Scientific Visualization – Exercise 2

DEADLINE: SUBMIT YOUR CODE + REPORT BEFORE MONDAY, NOVEMBER 17TH, 09:00.

Prerequisites

This assignment introduces you to the development of visualization programs using the Visualization Toolkit (VTK). It assumes you were present at the lecture introducing VTK. If not, go through the lecture slides on Blackboard.

VTK can be downloaded for free from http://www.vtk.org/ but it is also installed on the dual-boot PCs at Science Park (both for Windows and Linux).

This assignment consists of a hypothetical challenge that must be solved through visualization. Solve the challenge and submit a report that explains your approach. Submit your solution through Blackboard.

Submit one report per group pair. Make sure you mention both your names and student numbers in the report.

Introduction

Download the mystery dataset from http://bit.ly/1yY43gn

This is what we know: this dataset was produced by a CT ("Computed Tomography") scanner. This particular dataset was generated during the investigation of a crime scene where a body was found. A range of experts have attempted to determine the cause of death but none have succeeded. As a last resort, the investigators have asked a radiologist at an academic hospital to scan the body in order to look *inside* the patient to find a possible cause. Unfortunately, the radiologist who produced the scan has disappeared, so all you have is this dataset without any further explanation on its structure or contents. This is where you come in: it is your job to visualize the data and help the experts to determine the cause of death.

Assignment

Unzip the data and notice that there are 94 files, each 131,072 bytes (128kB) in size. From experience you know that CT scanners produce 3D image volumes as stacks of 2D images that are either 256 x 256 or 512 x 512 pixels each and where each pixel is represented as one scalar value. Knowing that; determine the resolution of each image and the size of each pixel (in bits). By the way: in general it is a good idea to assume that data is "unsigned" (i.e. without negative values), at least at first.

Source of the visualization pipeline

Create a VTK program and build the "source" stage of a visualization pipeline that reads in this dataset. For this you need to figure out the following:

1. Identify the correct VTK class capable of reading in a collection of binary images. There are several that can do this, but some have features that make it more suitable than others - see the VTK documentation for more details.
2. Configure this source stage by invoking the methods of this class that tell it the specifications of the input data:
    - dimensionality of each file (this is known: 2);

- number of images in the dataset (this is known: 94);
- resolution of each image (determined above)
- size / data type of each pixel (determined above);
- pattern of the filenames in the data set (think in the lines of a “printf” pattern).

Contour visualization

Your first impulse is to use isosurface contour visualization. For this you need to know the possible value range for the contour value, that is: the range in values in the input.

3. Force the source stage to read the dataset by calling its “Update()” method. Now figure out a way to ask the output of the source stage for the range of scalar values in the dataset.

Now complete the isosurface contour visualization pipeline:

4. Add VTK classes to create a visualization pipeline that renders an isosurface contour of the dataset. Make sure you add an interactor so that you can interact with the resulting visualization. For the initial contour value, use a value that is halfway the scalar value range. Or even better; allow this value to be passed from the commandline so that you can pass a different value each time you run your program.

You should see one solid object that you will probably recognize if you are familiar with human anatomy. If you see multiple objects, or if you do not recognize what you are seeing, you probably misconfigured the source stage of the pipeline.

Colours

In all likelihood your object will show up in a dark blue on a black background. The reason for the blue object is this:

- The contour filter by default passes scalar values along with the triangles that span the isosurface. These scalar values are used by the mapper to colour the triangles. In this case the contour filter passes the same value for all triangles: the contour value that was used to create the isosurface.
- By default the mapper maps scalars in the range from 0.0 – 1.0 to a red – blue lookup table (LUT), with values below 0.0 truncated to red, values over 1.0 truncated to blue. The contour value that you passed is higher than 1.0, therefore the object is blue.

Here are three ways to change the colour of the object. Experiment with all three and configure your visualization pipeline to create an effective colouring:

5. Tell the contour filter stage to *not* compute scalar values and then set a colour in the actor, or
6. Tell the mapper stage to ignore scalar values and then set a colour in the actor, or
7. Tell the mapper what the *actual* scalar range is. In this case the mapper will use a different colour from the red – blue lookup table depending on the contour value you used.

Changing the background colour from the default black is somewhat easier: see the documentation of the renderer.

8. Change the background colour. It is also possible to change the background to a gradient. It’s your choice. Just make sure it does not interfere with the colours in your visualization. While you are at it: increase the size of the render window as well.

Data spacing

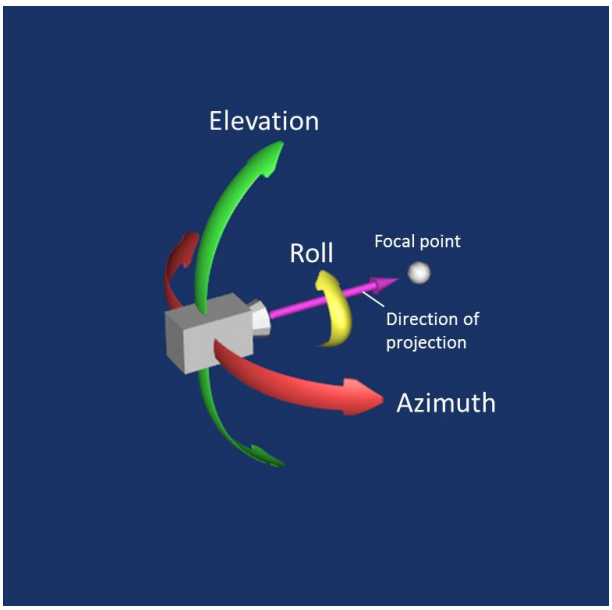
You will notice your object looks “compressed”. You might be tempted to fix this by altering the actor’s scaling. This will work, but the real reason is that in your source stage the “data spacing” has not been configured. This data spacing specifies the relative size of a voxel (volumetric pixel) in each direction. If you don’t specify the data spacing, the source stage assumes the voxels are “isotropic”: 1 × 1 × 1.

Normally you would be given these proportions but since the radiologist has disappeared, you will have to guess!

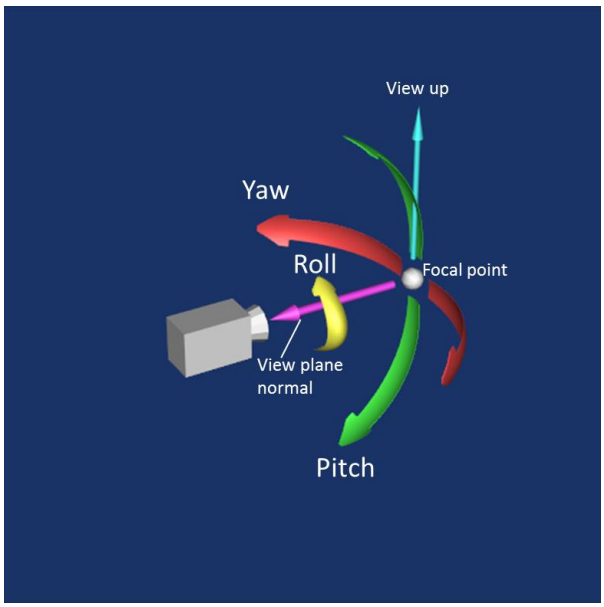
9. Experiment with the values for the data spacing and report on the numbers that you think look appropriate.

Setting the scene

When you run your program, the renderer will create a camera and configure it so that the visualization will be visible in the render window. However, in most cases this will be suboptimal; the initial camera angle will not provide a useful view on your visualization. In that case it will be necessary to manipulate the camera. VTK uses two models to manipulate the camera (see figure below).



Camera movement around focal point.



Camera movement centered at camera position.

In the model on the left, the camera is focused at a focal point (your visualization) and moves around this focal point using the Elevation, Roll and Azimuth methods. In the second model, the movement of the camera is centered at the position of the camera and the orientation of the camera is controlled using the Yaw, Roll and Pitch methods.

10. Modify your software in order to create a better camera angle on your visualization.
11. With this new camera angle, it becomes harder to determine the orientation of the visualization. Add an outline to your visualization that shows a bounding box to represent the extent of the input data.

Scanning the data

It is now time to put your visualization pipeline to good use as a forensic scanning tool. There are two features of this visualization method that you will exploit. First; isosurface contour visualization only represents an isosurface at the exact value that you specify in the contour filter. Second; CT is based on the fundamental principle that the density of the tissue passed by the X-ray beam can be measured from the calculation of the attenuation coefficient.[1] This means that areas of similar density will have similar values in the dataset.

12. Modify your software so that you can visually explore all contour values in the dataset. You could either use a simple loop to do this, or if you feel adventurous: add a user interface that supports a slider to set the contour value interactively. Whatever you use; scan through the visualizations to detect any anomalies in the data.
13. Finally; use the insights obtained from the previous step to generate one final image that clearly shows the anomaly/lies you found. Make sure the normal anatomy is also visible so that the anomaly/lies can be seen in the right context. In your report, describe your approach and your findings.

Good luck!

[1] http://radiopaedia.org/articles/computed-tomography