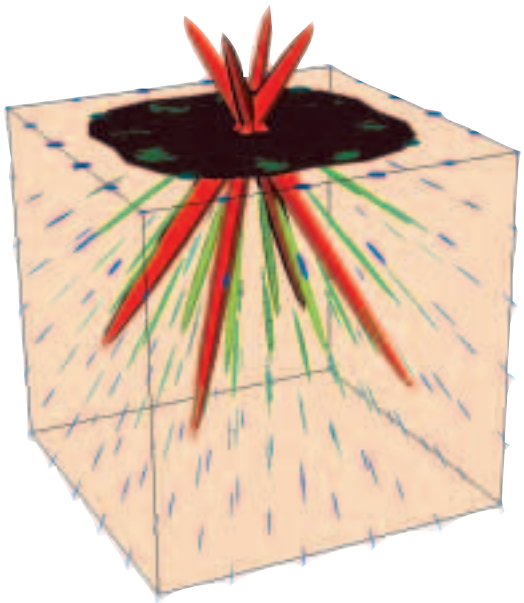
# Scientific visualisation with VTK and Tcl

# EYECATCHER

**Graphical visualisation of complex data is no problem with the VTK library. This turns mountains of multi-dimensional data into clear images – and as Carsten Zerbst explains, it can be programmed using Tcl/Tk**



As the saying goes, a picture is worth a thousand words, and with good reason: visualisation is helpful in understanding complex correlations. For 2D data there are several tools available under Linux, such as Gnuplot, Grace of ScigrAphica. For anyone who needs more, a suitable free software solution is also available in the form of the Visualization Tool Kit (VTK). With its many display variants, VTK produces meaningful 3D graphics from multi-dimensional measurement data or calculations.

VTK is not an independent tool like Gnuplot, but rather a class library that handles both visualisation and image processing. This library, written in C++, can also be used with Tcl, Python and Java and its vast array of options offers solutions for any requirements. The sources are available at *http://public.kitware.com* and SuSE also includes VTK as an RPM. The demo programs and documentation files are also recommended.

The library, language bindings and other files amount to a whopping 800Mb. If your space is limited, however, you can restrict yourself to the languages you really need and install only a selection of the associated files.

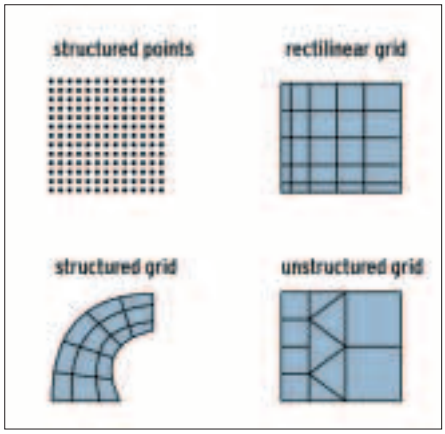



**Figure 1: Possible topologies of structured points. Each point relates to one or more measurement values that apply to this location**

The many demo programs are worth their weight in gold and you can often solve your own problems just by looking through them. The VTK book by the library's authors is also recommended for anyone seriously interested in the subject.

VTK originated in the medical science division of General Electric but has been under Open Source for many years. Consequently it has a lively user community with an active mailing list, and Sebastian Barré has compiled a useful selection of links. Some of the algorithms contained in VTK are patented, however: in their FAQ the authors point out that commercial use may incur license fees.

## Another field

Visualisation is based on data from calculations or measurements available for individual structured points. In VTK these points are called a dataset. The topology (arrangement) of these points can take various forms, depending on the methods used for the measurements or calculations. Finite element method calculations normally use an unstructured mesh, while computer tomographs measure points on a fixed grid. Figure 1 shows some of the common variants. Data exists for each of these structured points, these can be scalar values (such as temperature, density), vector values (displacement, flow speeds) or tensors (tension). Visualisation is much more than just the representation of forms.

While CAD systems simply need to represent the geometry, things are more complicated when visualising measurements. Here, the aim is to render a mass of data in a suitable format through the use of various techniques. This means that the character of the data will sometimes have to be changed completely in order to achieve a comprehensible result. We will try to clarify this with the help of some examples.

The magnification function (also referred to as amplitude frequency) of a simple oscillator depending on harmonisation and absorption (Figure 2) is a pretty simple task for VTK. Here an area is mounted and coloured in according to the formula.

## Tcl update

Another two months have passed in Tcl land. The current release 8.3 fixed some errors in the Mac port and then 8.3.4 came out at the end of October. Apple is supporting Jim Ingham in a native port of Tk to Mac OS X; the result is going to be available in 8.4. Although the developers are adding more and more functionality to 8.4 they seem to be lacking the will to finally release it.

Aside from a lot of background work, the Tcl Core Team (TCT) has decided to integrate two new widgets: a frame with label by Peter Spjuth and a paned window by Eric Melski. Both widgets will be contained in Tcl/Tk 8.4. Although the new version is entirely useable by now, it is officially still in its alpha stage. Thanks to CVS access at SourceForge, the wait for the next edition has lost some of its horror, but a crowd-pleaser in form of the completion of 8.4 would be even better.

Away from those little steps another decision is much more important: TIP (Tcl Improvement Proposal) number 50 has been signed off by the TCT. This is the proposal to deliver the OO extension [incr Tcl] together with the core release. That makes classes and inheritance available as part of the normal Tcl distribution from Tcl 8.4 (similar to Tk at the moment). A set of additional widgets, such as calendar, a progress display and many others, is also part of [incr Tcl].

Thread extensions are already available prior to the release of 8.4. Tcl itself has long been thread-enabled but this was mainly used from C, in applications with embedded interpreters. At SourceForge you can now get extensions, which enable the script side to use several threads as well. For Tcl this marks another step away from its father John Ousterhout, who thinks that threads are generally a bad idea.

Whatever you think of .NET, the SOAP protocol is getting more and more support under Linux as well as elsewhere. TclSOAP can be used to develop servers and clients for the SOAP protocol. The extension uses TclDOM and TclXML and has been written entirely in Tcl. TclSOAP 1.6 implements SOAP 1.1 and fulfils the Userland test suites.

Csaba Nemethi has introduced new versions of his Tcl extensions. All three packages have been written in Tcl and can therefore be used without compiling independent of platform. The widget callback and multi entry packages help with those niggling little user entry problems. These packages can reduce possible entries to a particular format (date, for instance), Ethernet address or telephone number. The multi column listbox is a feature that is sadly lacking in Tk at the moment. It is easily adapted to your own requirements, be they sorting functions or special colours, right down to individual cells.

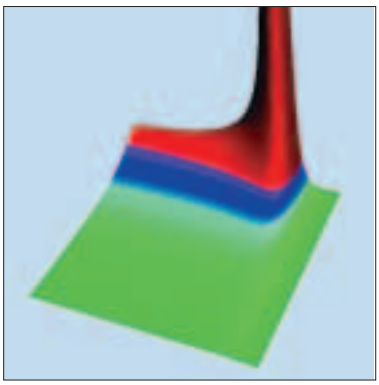


**Figure 2: The magnification function of a simple oscillator. The colour (and height) represents the extent to which the oscillator reacts to a stimulus, depending on its harmonisation and absorption**

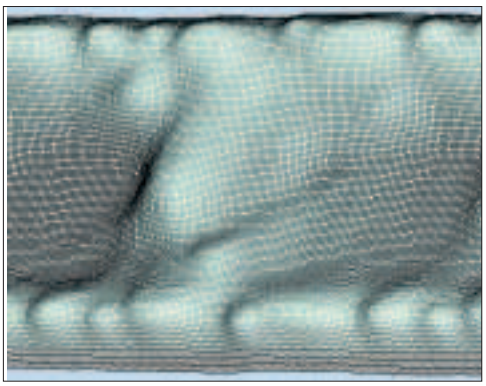


**Figure 3: The Tux Racer course as a grid: for each of the 80,000 structured points information exists on the height of the landscape at this point**

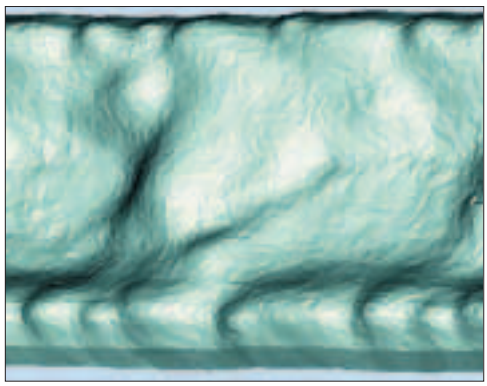


**Figure 4: The Tux Racer course, but this time represented only by a few triangles. The graphic appears almost as detailed, although it contains significantly less information.**

## Off the straight and narrow

The Bumpy Hill course from the Tux Racer game is a bit more demanding. It is based on a bitmap with a height profile, which provides height information for every point of the landscape. In the first step the profile is represented as a grid (Figure 3). This grid contains 80,000 rectangles, far too many to handle easily. This is a common visualisation problem, the original volume of the data is simply too large. In Figure 4 the number of elements has been reduced significantly, although the visual appearance remains much the same, thanks to intelligent decimation techniques.

In our last example Figure 5 shows a human skull. This is based on a file with density values from a computer tomograph. Using the marching cubes method, a surface consisting of triangles is calculated from these. The surface covers the points whose density corresponds to that of bones. This changes the topology of the data fundamentally: the structured (3D) density value grid is turned into an area consisting of polygons.

## The toolbox

To be able to keep track of all these conversions and renderings VTK splits them into separate steps, with each step implementing its own object. The idea behind this is a flow of data from source to rendering. The individual objects modify this stream of data until the dataset (measurement values) finally

**Figure 5: The surface of the skull was calculated from density values measured with a computer tomograph. While the data is provided in form of a volume grid, the surface consists of polygons.**

## Listing 1: Pressure distribution

```
01 #!/bin/sh
02 # \
03 exec vtk "$0" "$@"
04
05 # read in data
06 vtkStructuredPointsReader reader
07 reader SetFileName "press03"
08 reader SetScalarsName "pressure"
09
10 # create surface for values ➥
of 1.0
11 vtkMarchingCubes iso
12 iso SetInput [reader GetOutput]
13 iso SetValue 0 1.0
14
15 # map model to graphical ➥
primitives
16 vtkPolyDataMapper isoMapper
17 isoMapper SetInput [iso ➥
GetOutput]
18 isoMapper ScalarVisibilityOff
19
20 # the actor
21 vtkActor isoActor
22 isoActor SetMapper isoMapper
23
24 # colour surface seagreen
25 set isoProp [isoActor ➥
GetProperty]
26 # X11 colour sea_green_light
27 $isoProp SetColor 0.1255 ➥
0.6980 0.6667
28 $isoProp SetAmbient 0.4
29
30 # create renderer and window
31 vtkRenderer ren1
32 ren1 AddActor isoActor
33 ren1 SetBackground 1 1 1
34
35 vtkRenderWindow renWin
36 renWin SetSize 600 480
37 renWin AddRenderer ren1
38
39 vtkRenderWindowInteractor iren
40 iren SetRenderWindow renWin
41
42 # start representation
43 renWin Render
```

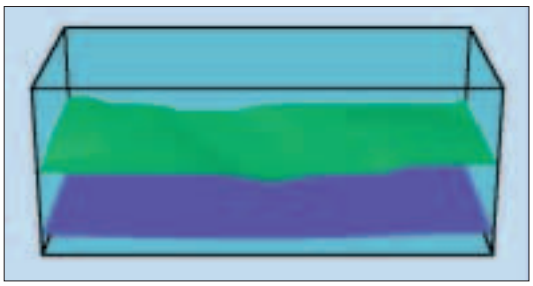


**Figure 6: Example output: pressure distribution in high seas. The image shows the wave surface, which is situated where the pressure is at 1hPa.**

turns into the required rendition. It is the vast number of possible intermediate steps and their interrelations that make VTK so powerful.

How these VTK objects can be used in Tcl is again best illustrated using an example. It is based on a file containing pressure values in high seas. The aim is to calculate the wave surface and to render it as a simple image.

As a first step, the measurement values must find their way into VTK. The data can be read with one of the various reader classes. Apart from ones for VTK's own format these also exist for many other 3D formats (e.g. 3D Studio, VRML, PLOT3D, BYU, SLC, STL) as well as for bitmap formats.

The measurement values are normally in the form of a grid. The location of the wave surface is where the pressure is 1hPa, only this surface is to be displayed. This is achieved using the marching cubes method, which calculates triangular surfaces from fields of scalar values. Listing 1 shows in detail how the VTK pipeline is constructed; the result is can be seen in Figure 6.

## From grid to wave

VTK is going to calculate the wave surface from the raw data contained in the file press03. The conversion is handled by the object iso, which is of the type vtkMarchingCubes and which implements the algorithm of the same name. This object is allocated the output of the read object reader as input using the method SetInput. All VTK classes use this mechanism for implementing the stream of data.

Once the area has been calculated it needs to be mapped to graphical primitives (for example points, lines or triangles) for rendering. This is done using mapper objects, which represent the geometry of the model. The mapper isoMapper is linked to the output of the iso object and thus integrated into the data stream.

Both geometry and colour are then represented by an actor. To give the whole thing a maritime touch the area is going to be displayed in sea green. Part of every actor is a property object for colours and ambience. This object can be returned using the method GetProperty, and is to be stored by our example program in the isoProp variable. The method SetColor then sets the colour as requested, while the SetAmbience method changes the lighting characteristics.

There are still two steps missing from the pipeline before we get to rendering: a window which brings the graph to the screen and before that an object that calculates what is to be rendered. Calculation is handled by the render class, and an object from this class is linked to the actors. Here, the procedure is slightly different from before: instead of linking input and output, the AddActor method tells the renderer which actor to render.

## VTK, windows and Tk

Now we just need the output window. VTK recognises several window classes, in this case we are using vtkRenderWindow. You can also zoom in on or out of graphs and turn or move them within the window using the vtkRenderWindowInteractor object.

Apart from vtkRenderWindow there is also a vtkTkRenderWidget. This behaves like a normal Tk

## Info

| | |
|---|---|
| VTK sources | *http://public.kitware.com* |
| Manual | *ftp://public.kitware.com/pub/vtk/nightly/vtkMan.tar.gz* |
| S. BarrÈ | *http://www.barre.nom.fr/vtk/links.html* |
| VTK pipeline | *http://brighton.ncsa.uiuc.edu/prajlich/vtkPipeline/* |
| TclSOAP | *http://tclsoap.sourceforge.net/* |
| Csaba Nemethi | *http://www.nemethi.de* |
| Thread Extension | *http://sourceforge.net/projects/tcl/* |
| John Ousterhout | *http://home.pacbell.net/ouster/threads.ppt* |
| Will Schroeder, Ken Martin and Bill Lorensen – The Visualization Toolkit (Prentice Hall, 1997) | |


**Figure 7: Decimator is a display utility for 3D formats. It uses Tcl/Tk for its GUI and VTK for 3D rendering**

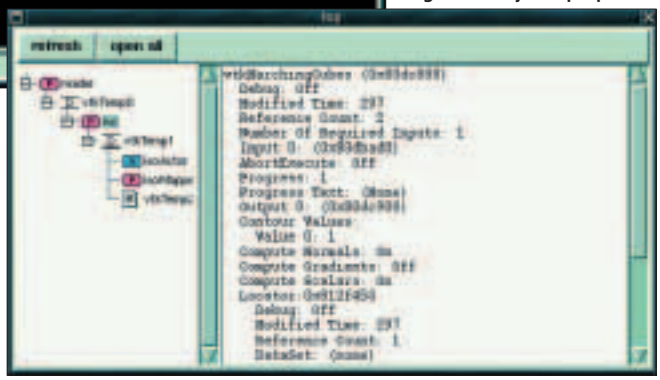



**Figure 8: ctkPipeline illustrates the data flow of our example program. On the left it shows the objects and their relations, on the right the objects' properties**

widget; it allows you to construct the entire surface with Tk and to use 3D visualisation as a sort of 3D canvas. A good example for the integration of Tk and VTK is Decimator, a utility for displaying 3D formats like BYU, STL and Wavefront (Figure 7).

Now that the entire pipeline from file to window has been linked, renWinRender starts the data flow. Instead of a confusing mass of data we are presented with the desired image.

The vtkPipeline utility can prove very useful: it represents the pipeline structure graphically and provides a good insight into its interdependencies and processes (Figure 8). The reader transfers the raw data via the connection vtkTemp0 to the iso object. This is the object selected for rendering in our program – consequently the right half of the window shows the name of its class (i.e. vtkMarchingCubes) and its properties.

## Conclusion

Once you have familiarised yourself with its pipeline structure and its many options VTK offers a pretty quick route to producing images. In combination with Tcl, visualisation programs can be created quickly, and with user-friendly interfaces to boot, thanks to Tk.

## The author

Carsten Zerbst works for Atlantec on a specialised PDM-System for the ship-building industry. Apart from that he devotes his time to the general application of Tcl/Tk.