Visualisation Course 2008, UG4/M.Sc

Handout 3 – Creating images and animations

In this exercise you will see how to create images and animations using VTK. You will need to have completed the exercises on the earlier handouts in order to complete this.

Writing an image

To write an image in VTK, use the filter `vtkWindowToImageFilter`. The input to this filter is set to a `vktRenderWindow` directly, i.e:

```
vtkRenderWindow ren1
vtkWindowToImageFilter w2if
w2if SetInput ren1
```

We now create an image writer object, for instance `vtkJPEGWriter`. The writer is linked to the image filter in the usual way, and has a method `SetFileName` to set the name to save the file as.

To write the file call the method `Write`, however first you will need to call the `Render` method for the `RenderWindow`, and if you're doing an animation, you must call the modified method of the `ImageFilter` to update the filter first, otherwise it will only save the first frame.

Animating the model

In lecture 2 you were introduced to the for loop construct in TCL. With VTK this can be used to iterate over a set of visualisation commands to create an automated animation in the VTK window.

Try creating a loop to iterate over different density units (i.e. iso-surface value) or subsampling levels with the knee example from handout 2 or to change the polygon density of the cone from handout 1.

You may find the TCL command `after` useful in your loop. Usage as `after N` introduces an N millisecond delay into the TCL program at that point.

All variables in TCL are strings, in order to perform numerical operations on variables, remember to use the `expr` command, which evaluates numerical expressions. Trigonometric operators are also available if you want to try rotating the camera in a circle.

Animating the camera

The camera object can be retrieved by a method `GetActiveCamera` in the `vtkRenderer` object. Look at the VTK manual webpage for `vtkCamera` to see the available methods. Some of the more useful ones for animating simple camera paths are `Zoom`, `Dolly`, `Azimuth`, `Elevation`, and `Pitch`. Try these for yourself.

To rotate the model use the `Rotate [X,Y,Z]` methods in the `vtkActor` object.

Writing multiple image files

All that has to be done to write out a sequence of image files is to call the image writing methods at each iteration of a loop. Of course the files will need to be named sequentially. The TCL command `format` is useful for this and takes arguments in the same manner as printf in C (type `man format` in shell for more information, see TCL command reference manual). The first argument is a format string followed by the variables to be formatted. A string is output by the command that can be used as the filename.

Don't forget you must call the `Render` method in the `vtkRenderer` and the `Modify` method in the `vtkWindowToImageFilter` before writing out each frame! This is a consequence of the architecture of

VTK.

Creating an MPEG movie file

Now that all the JPG image files have been created, they can be made into a movie. In

```
/group/teaching/cs4/vis/MPEG
```

are a number of tools for displaying and creating MPEG files written by the media group at the University of California Berkeley. To create a movie file use the program mpeg_encode, and to display it, mpeg_play. The tricky part is setting the plethora of parameters required for an MPEG movie. In the directory you will find a template file for the encoder called `vtk.template`.

Copy this file to the directory where your sequence of image files is stored. You will need to edit this file to point to the image files that you have created. The file is quite well documented but specifically you need to edit the line:

```
file* [8-30]
```

where "file" is the starting pattern of your sequence files and the numbers in square brackets indicate the minimum and maximum sequence numbers to use (in terms of the file sequence).

Once edited, simply call the encoder with the name of the template:

```
mpeg_encode vtk.template
```

The template is set up to read JPEG files, but can easily be changed if required. Unfortunately the JPEG files produced by vtk by default are 'progressive JPEGs' and are not compatible with the rather old mpeg encoder, so the template is set up to convert the JPEG files into PPM before reading them in by running them through the 'djpeg' command. The resulting output will appear as an MPEG movie in the file `output.mpg`. This can be played with either the `/group/teaching/cs4/vis/MPEG/play_mpeg` or `gmplayer` commands.

Changing MPEG parameters

MPEG encoding is a complex process involving estimation of motion within the frame and re-use of parts of the images. Frames with no motion estimation are called I-frames, frames with motion estimation both forward and backwards (bi-directional) are called B-frames and P-frames are interpolated from those. The template has a line which controls the pattern of I,P and B frames in the movie.
You don't have to make any modifications to the default pattern specified in the template provided, but it is set up for fairly aggressive encoding so don't be surprised if the quality of your movies is quite poor. If you want to have a go at improving the quality of your MPEG file, you can modify the template to add more I-frames. This will also reduce coding time – but at the expense of a larger filesize.

Capturing interaction

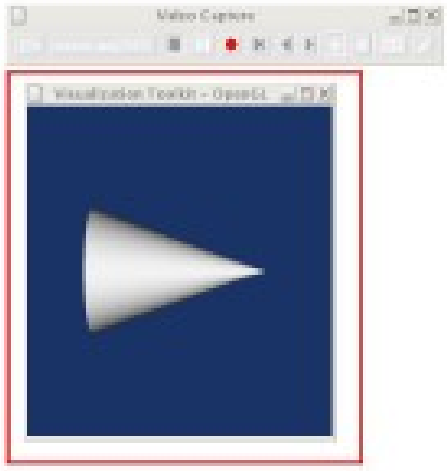
In addition to capturing a pre-defined VTK animation we can also capture user interaction with a VTK window using the `xvidcap` utility. This program records the contents of a specified screen area.

To use `xvidcap` first copy the (common) settings file to your home directory as follows (note the leading period "." in the second filename):

```
cp  /group/teaching/cs4/vis/xvidcap/xvidcap.scf ~/.xvidcap.scf
```

Run the command `/group/teaching/cs4/vis/xvidcap/xvidcap`

A window taskbar with an attached red box will appear. Position the window and box so that your VTK window is now inside this red box outline. The third icon from the right on the `xvidcap` taskbar allows you to increase/decrease the sizes of the box. e.g. :



1. Use the record button (red) to start recording.
2. Interact with the VTK window with the mouse.
3. Use the stop buttion (square) to stop recording.
4. File menu -> Quit

Resulting movie file stored as `output.mpg` in the current directory.

The format of the file `output.mpg` is MPEG-4. It can be played with `gmplayer` under DICE.

xvidcap can, of course, also be used to record model and camera animation but the initial window placement can be more difficult to achieve before the animation is complete!

xvidcap can be used to capture all or part of any window on the linux destop making it a very useful tool for capturing "canned" demos instead of relying on a live event. File menu->options gives a number of options to change video quality and alike. See http://xvidcap.sourceforge.net/ for further information.

Taku Komura (based on a previous exercise by Gordon Watson)