Visualisation Course 2008 UG4/M.Sc.

Handout 2 – Creating a pipeline in VTK

In this exercise you will familiarise yourself with the stages required to create a visualisation pipeline in VTK. The pipeline you are going to create will display the bone surface of a knee joint from a CT scan of the visible woman dataset.

Refer to the VTK tutorial handout and associated URLs for assistance with this exercise.

Reading the data : The datafile from the CT scanner is located on the web page and is called `vw_knee.slc`. In order to read this file into VTK you will need an SLC file reader object called `vtkSLCReader` to read this volumetric file format.

Creating the Surface: We use the Marching Cubes algorithm to create the surface, covered later in the lectures. This algorithm is implemented by the `vtkContourFilter` object. Looking at the web page definition for this object you will see it has a method to specify the value of the created surface, try starting with 80 density units. The filter can create more than one contour surface, they are specified with an index before the contour value.

Extra objects: To finish the application you will need to add an actor, a mapper and a rendering window. The marching cubes algorithm produces triangular polygons in the same manner as the `vtkConeSource` object did in the previous exercise, so the same `PolyDataMapper`, actor and window objects can be used.

Linking together: Now that all the objects have been initialised, the pipeline can be set up. The syntax is the same as in the cone example – use the method `SetInput` with the substituted results from calling the object to connect with the method `GetOutput`. To connect a mapper to an actor use the method `SetMapper` in the actor, and similarly to add an actor into the renderer, use the method `AddActor`.

Setting properties in the actor: One thing you will notice when you build the pipeline is the hideous default colour of the surface. This is due to the mapper colouring the surface by its default red-blue colour table. In order to fix this you need to set the mapper to not colour its output surface and set the colour explicitly in the actor. Firstly use the method `ScalarVisibilityOff` in the mapper. Colours in VTK are set via the `vtkProperty` object. In order to access this object you need to call the actor with the method `GetProperty`. This will return an object which has a method `SetColor` which can be called with 3 floating point numbers describing red, green and blue colour values. Try setting the colour to something approximating bone.

Sub-sampling the data: The second thing you will notice after the hideous colour is the appallingly slow interaction and loading speed, especially if you're rendering in software. This is caused by the MarchingCubes object (`vtkContourFilter`) producing too many triangles for rendering at interactive rates. One way of fixing this is to sub-sample the original grid of points. VTK provides the object `vtkExtractVOI` to perform this task, use the method `SetSampleRate` with an appropriate factor, specified for each axis.

Tidying up: As a finishing touch add an outline round the data with a `vtkOutlineFilter` object. This

object takes as input the data and produces polydata which can be mapped using a `vtkPolyDataMapper` object. See the `VisQuad.tcl` example presented in lecture 3 for further details.

Finally note that variables can be created in TCL using `'set <varname> <value>'` and referenced using `$varname`. Try creating variables to control behaviour such as isosurface value and define their values at the start of the program. This will make your code easier to read and modify at a later stage. Alternatively, you could investigate how to set your variables from command line parameters to your VTK script (see TCL online command reference).

An alternative volume data file `neghip.slc` is also available on the course web page. For this file try a surface value of 0.5 with your VTK script.

Taku Komura 25/1/08(Based on earlier exercise by Gordon Watson)