at the origin of the scene. The attributes of this object can then be changed in the same way as described earlier. !! Note that scaling a camera does nothing while scaling a light changes its size not its intensity. Light intensity is a separate attribute. Lights and cameras are a little different from the surfaces. Besides specifying their positions, their 'look-at' point must also be specified. This is most easily accomplished by using the 'Show Manipulator Tool' (SMT). The SMT is created by clicking on the box below the cube/scale box on the left side of the UI. With the SMT, the position of the object as well as its look-at point can be changed with the mouse. The second way to create cameras and lights is with MEL commands. These commands may be seen in the eq_view.mel script. My suggestion for creating new cameras and lights is as follows. Use the SMT to get the orientation you want, write down the translation and rotation attributes, and then put those numbers in MEL command format in a script.

4.8 Getting Help in Maya

Maya is a humongous program. Accordingly, the help index is humongous also. In order to access Maya help, start a Netscape session on whichever IM you are on. (Kill any other Netscape process you might be running on your local machine). In the upper right corner of the UI, choose *Help*. There are several options, but *Search* and *MEL Command Reference* are probably the most useful. When using Maya help, don't expect to ever find exactly what you are looking for on the first try. It always takes a few misses to get anything. Also, there are two other help features. At a UNIX prompt, type maya -help or Render -help. You will see the options available to the maya and Render commands.

5 The Current Scheme

In brief, the current scheme is as follows.

1. The hydrocode produces files to be imaged.

2. Those files are moved to an IM.

3. The make_images.sh script is called. This is the master script for making images (obviously) and should reside in the ~/maya/4.0/scripts/ directory. This script (a copy of which can be found in /usr/people/barnes/maya/4.0/scripts/) does several things.

   (a) It asks for you to input the beginning and ending frame numbers, e.g., 1001 and 1100.

   (b) It then loops over these numbers and for each frame number transfers the proper raw density file (found in ~/infiles/) to the ~/maya/myOBJfiles directory. (Remember that this is the directory where polyr resides).

   (c) It calls the make_obj_cyl.sh script that creates 4 .obj files.

   (d) It creates a Maya scene (eq_scene.mb, the .mb denoting mayaBinary format) using the eq_view.mel MEL script.

   (e) It renders the scene, creating a file called equator<frame number>.tif in the ~/images/ directory.

These important scripts are included below.

MAKE_IMAGES.SH:

```
#!/bin/sh
BHOME=/usr/people/barnes
INDIR=$BHOME/infiles
OBJDIR=$BHOME/maya/myOBJfiles
SCENEDIR=$BHOME/maya/projects/default/scenes/
IMGDIR=$BHOME/images
echo "Enter beginning filenumber:"
read bt
echo "Enter final filenumber:"
read ft
echo " "
while [ $bt -le $ft ]
do
# get the raw density file
file=$INDIR/disk$bt
/bin/cp $file $OBJDIR/denfile
# call the script that makes .obj files
$OBJDIR/make_obj_cyl.sh
echo ".obj files made"
# next do the following 2 things; 1) makes a maya scene from 4 .obj wire-
frame
# files that come out of polyr, 2) renders that scene to produce a .tif image
# make the scene file called eq_scene.mb and pipe output to 'maya_output'
touch maya_output
maya -batch -script eq_view.mel >> maya_output 2>> maya_output
echo "Maya scene created"
# render that scene with the following flags set and pipe output to 'ren-
der_output':
# -cam specifies which camera to render the image from
# -ih & -iw specify the height and width of the image in pixels
# -ert on enables raytracing
# -eaa highest signals for the rendering to be done with the highest level of
edge anti-aliasing
# -ufil on specifies that a filter should be used (for smoothing the image)
# -pft box specifies a box filter to be used
# -of tif specifies an output file in .tif format
# -rd ¡path¿ specifies the output directory
```

```
# -p equator specifies and output file named equator.¡file format¿
# render_output has all of the render diagnostics including the total time of
rendering
touch render_output
Render -cam camera1 -ih 480 -iw 640 -ert on -eaa highest -ufil on -pft box -of
tif -rd $IMGDIR -p equator$bt $SCENEDIR/eq_scene.mb >> render_output
2>> render_output
echo "Rendering complete"
bt=‘expr $bt + 1‘
done
# get rid of the left-over denfile and .obj files
/bin/rm -f $OBJDIR/denfile $OBJDIR/den.*.obj
```

MAKE_OBJ_CYL.SH:

```
#!/bin/csh -f
set HEREDIR = /usr/people/barnes/maya/myOBJfiles
set isym = 1
#cd $HEREDIR
touch $HEREDIR/den.cyl
# Strip leading 4 bytes and make den.cyl file
$HEREDIR/strip4 $HEREDIR/denfile $HEREDIR/den.cyl
# Read density files
set den1 = `cat $HEREDIR/.den1` # Highest density isosurface (innermost
shell)
set den2 = `cat $HEREDIR/.den2`
set den3 = `cat $HEREDIR/.den3`
set den4 = `cat $HEREDIR/.den4` # Lowest density isosurface (outermost
shell)
#echo "isosurfaces at $den1 : $den2 : $den3 : $den4"
touch polyr_output
$HEREDIR/polyr -r 10000 -cyl$isym 130 130 128 -O $HEREDIR/den.1
$HEREDIR/den.cyl $den1 > polyr_output
$HEREDIR/polyr -r 10000 -cyl$isym 130 130 128 -O $HEREDIR/den.2
$HEREDIR/den.cyl $den2 >> polyr_output
$HEREDIR/polyr -r 10000 -cyl$isym 130 130 128 -O $HEREDIR/den.3
$HEREDIR/den.cyl $den3 >> polyr_output
$HEREDIR/polyr -r 10000 -cyl$isym 130 130 128 -O $HEREDIR/den.4
$HEREDIR/den.cyl $den4 >> polyr_output
# remove the den.cyl file
/bin/rm -f $HEREDIR/den.cyl
```

EQ_VIEW.MEL:

```
file -import -type "OBJ" -rpr "den" "/usr/people/barnes/maya/myOBJfiles/den.1.obj";
file -import -type "OBJ" -rpr "den" "/usr/people/barnes/maya/myOBJfiles/den.2.obj";
file -import -type "OBJ" -rpr "den" "/usr/people/barnes/maya/myOBJfiles/den.3.obj";
file -import -type "OBJ" -rpr "den" "/usr/people/barnes/maya/myOBJfiles/den.4.obj";
select -r polySurface1 polySurface2 polySurface3 polySurface4;
setAttr "polySurface1.scaleX" 0.2;
setAttr "polySurface1.scaleY" 0.2;
setAttr "polySurface1.scaleZ" 0.2;
setAttr "polySurface1.rotateX" 270;
setAttr "polySurface1.rotateY" 90;
setAttr "polySurface2.scaleX" 0.2;
setAttr "polySurface2.scaleY" 0.2;
setAttr "polySurface2.scaleZ" 0.2;
setAttr "polySurface2.rotateX" 270;
setAttr "polySurface2.rotateY" 90;
setAttr "polySurface3.scaleX" 0.2;
setAttr "polySurface3.scaleY" 0.2;
setAttr "polySurface3.scaleZ" 0.2;
setAttr "polySurface3.rotateX" 270;
setAttr "polySurface3.rotateY" 90;
setAttr "polySurface4.scaleX" 0.2;
setAttr "polySurface4.scaleY" 0.2;
setAttr "polySurface4.scaleZ" 0.2;
setAttr "polySurface4.rotateX" 270;
setAttr "polySurface4.rotateY" 90;
select -cl;
select -r polySurface1;
createAndAssignShader blinn "";
setAttr "blinn1.color" -type double3 0 1 0 ;
setAttr "blinn1.ambientColor" -type double3 0.145098 0.145098 0.145098 ;
select -cl;
select -r polySurface2;
createAndAssignShader blinn "";
setAttr "blinn2.color" -type double3 1 1 0 ;
setAttr "blinn2.transparency" -type double3 0.603922 0.603922 0.603922 ;
setAttr "blinn2.ambientColor" -type double3 0.133333 0.133333 0.133333 ;
select -cl;
```

```
select -r polySurface3;
createAndAssignShader blinn "";
setAttr "blinn3.color" -type double3 1 0 0 ;
setAttr "blinn3.transparency" -type double3 0.678431 0.678431 0.678431 ;
setAttr "blinn3.ambientColor" -type double3 0.113725 0.113725 0.113725 ;
select -cl;
select -r polySurface4;
createAndAssignShader blinn "";
setAttr "blinn4.color" -type double3 0 0 1 ;
setAttr "blinn4.transparency" -type double3 0.815686 0.815686 0.815686 ;
setAttr "blinn4.ambientColor" -type double3 0.172549 0.172549 0.172549 ;
// Make the camera
camera -centerOfInterest 5 -focalLength 35 -lensSqueezeRatio 1 -cameraScale
1 -horizontalFilmAperture 1.41732 -horizontalFilmOffset 0 -verticalFilmAperture
0.94488 -verticalFilmOffset 0 -filmFit Fill -overscan 1 -motionBlur 0 -shutterAngle
144 -nearClipPlane 0.01 -farClipPlane 1000 -orthographic 0 -orthographicWidth
30;
select -r camera1;
setAttr "camera1.translateX" 40;
setAttr "camera1.rotateY" 90;
select -cl ;
// Make the lights
defaultDirectionalLight(1, 1,1,1, "0", 0,0,0);
setAttr "directionalLight1.translateX" 12.7;
setAttr "directionalLight1.translateY" 4.7;
setAttr "directionalLight1.translateZ" 12.2;
setAttr "directionalLight1.rotateX" -13;
setAttr "directionalLight1.rotateY" 46.4;
setAttr "directionalLight1.scaleX" 4;
setAttr "directionalLight1.scaleY" 4;
select -cl ;
defaultDirectionalLight(1, 1,1,1, "0", 0,0,0);
setAttr "directionalLight2.translateX" 6;
setAttr "directionalLight2.translateY" 8.7;
setAttr "directionalLight2.translateZ" -15;
setAttr "directionalLight2.rotateX" 150.3;
setAttr "directionalLight2.rotateY" 10;
setAttr "directionalLight2.rotateZ" -180;
```

```
select -cl ;
// Save the current scene to a mayaBinary file
file -rename "eq_scene.mb";
file -save -type "mayaBinary";
```