Scientific visualisation in 2D and 3D

Roman Putanowicz
R.Putanowicz@L5.pk.edu.pl

Right tool for the right job

- gnuplot – All sorts of 2D visualisation and simple 3D visualisation
- octave – More advanced 2D visualisations that require extensive data manipulations
- octave + octaviz – Simple 3D programming
- VTK – Advanced computer graphics and 3D programming
- ParaView, OpenDX – 3D visualisations

Part I

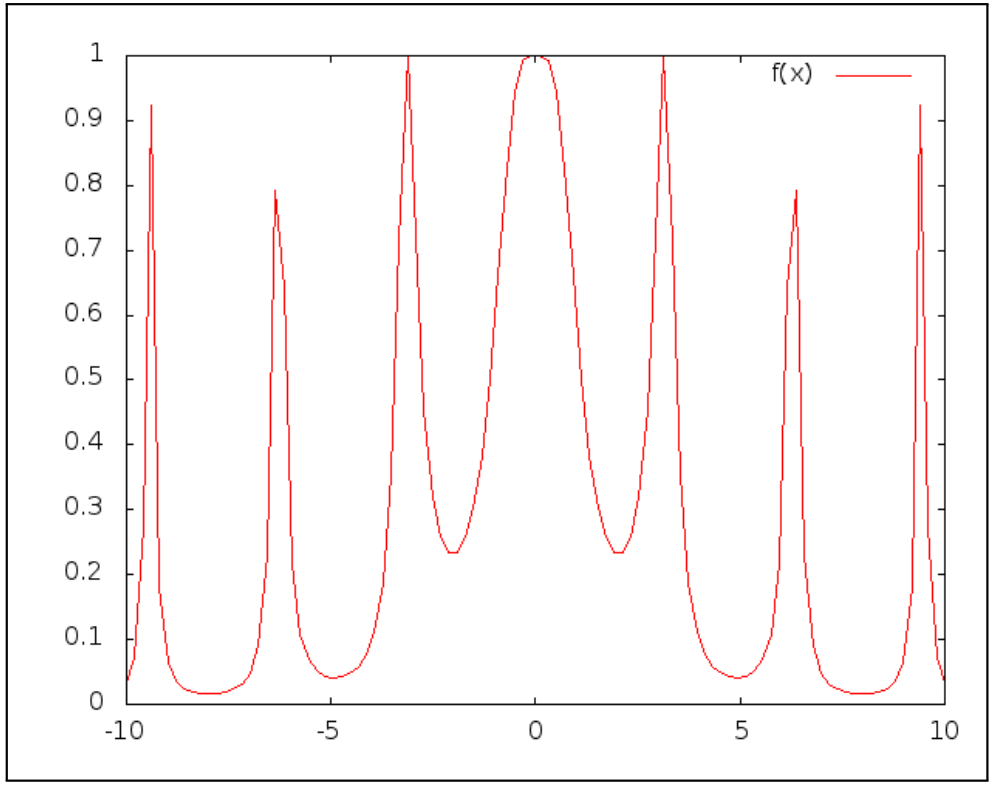
Gnuplot Overview

Gnuplot

- Home page: `http://www.gnuplot.info/`
- Portable: MS Windows, GNU/Linux, UNIX, and OSX.
- Book: "Gnuplot in Action. Understanding Data with Graphs", Philipp K. Janert, Manning Publications Co., 2009

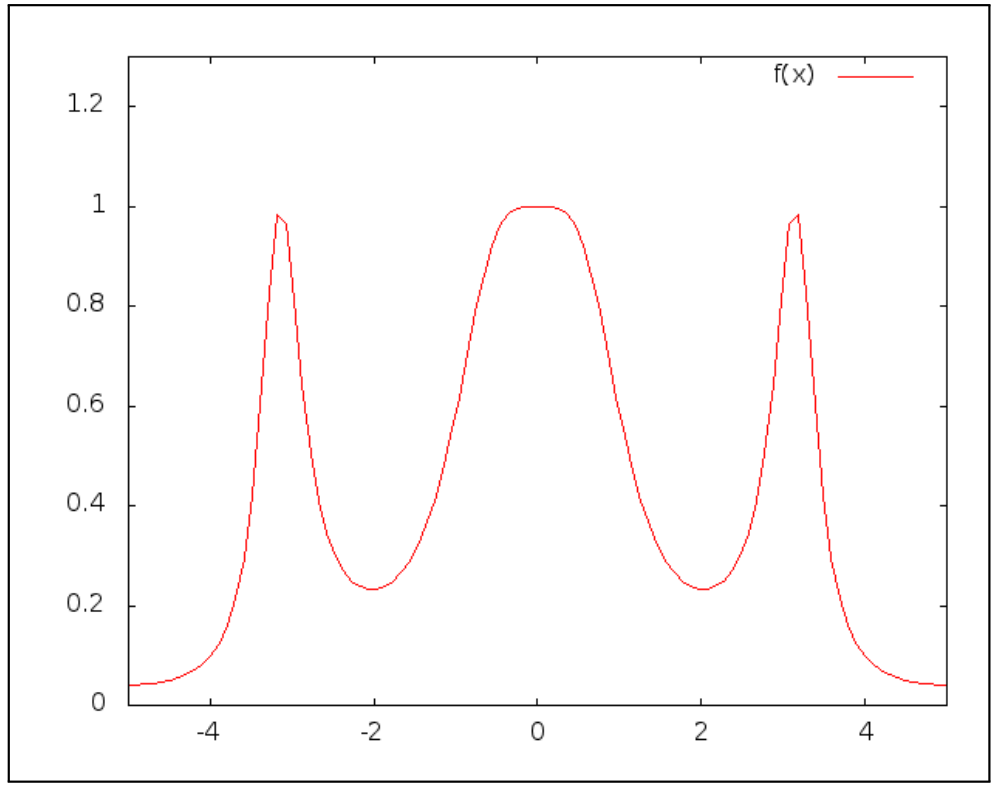
Plotting functions given by formulas

```
1  f(x) = 1/(1+(x*sin(x))**2)
2  plot f(x)
```



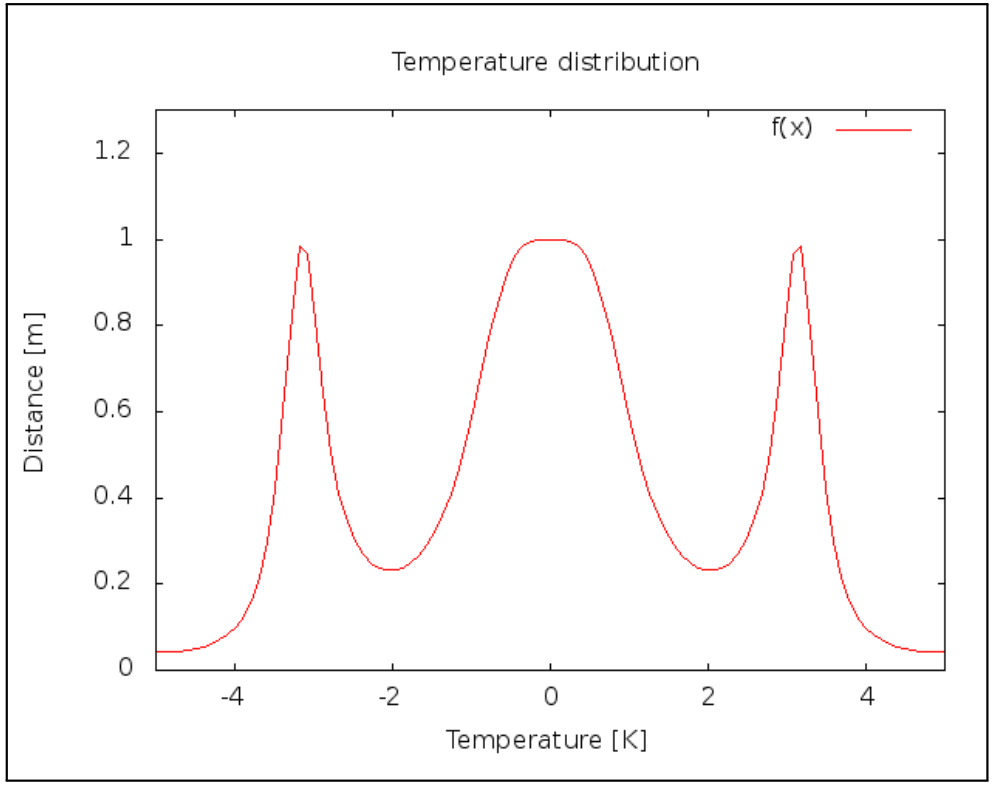
Axes setup

```
1  f(x) = 1/(1+(x*sin(x))**2)
2  plot f(x)
3  set xrange[-5:5]
4  set yrange[0:1.3]
5  replot
```

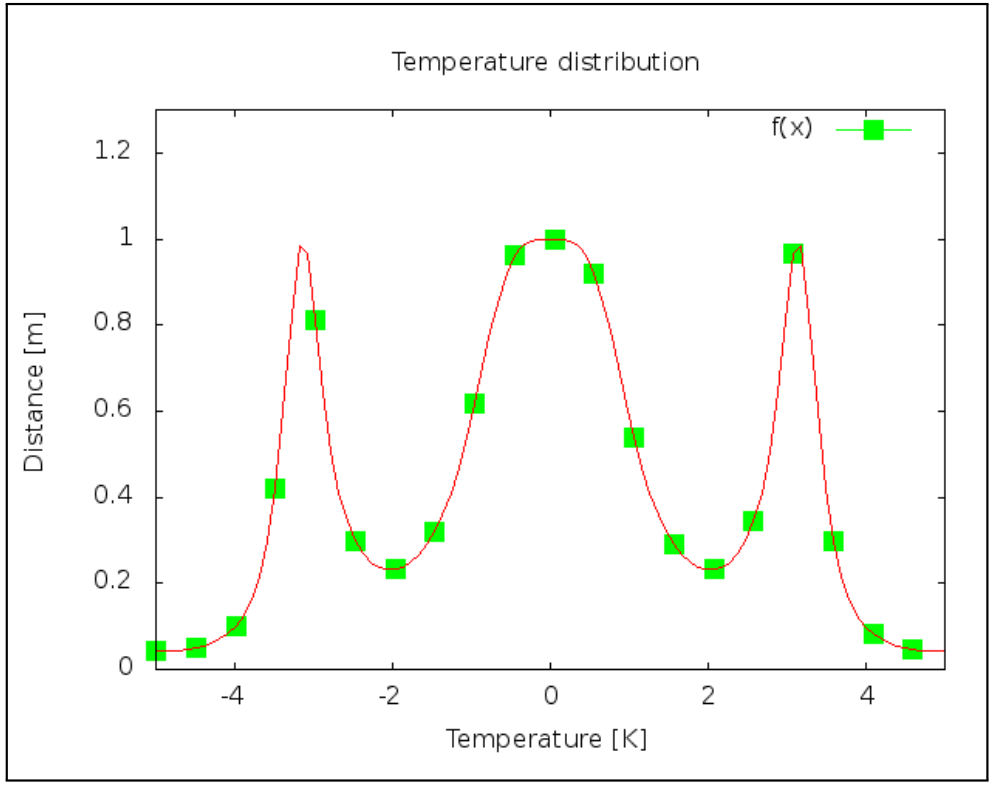


Labels and graph tytle

```
1  f(x) = 1/(1+(x*sin(x))**2)
2  plot f(x)
3  set xrange[-5:5]
4  set yrange[0:1.3]
5  set xlabel "Temperature [K]"
6  set ylabel "Distance [m]"
7  set title \
8  "Temperature distribution"
9  replot
```

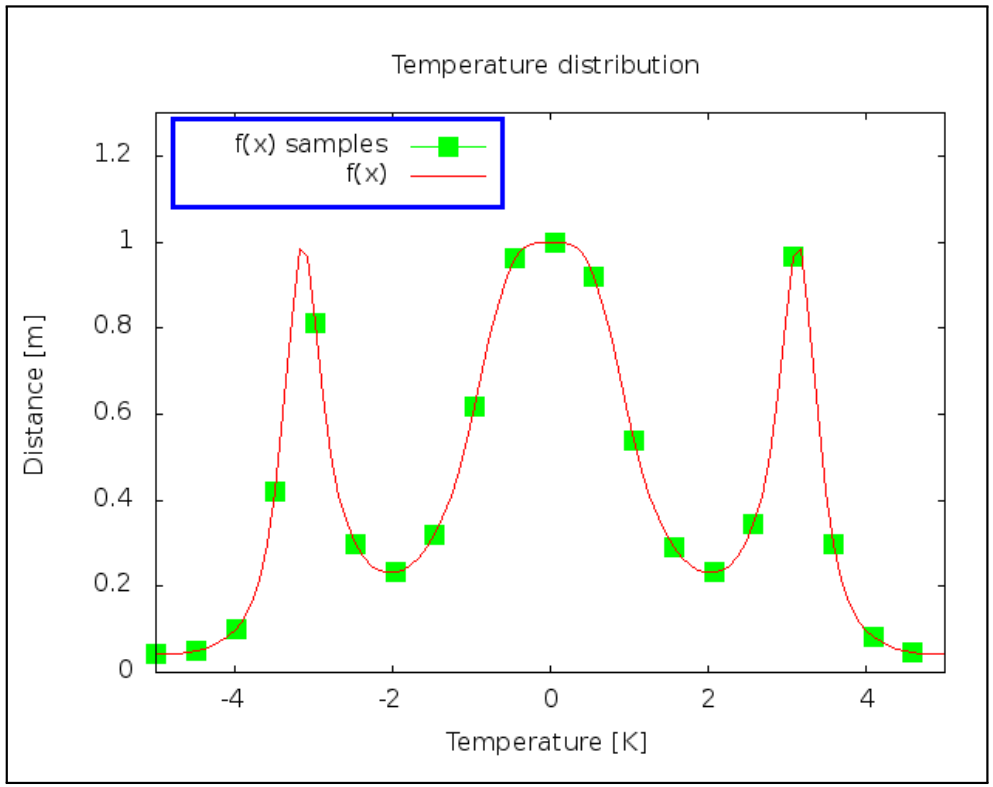


Setting lines and markers style




```
1  f(x) = 1/(1+(x*sin(x))**2)
2  set xrange[-5:5]
3  set yrange[0:1.3]
4  set xlabel "Temperature [K]"
5  set ylabel "Distance [m]"
6  set title \
7  "Temperature distribution"
8  set style lines 1 lw 0 linecolor rgb "green" pt 5 ps 2 pointinterval 5
9  set style lines 2 lc rgb "red"
10 plot f(x) with linespoints ls 1, f(x) notitle ls 2
```

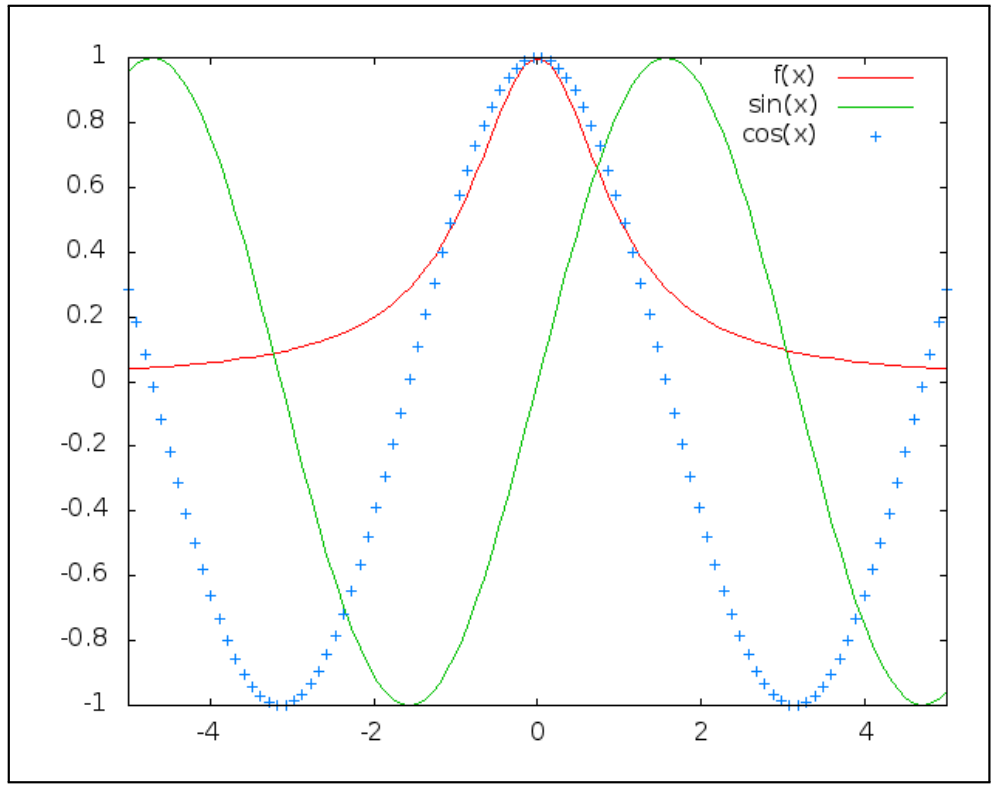
Legend



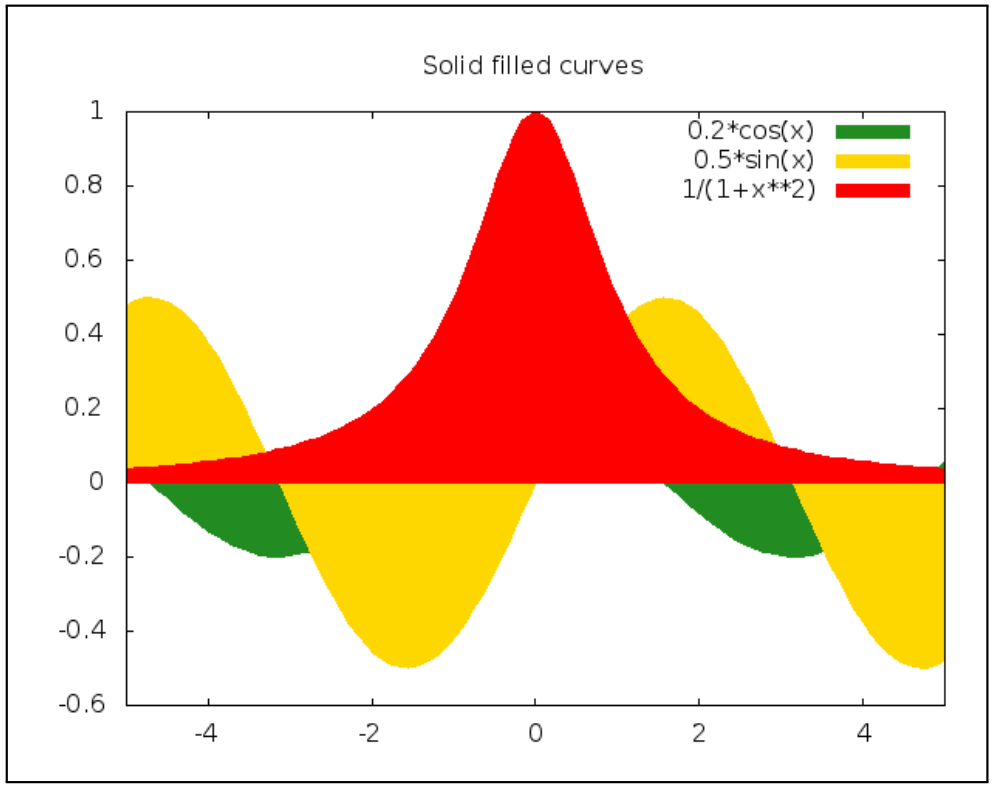

```
1   f(x) = 1/(1+(x*sin(x))**2)
2   set xrange[-5:5]
3   set yrange[0:1.3]
4   set xlabel "Temperature [K]"
5   set ylabel "Distance [m]"
6   set title \
7   "Temperature distribution"
8   set style lines 1 lc rgb "red"
9   set style lines 2 lw 0 linecolor rgb "green" pt 5 ps 2 pointinterval 5
10  set style lines 3 linecolor rgb "blue" lw 3
11  set key left width 1 height 1 box ls 3
12  plot f(x) w linespoint title "f(x) samples" ls 2, f(x) ls 1
```

Plotting many functions

```
1  f(x) = 1/(1+x**2)
2  set xrange[-5:5]
3  plot f(x), sin(x)
4  replot cos(x) w points
```

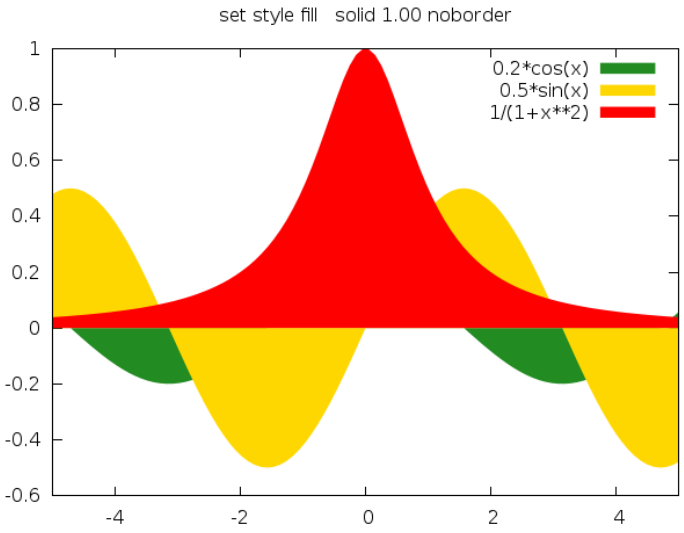


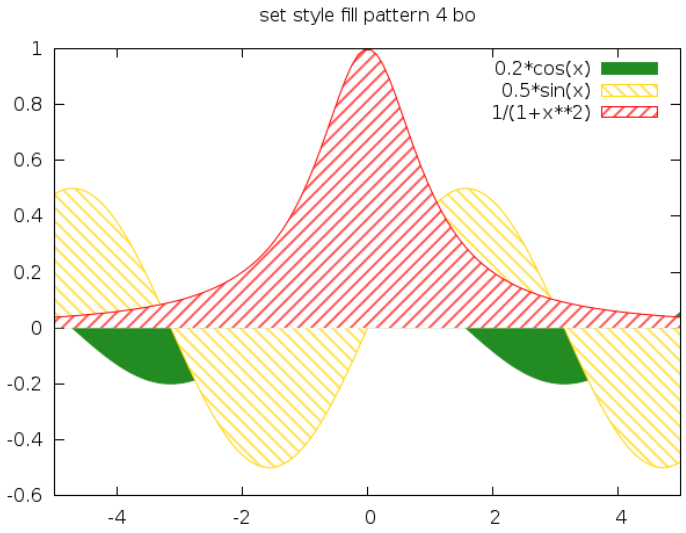
Filled functions

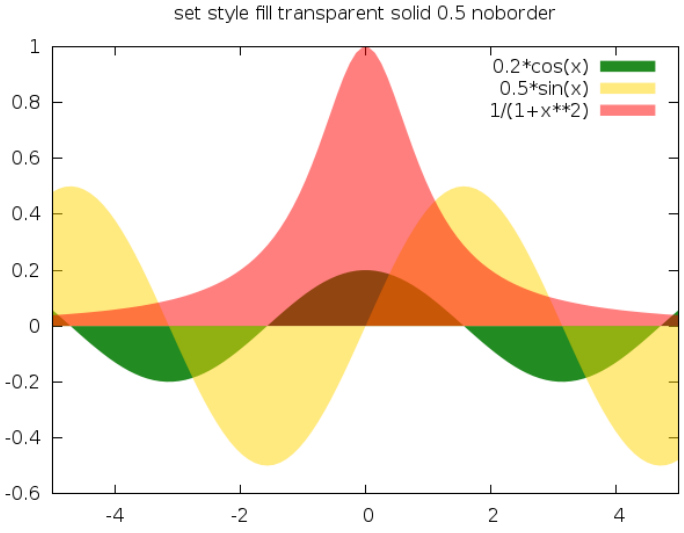


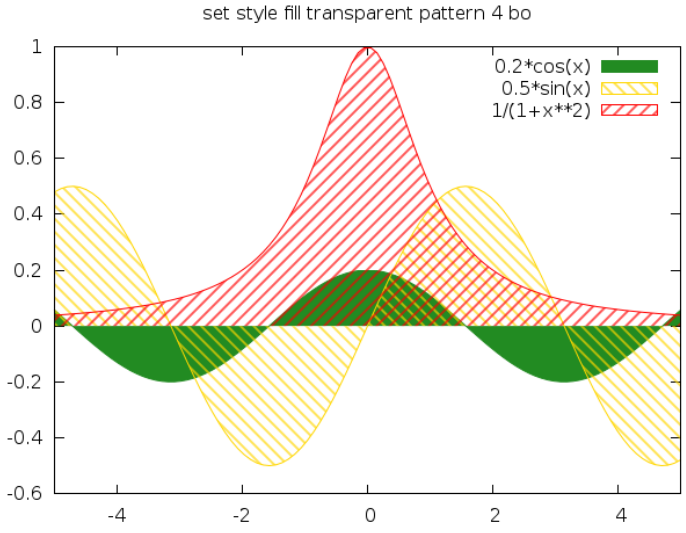

```
1   f1(x) = 0.2*cos(x)
2   f2(x) = 0.5*sin(x)
3   f3(x) = 1/(1+x**2)
4   t1 = "0.2*cos(x)"
5   t2 = "0.5*sin(x)"
6   t3 = "1/(1+x**2)"
7   set clip two
8   unset colorbox
9   set xrange [-5:5]
10  set style fill
    solid 1.00 noborder
11  set style function filledcurves y1=0
12  set title "Solid filled curves"
13  plot 0.2*cos(x) fs solid 1.0 lc rgb "forest-green" title t1
14  replot 0.5*sin(x) lc rgb "gold" title t2
15  replot 1/(1+x**2) lc rgb "red" title t3
```

Transparent filling









Saving figures

Selected formats:

Nazwa	Description
canvas	HTML Canvas object
cgm	Computer Graphics Metafile
corel	EPS format for CorelDRAW
dumb	ascii art for anything that prints text
dxf	dxf-file for AutoCad (default size 120x80)
fig	FIG graphics language for XFIG graphics editor
gif	GIF images using libgd and TrueType fonts
jpeg	JPEG images using libgd and TrueType fonts
latex	LaTeX picture environment
pdfcairo	pdf terminal based on cairo
png	PNG images using libgd and TrueType fonts
pngcairo	png terminal based on cairo
postscript	PostScript graphics, including EPSF embedded files (*.eps)
svg	W3C Scalable Vector Graphics driver
wxt	wxWidgets cross-platform windowed terminal
x11	X11 Window System

Saving in vector formats

fig

```
gnuplot> set output "rys.fig"
gnuplot> set term fig
Terminal type set to 'fig'
Options are 'color small pointsmax 1000 landscape inches \
    dashed textnormal font "Times Roman" fontsize 10 linewidth 1\
    depth 10 version 3.2'
```

PostScript

```
gnuplot> set output "rys.ps"
gnuplot> set term postscript color
Terminal type set to 'postscript'
Options are 'landscape noenhanced defaultplex leveldefault color \
    colortext dashed dashlength 1.0 linewidth 1.0 butt noclip \
    palfuncparam 2000,0.003 "Helvetica" 14
```

Saving in raster formats

GIF

```
gnuplot> set output "rys.gif"
gnuplot> set term gif
Terminal type set to 'gif'
Options are 'truecolor nocrop \
font /usr/share/fonts/truetype/ttf-liberation/LiberationSans-Regular.ttf 12\
 size 640,480 '
```

PNG

```
gnuplot> set output "rys.png"
gnuplot> set term png
Terminal type set to 'png'
Options are 'truecolor nocrop
font /usr/share/fonts/truetype/ttf-liberation/LiberationSans-Regular.ttf 12\
size 640,480 '
```

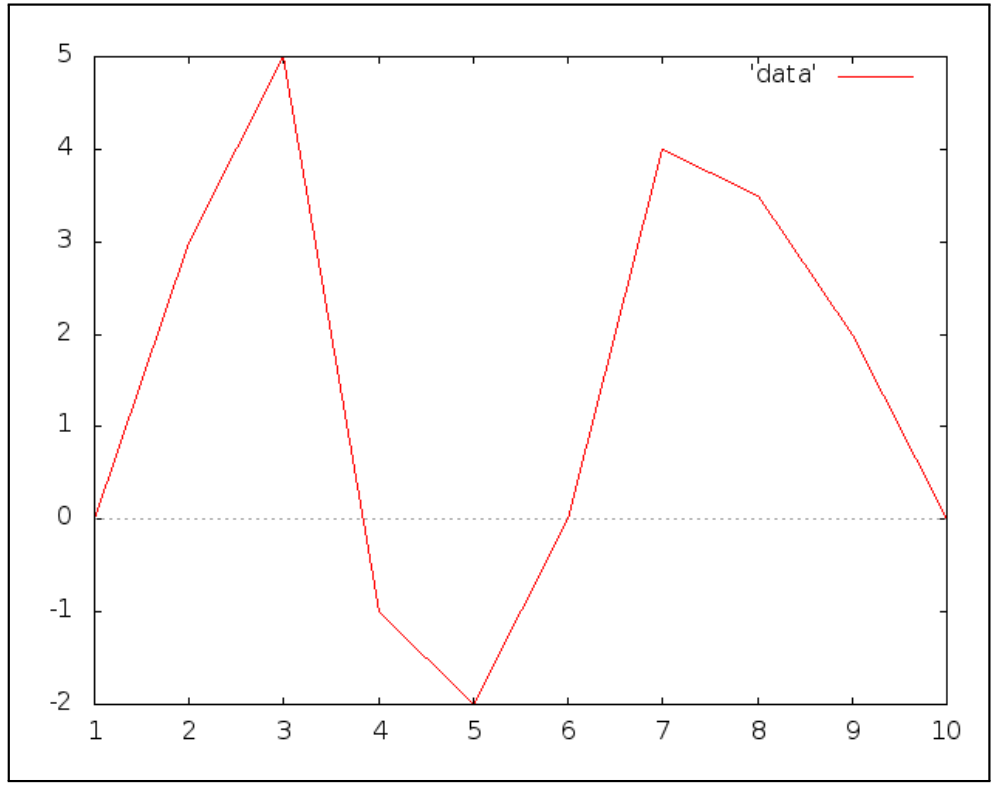
Data visualisation

Data file:

```
1  1 0
2  2 3
3  3 5
4  4 -1
5  5 -2
6  6 0
7  7 4
8  8 3.5
9  9 2
10 10 0
```

Script:

```
1  set xzeroaxis
2  plot 'data' w l
```



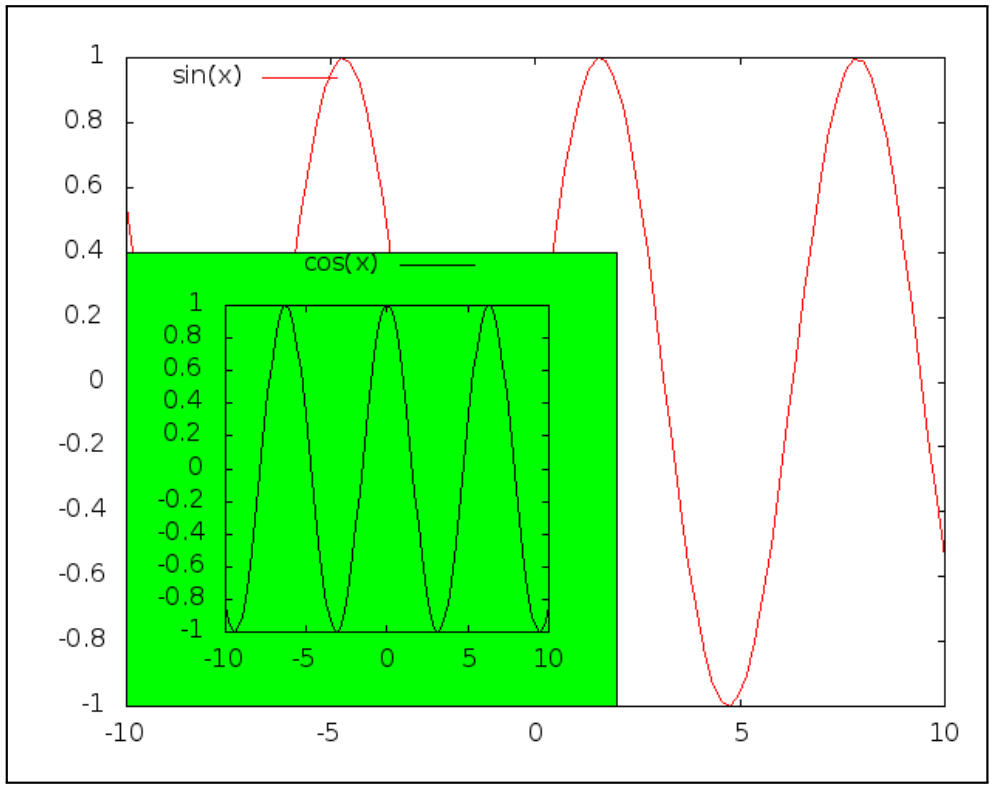
Operation on data stream and data filtering

Gnuplot provides tools for filtering input data stream and selecting data to be plotted.

- selecting data records – option `every`
- selecting data columns – option `using`
- selecting data sets – option `index`
- data interpolation and approximation – option `smooth`
- filtering through external programs

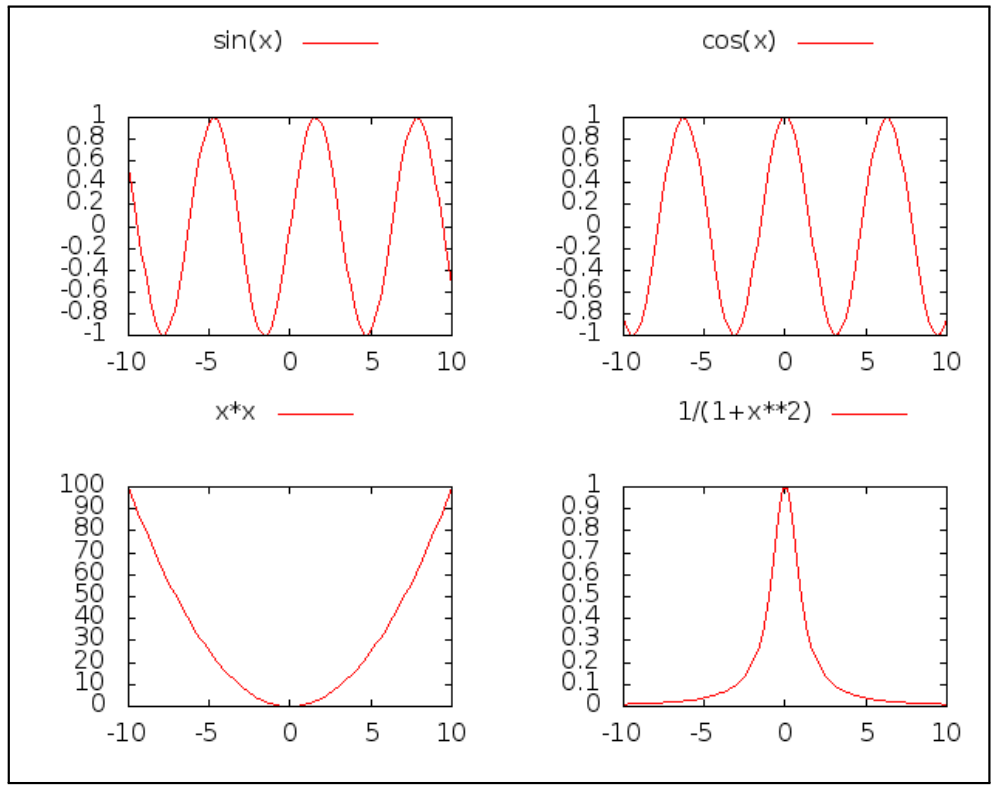
Overlays

```
1  set object 1 rect
   from graph 0, 0 to graph 0.6,
   front fc rgb "green"
2  set key left
3  set multiplot
4  plot sin(x)
5  unset object 1
6  set key outside center top
7  set size 0.5,0.6
8  set origin 0.1 , 0.1
9  plot cos(x) lc rgb "black"
```



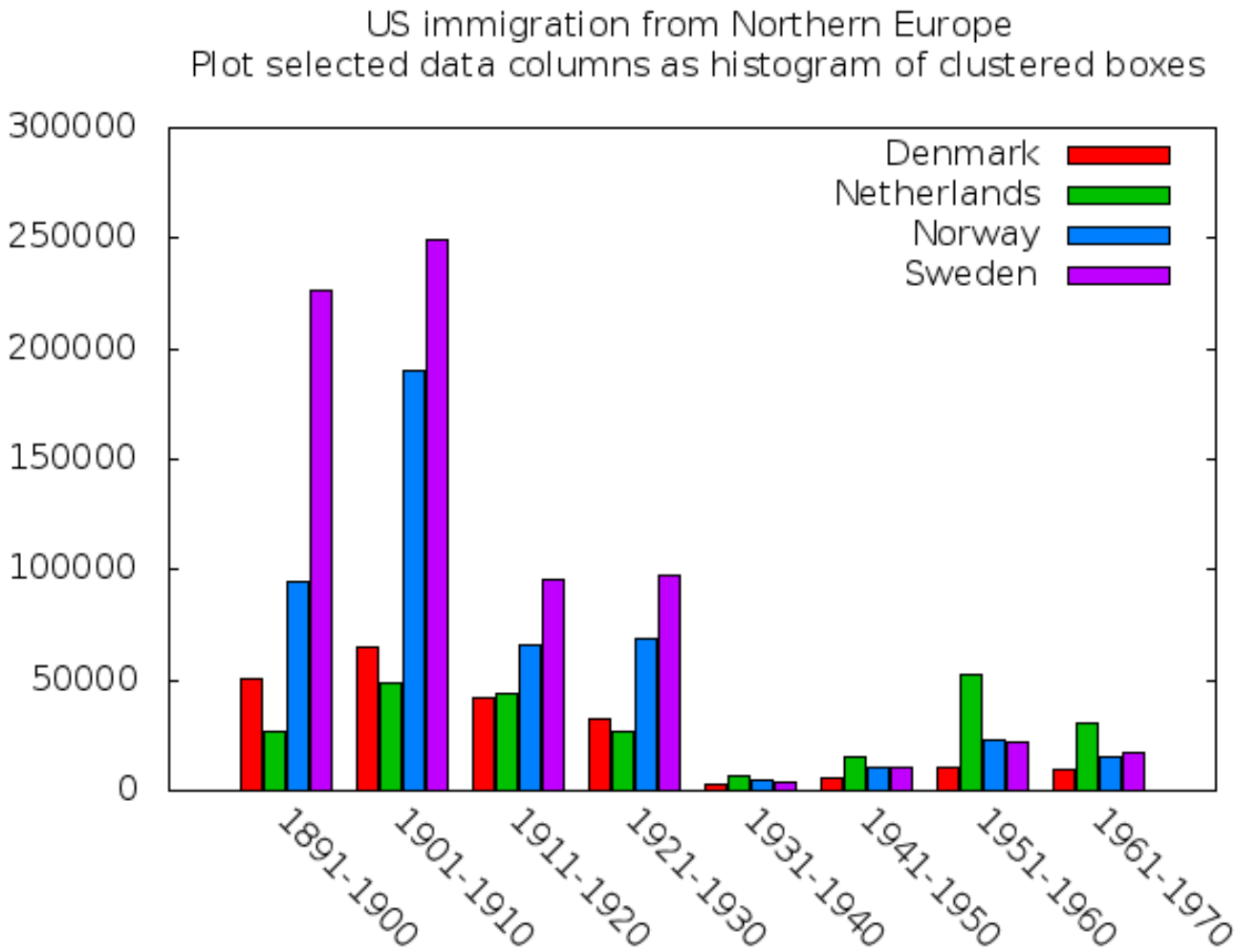
Subwindows

```
1  set multiplot layout 2, 2
2  set key outside center top
3  plot sin(x)
4  plot cos(x)
5  plot x*x
6  plot 1/(1+x**2)
```



Histograms

Script and data taken form gnuplot distribution.

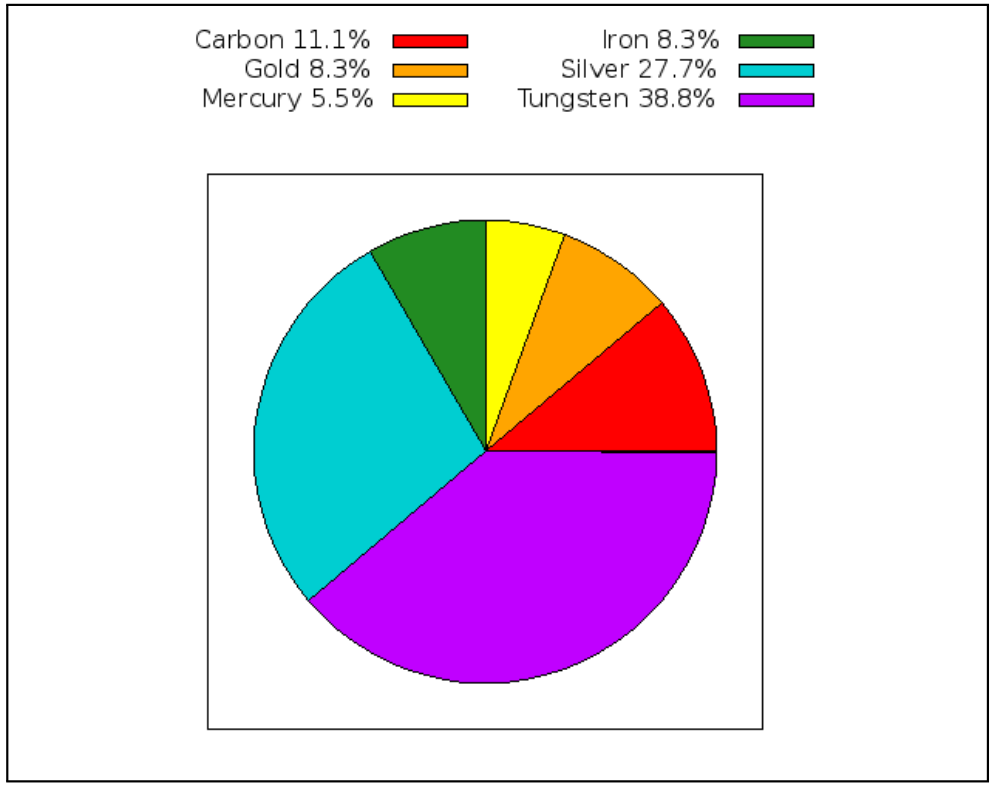


Histograms

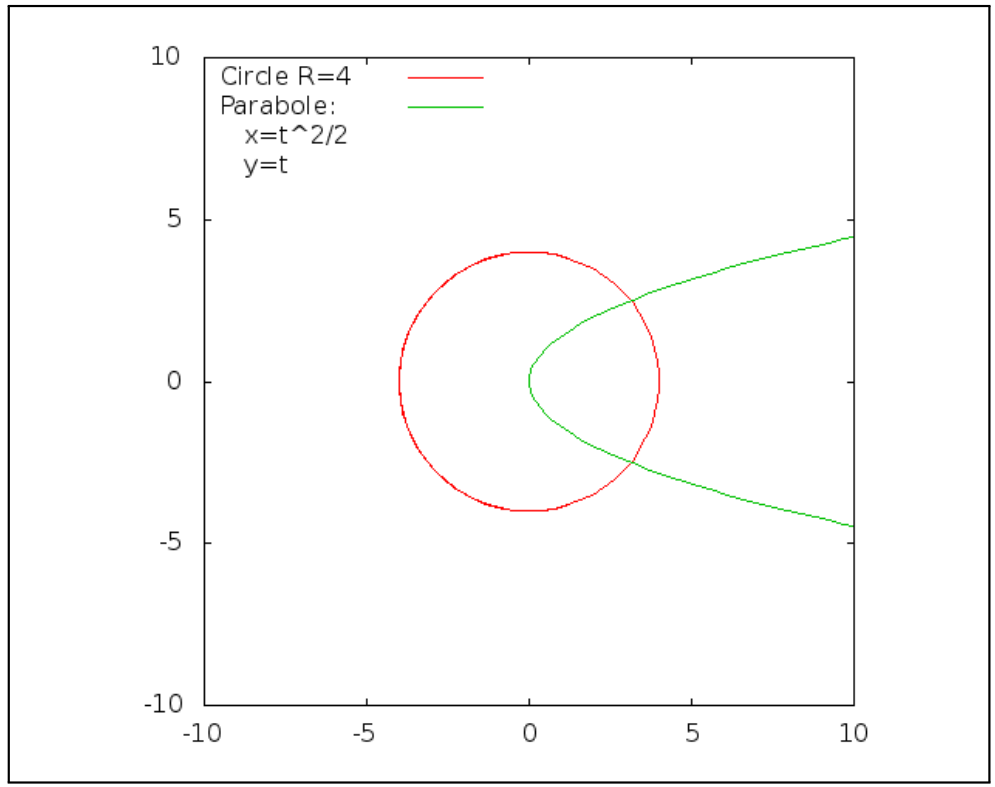
```
1  set boxwidth 0.9 absolute
2  set style fill solid 1.00 border -1
3  set style histogram clustered gap 1 title offset character 0, 0, 0
4  set datafile missing '-'
5  set style data histograms
6  set xtics border in scale 1,0.5 nomirror rotate by -45
   offset character 0, 0, 0
7  set xtics ("1891-1900" 0.00000, "1901-1910" 1.00000, "1911-1920"
   2.00000,"1921-1930" 3.00000, "1931-1940" 4.00000,
   "1941-1950" 5.00000, "1951-1960" 6.00000, "1961-1970" 7.00000)
8  set title "US immigration from Northern Europe\n
   Plot selected data columns as histogram of clustered boxes"
9  set yrange [ 0.00000 : 300000. ] noreverse nowriteback
10 plot 'immigration.dat' using 6:xtic(1) ti col,
   '' u 12 ti col, '' u 13 ti col, '' u 14 ti col
```

Pie plots

```
1   circ="circle at 0,0 size 1"
2   wedge(color, s, e, title)= sprintf("set obj %d @circ arc [%d:%d] fc rgb c%d;\
3   replot -2 linecolor rgb c%d title \"%s\"",color,s,e,color,color,title)
4   set style fill
    solid 1.0 border -1
5   set key outside
    horizontal center top
6   set size square
7   set notics
8   set xrange [-1.2:1.2]
9   set yrange [-1.2:1.2]
10  c1="red"
11  c2="orange"
12  c3="yellow"
13  c4="forest-green"
14  c5="dark-turquoise"
15  c6="dark-magenta"
16  set style function filledcurves
17  set macro
18  plot -2 notitle
19  eval wedge(1,0,40, "Carbon 11.1%")
20  eval wedge(2,40,70, "Gold 8.3%")
21  eval wedge(3,70,90, "Mercury 5.5%")
22  eval wedge(4,90,120, "Iron 8.3%")
23  eval wedge(5,120,220, "Silver 27.7%")
24  eval wedge(6,220,360, "Tungsten 38.8%")
```



Plotting parametric curves

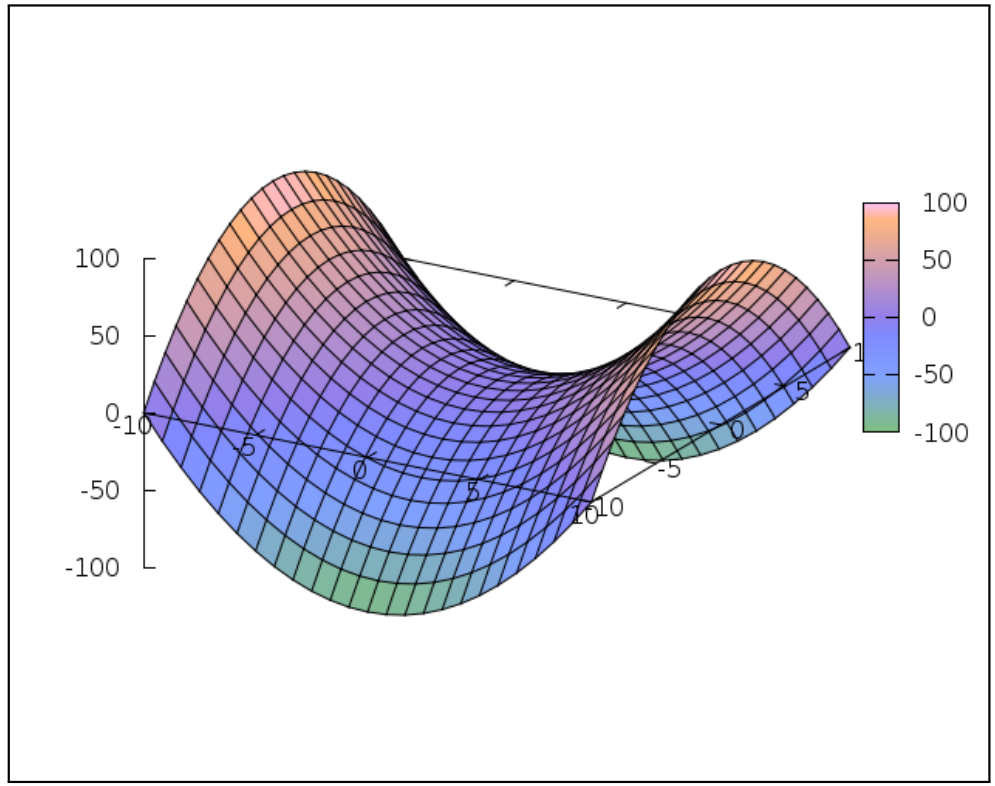



```
1  set xrange[-10:10]
2  set yrange[-10:10]
3  set size square
4  set parametric
5  set termoption enhanced
6  R=4
7  set key Left left
8  plot R*cos(t),R*sin(t) title "Circle R=4"
9  replot 0.5*t**2, t title "Parabole:\n
   x=t^2/2\n\r y=t"
```

Interactive elements

New terminal type based on `wx` library allows some user interaction like rotating, panning and zooming with mouse. It also provides the way to define keyboard short-cuts for user commands.

Visualisation of function of two variables




```
1   f(x,y) = x*x-y*y
2   set isosamples 25, 25
3   unset key
4   set style fill solid
    0.50 border
5   set hidden3d front offset 1
    trianglepattern 3
    undefined 1
    altdiagonal bentover
6   set xyplane at 0
7   set palette rgbformulae 31, -11, 32
8   set colorbox vertical origin screen 0.9, 0.2, 0
    size screen 0.05, 0.6, 0 front bdefault
9   splot f(x,y) with pm3d
10  replot f(x,y) with lines lt 1 lc rgb "#000000"
```

Command line history

- history show the complete history
- history 5 show last 5 entries in the history
- history quiet 5 show last 5 entries without entry numbers
- history "hist.gp" write the complete history to file hist.gp
- history "hist.gp" append append the complete history to file hist.gp
- history 10 "—head -5 >>diary.gp" write 5 history commands using pipe
- history 10 "hist.gp" write last 10 commands to file hist.gp
- history ?load show all history entries starting with "load"
- history ?"set c" like above, several words enclosed in quotes
- hi !reread execute last entry starting with "reread"
- hist !"set xr" like above, several words enclosed in quotes
- hi !hi guess yourself :-))

Gnuplot scripts

Gnuplot scripts can be run

- in terminal
- during interactive gnuplot session using `load` command.

Other gnuplot features

More gnuplot features can be seen at gnuplot demo pages:

`http://gnuplot.sourceforge.net/demo/`

Part II

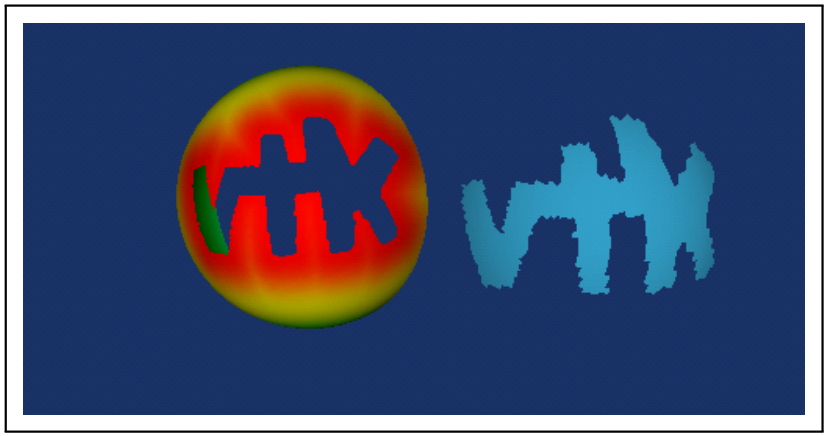
VTK Overview

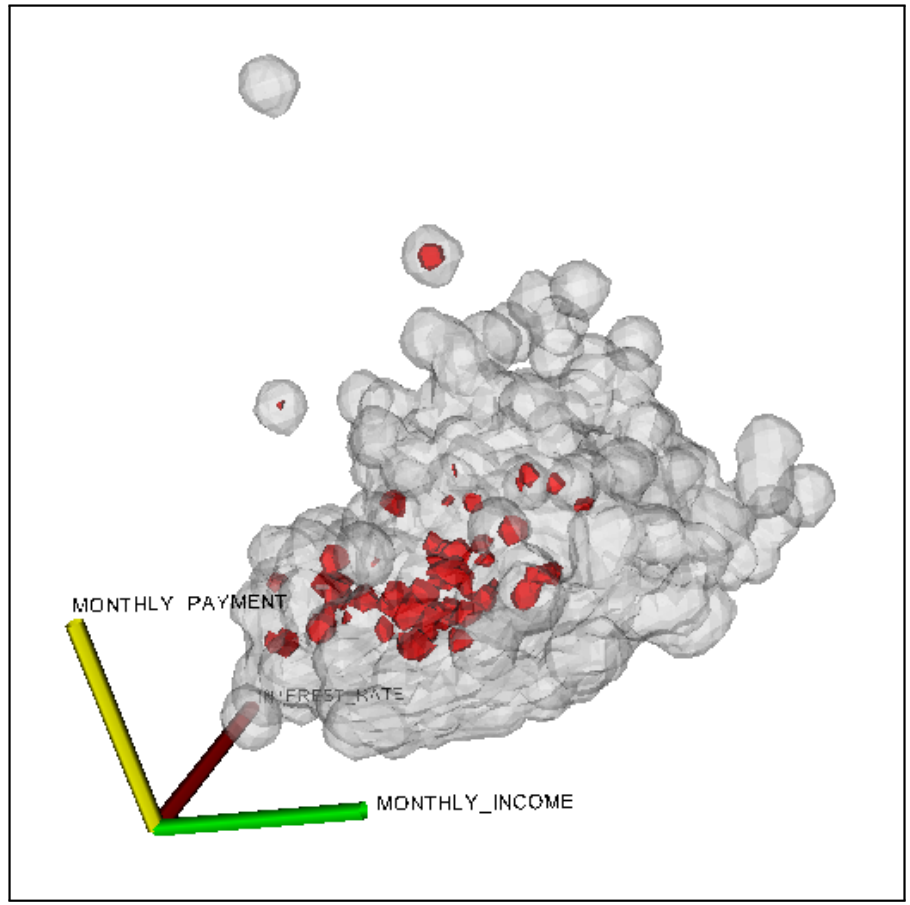
VTK – Visualization Toolkit

The Visualization ToolKit (VTK) is an open source, freely available software system for 3D computer graphics, image processing, and visualization.

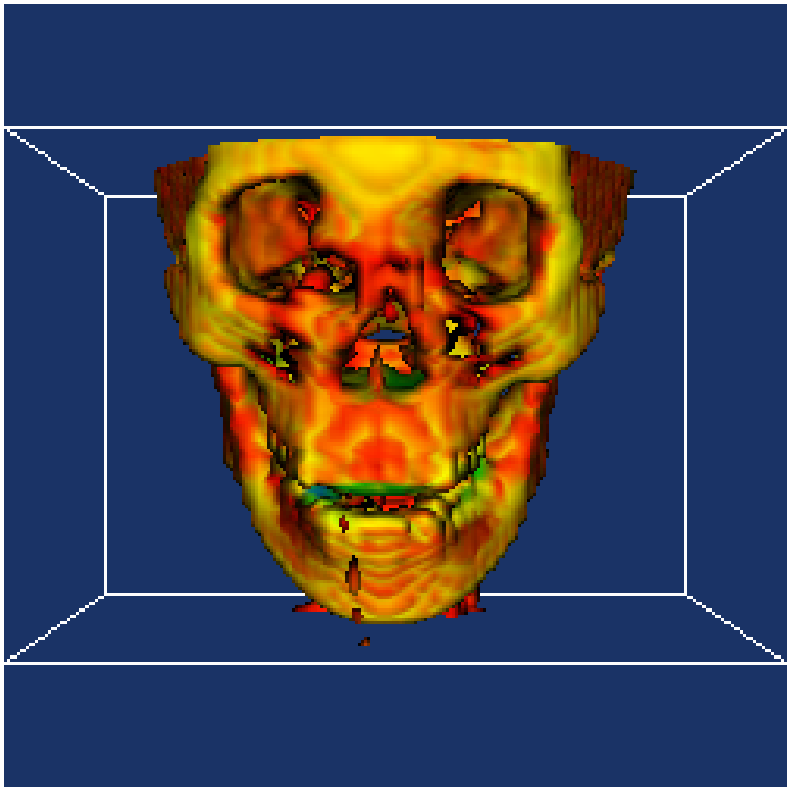
- Book: "The Visualization Toolkit, An Object-Oriented to 3D Graphics, 2nd edition", Prentice-Hall
- Software: `http://www.kitware.com/vtk.html`

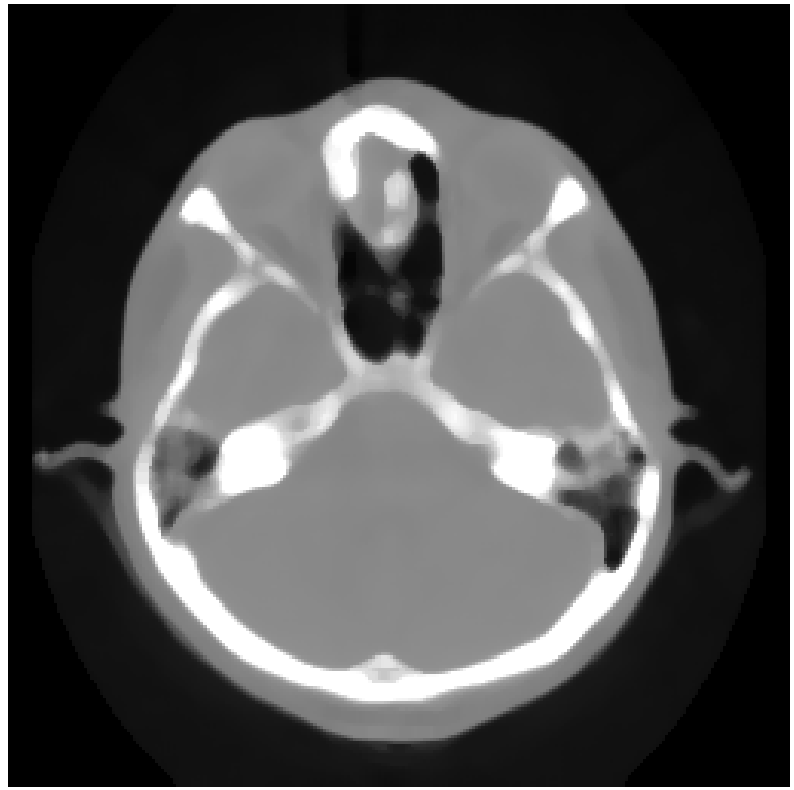
VTK Applications



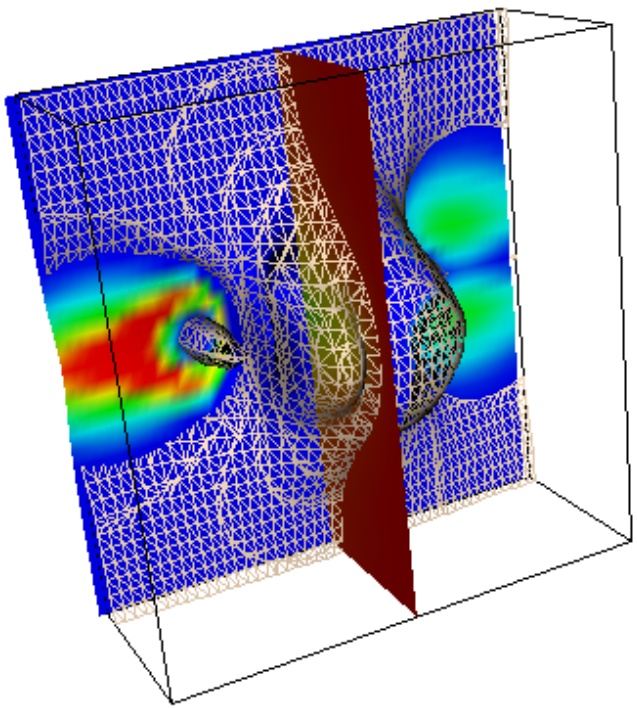


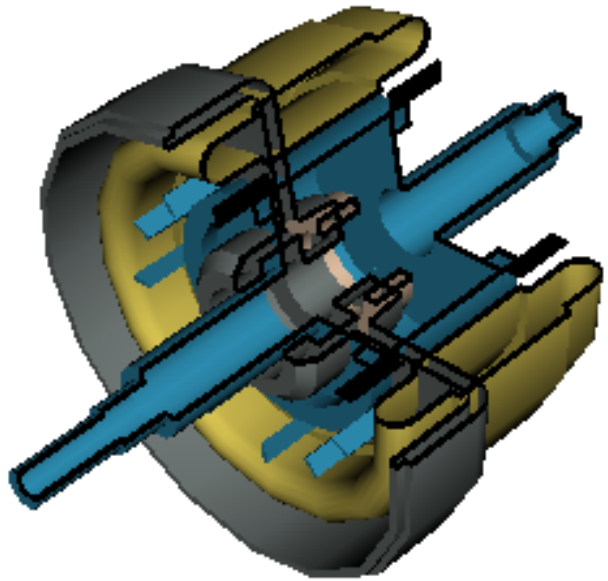
VTK Applications – cont.



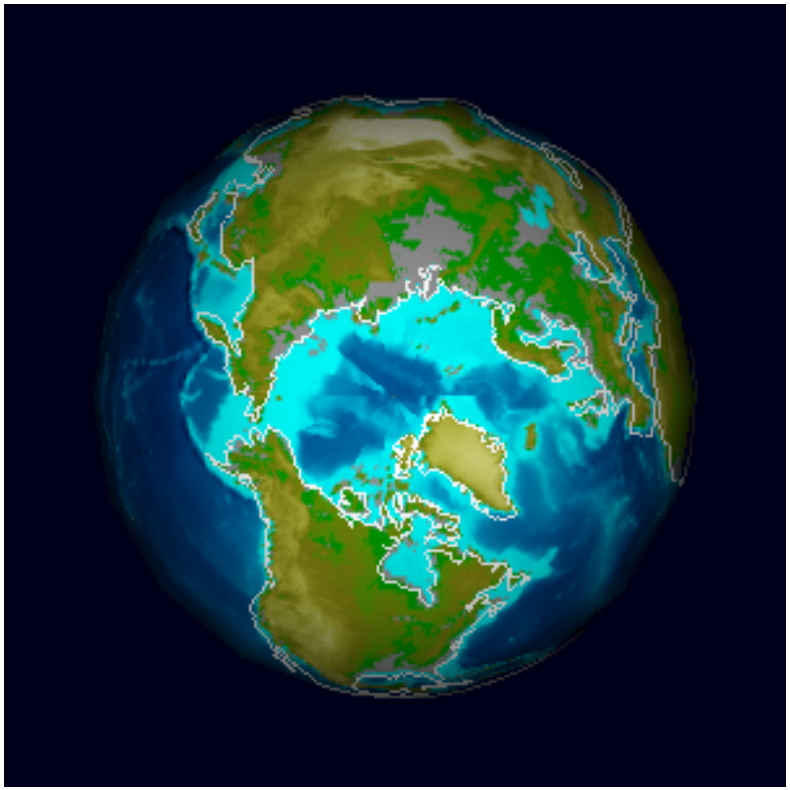


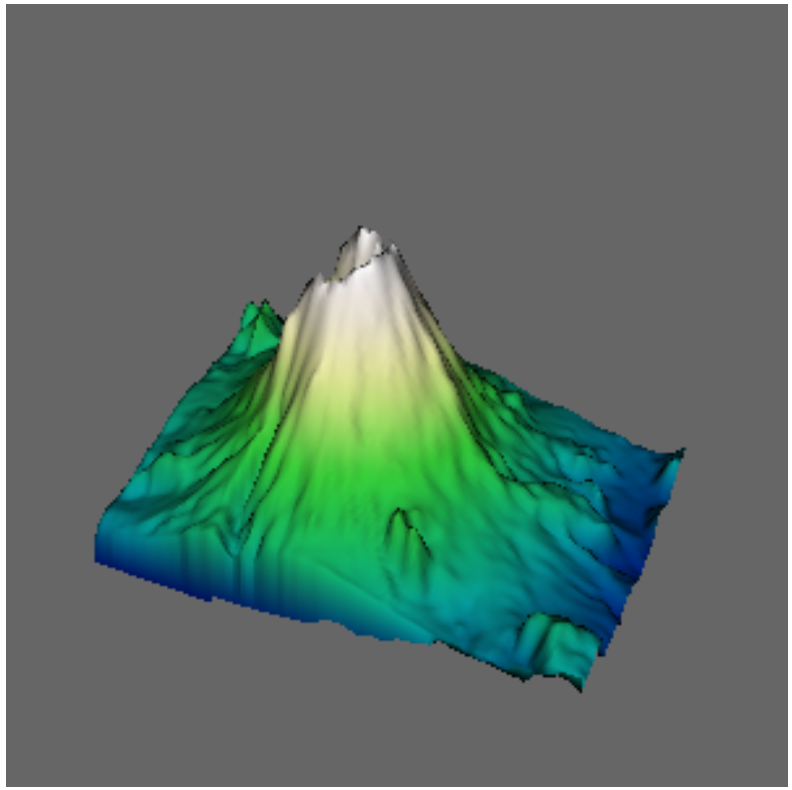
VTK Applications – cont.





VTK Applications – cont.





VTK on `jinx`

Which packages: `dpkg -l | grep -i vtk`

- Visualization Toolkit - A high level 3D visualization library
- VTK header files for building C++ code
- Python bindings for VTK
- Tcl bindings for VTK
- VTK class reference documentation
- C++, Tcl and Python example programs/scripts

Technical Overview

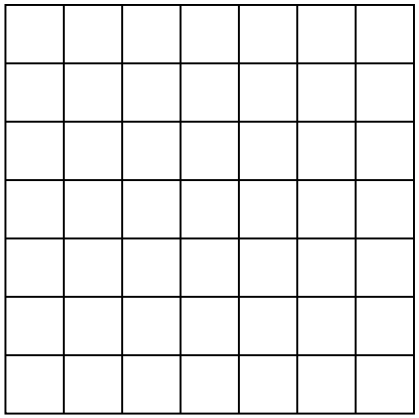
- C++ implementation
- Open source
- Scripting interface: Tcl/Tk, Python, Java
- Portable
- Supports parallelization
- Commercial support

- Not super-fast graphics engine
- Very large – not a toy
- Requires decent system to use it effectively

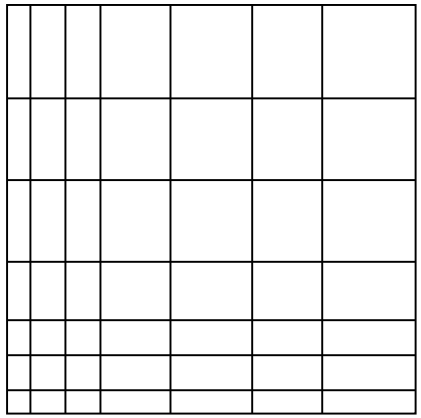
VTK: 3D Graphics

- Surface Rendering:
  OpenGL, gl, starbase, xgl
- Volume Rendering
  - Flexible ray casting implementation
  - Support for VoumePRO volume rendering hardware
  - Supports mixing opaque surface geometry and volume rendering
- Rendering primitives:
  points, lines, polygons, tringle strips, volumes

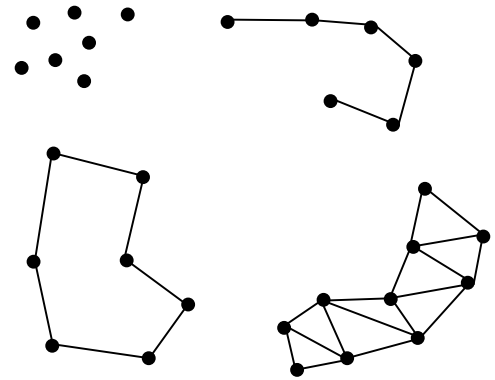
VTK: Datasets Types



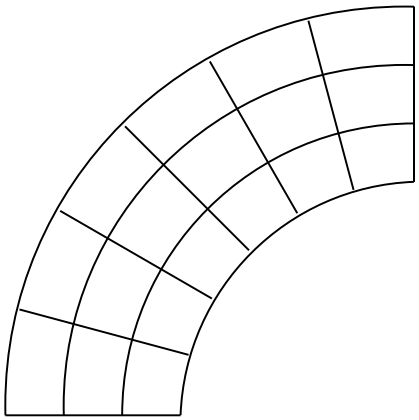
Structured Points



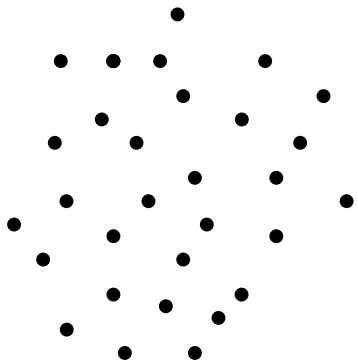
Rectlinear Grid



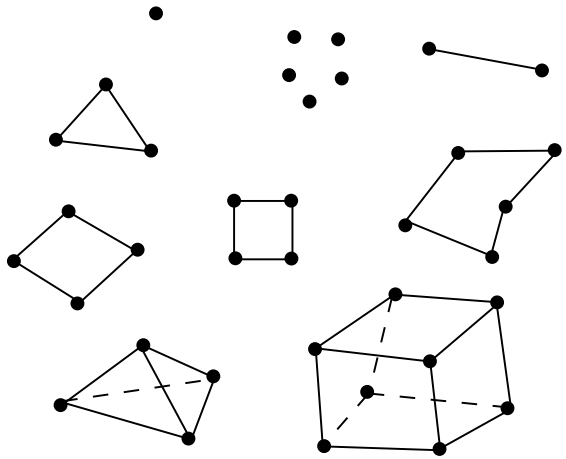
Polygonal Data



Structured Grid



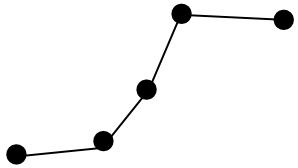
Unstructured Points



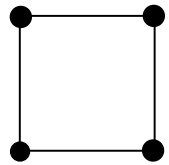
Unstructured Grid

VTK: Cell Types

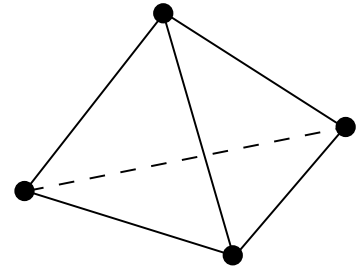
Vertex



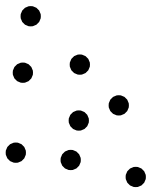
Polyline



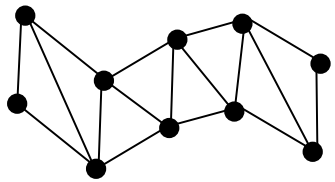
Pixel



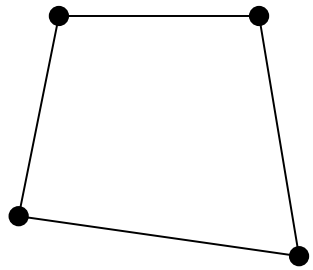
Tetrahedron



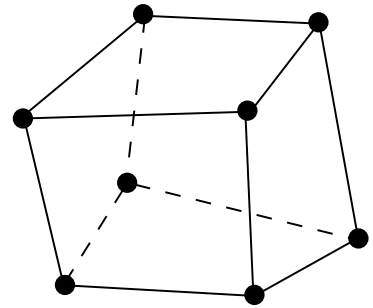
Polyvertex



Triangle strip

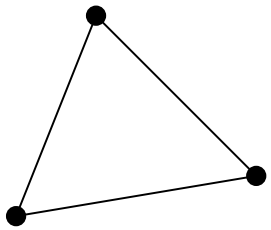


Quad

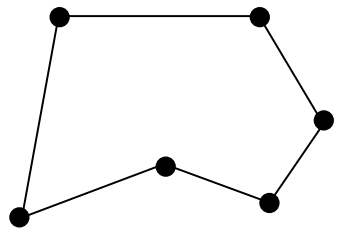


Hexahedron

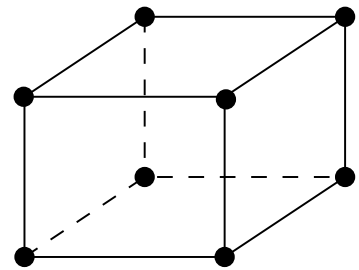
Line



Triangle



Polygon

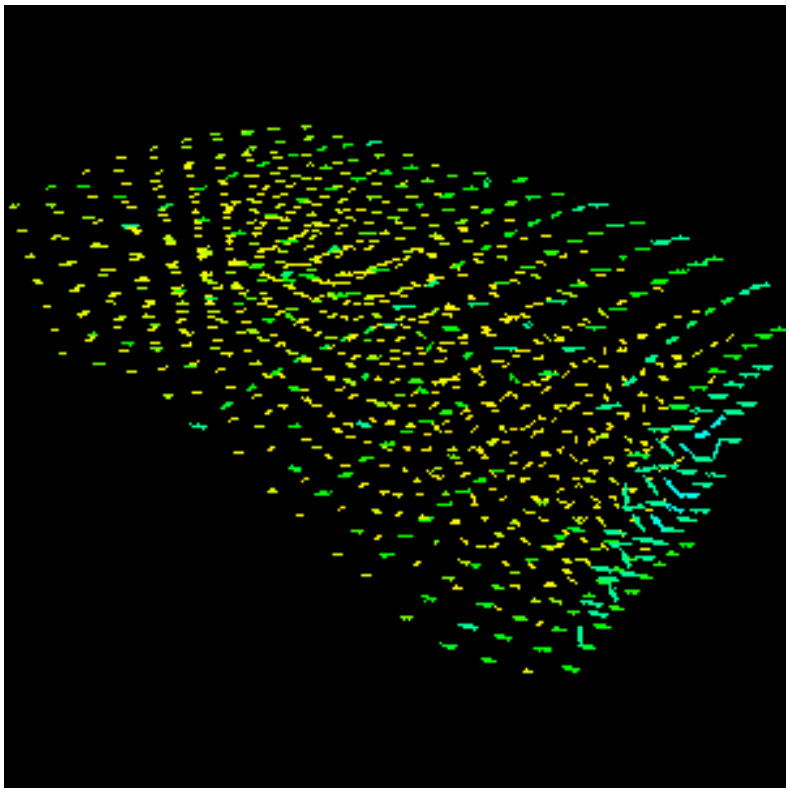


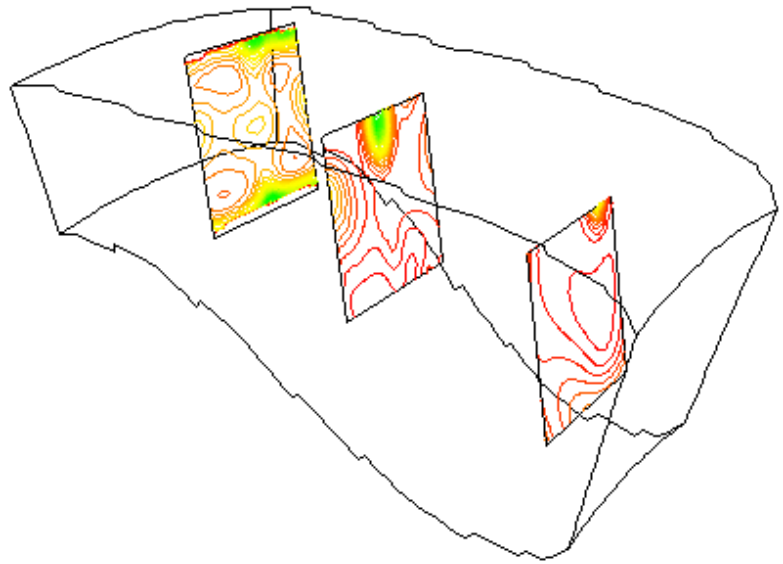
Voxel

VTK: Atribute Types

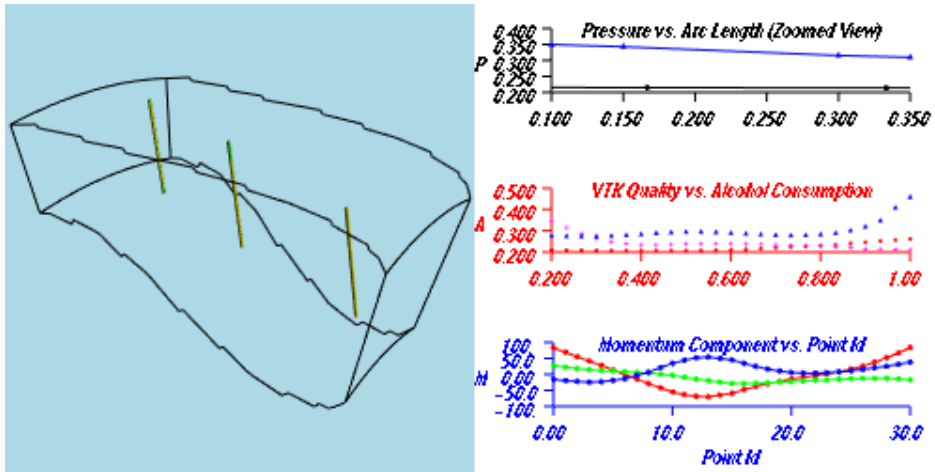
- Scalars (single valued + grayscale, grayscale-alpha, rgb, rgb-alpha)
- Vectors
- 3x3 Tensors
- Texture Coordinates (1-3D)
- Field Data

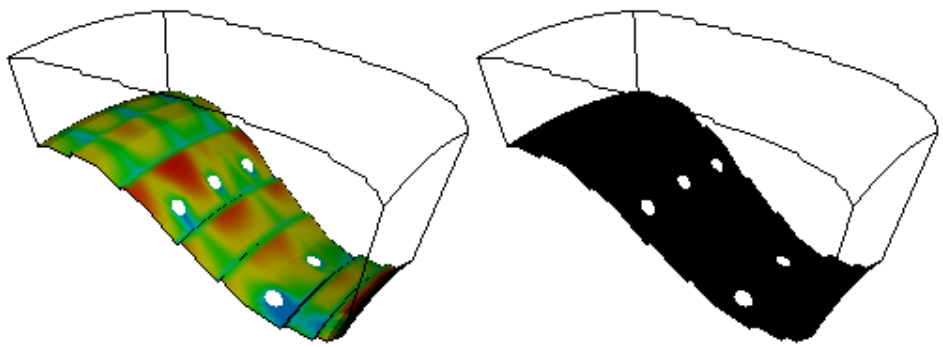
VTK: Scalar Algorithms



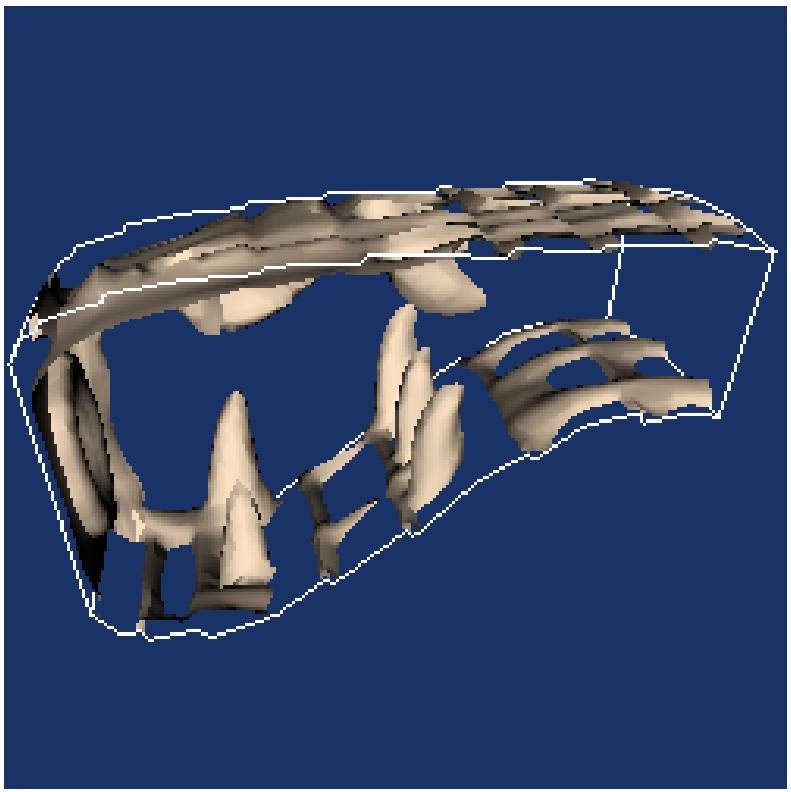


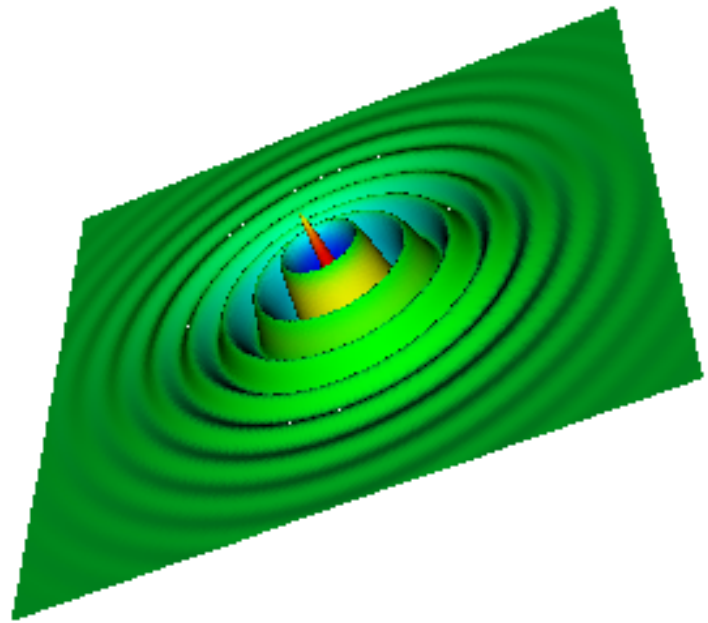
VTK: Scalar Algorithms – cont.



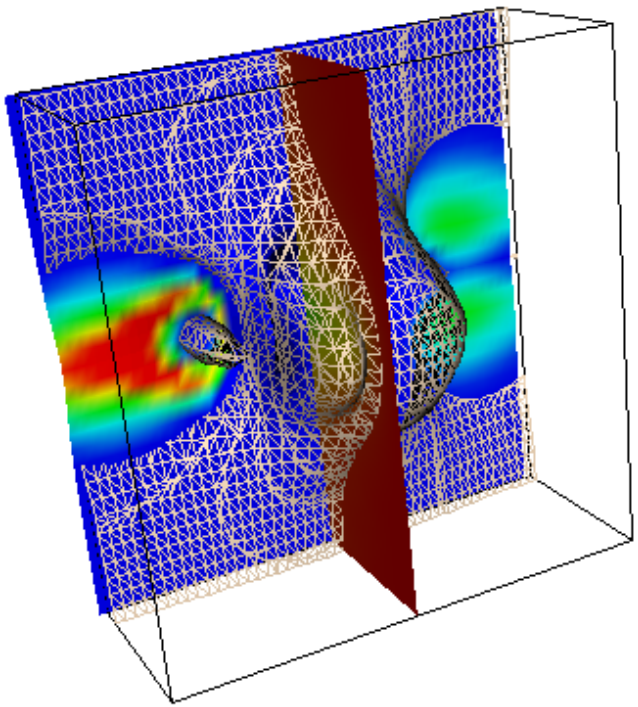


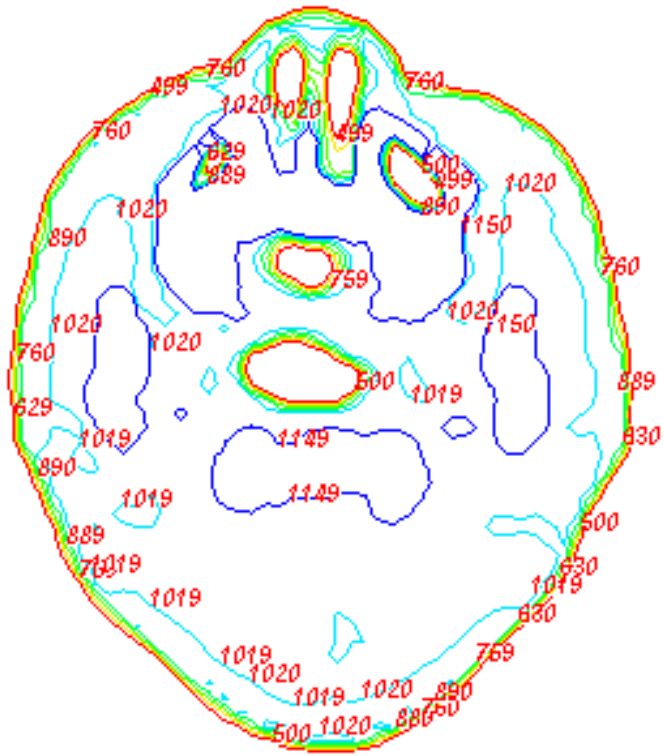
VTK: Scalar Algorithms – cont.



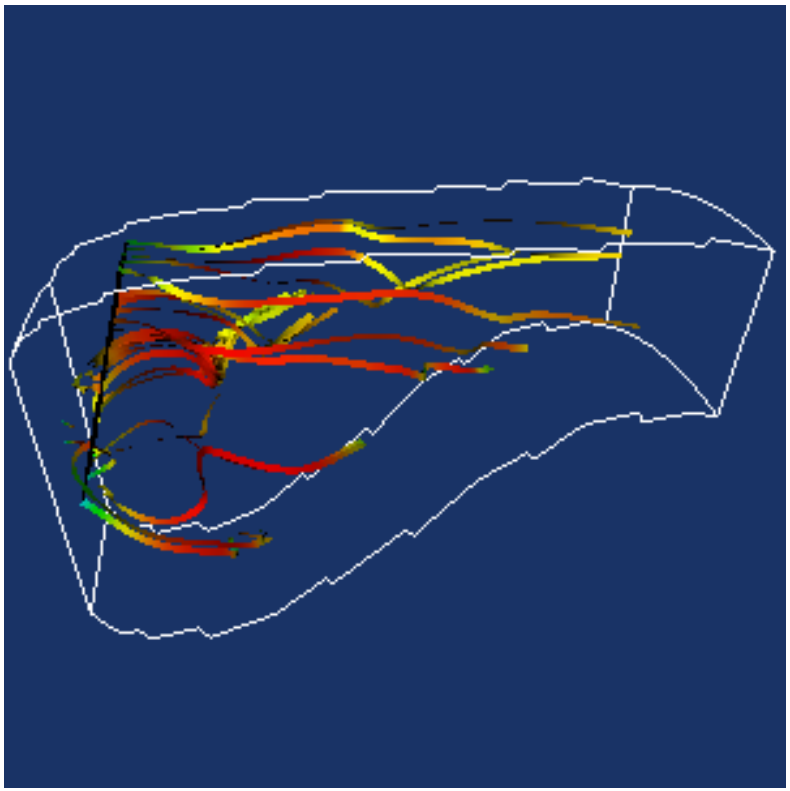


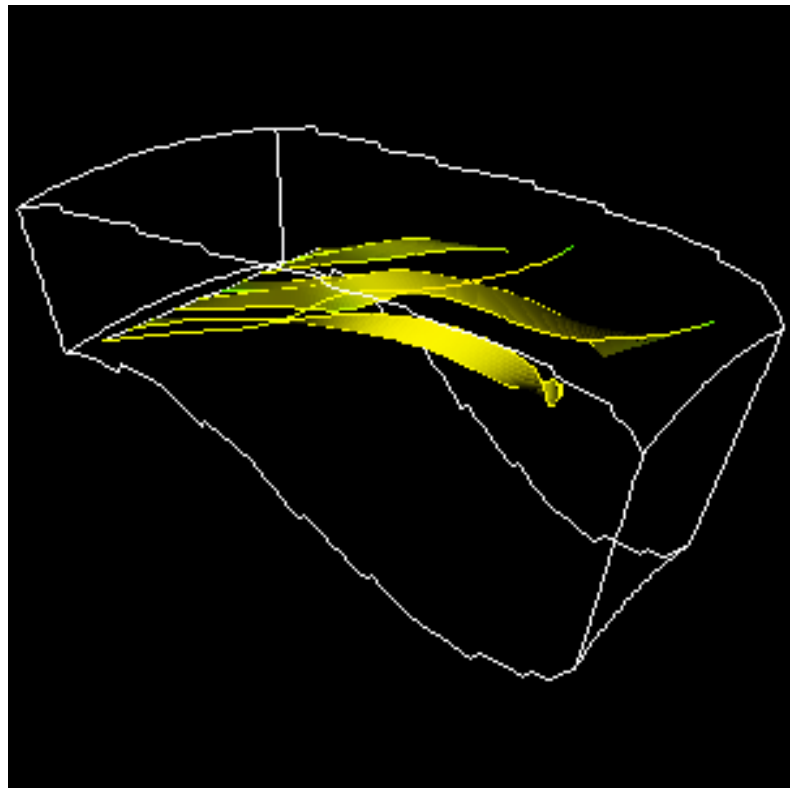
VTK: Scalar Algorithms – cont.



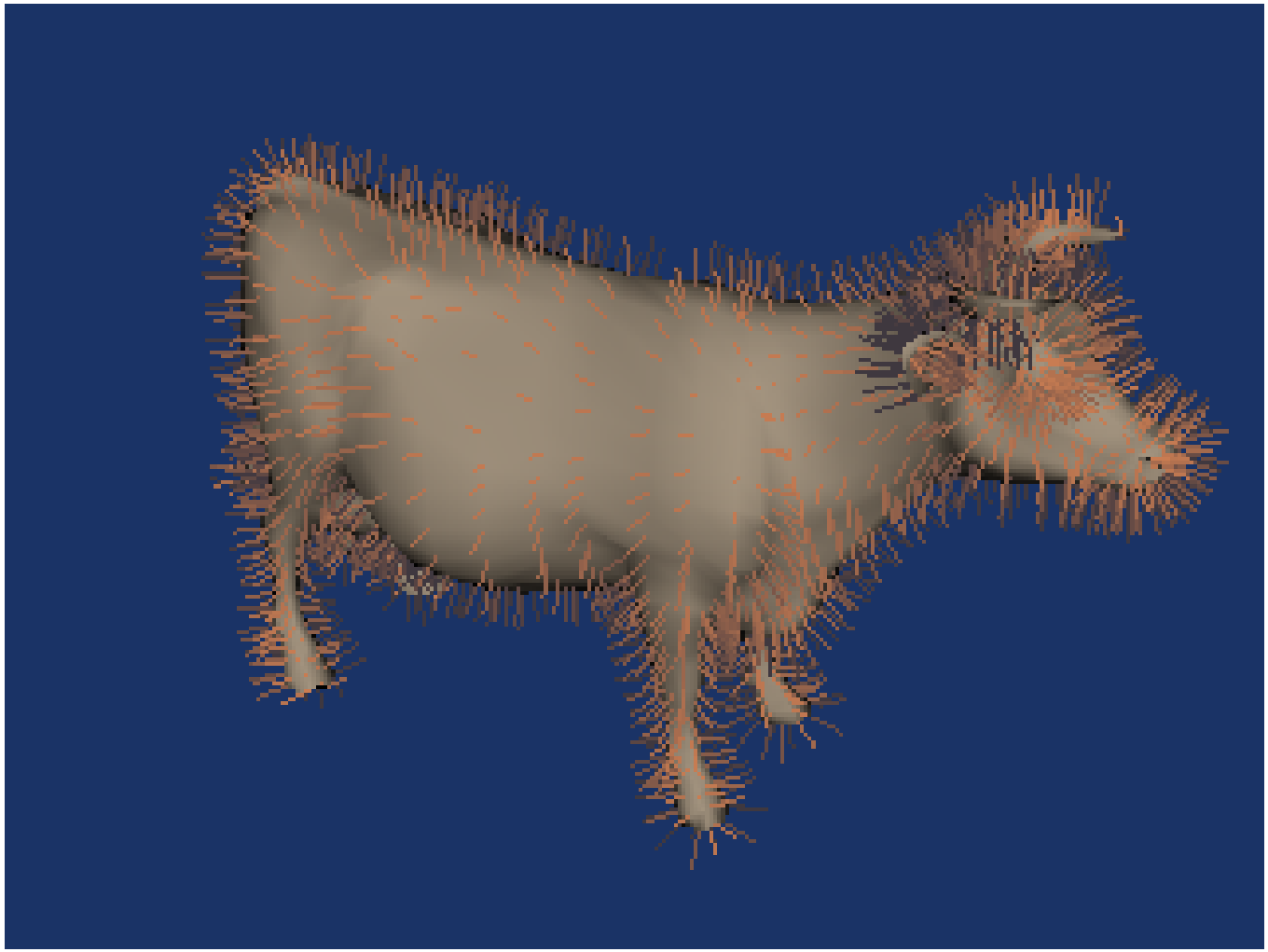


VTK: Vector Algorithms

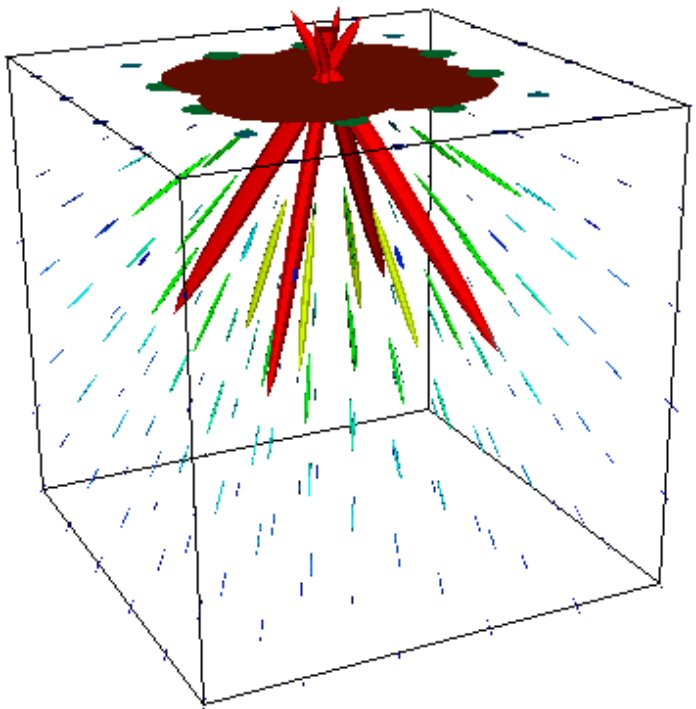


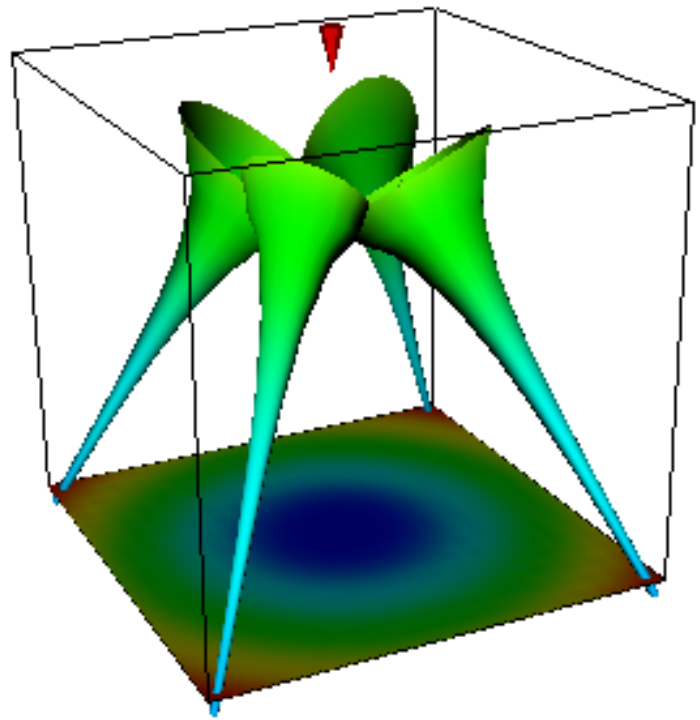


VTK: Vector Algorithms – cont.

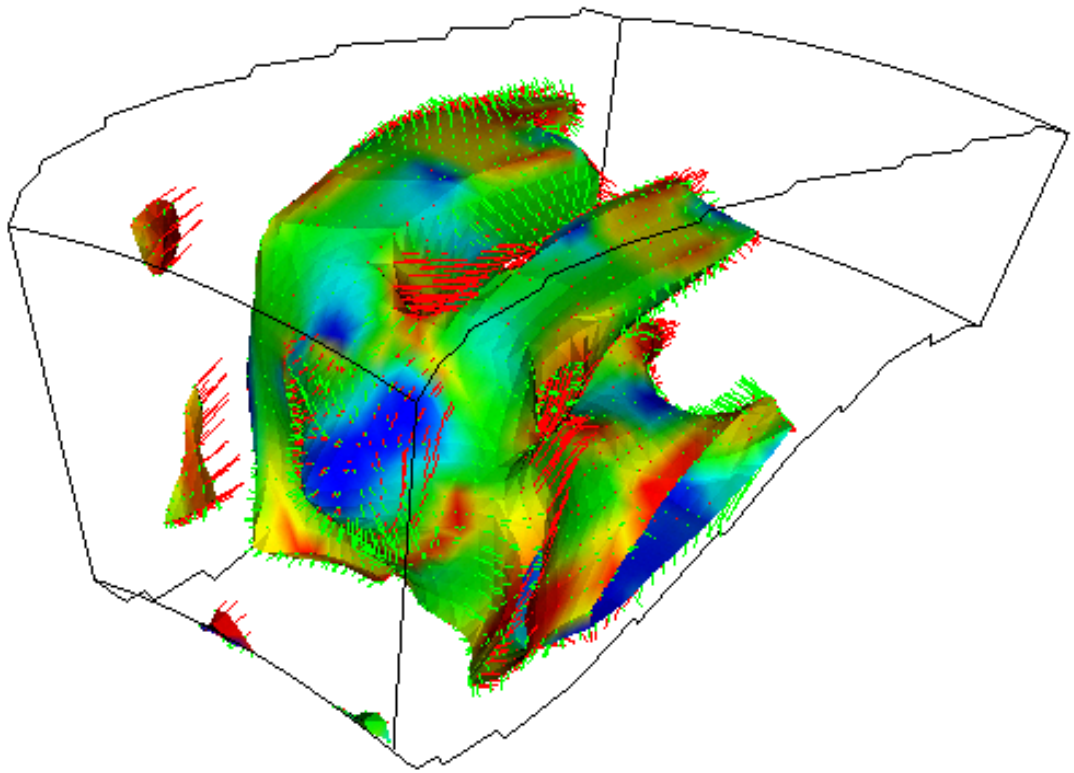


VTK: Tensor Algorithms

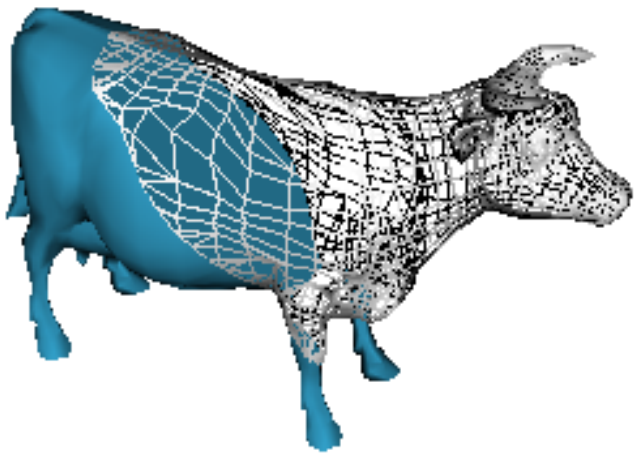


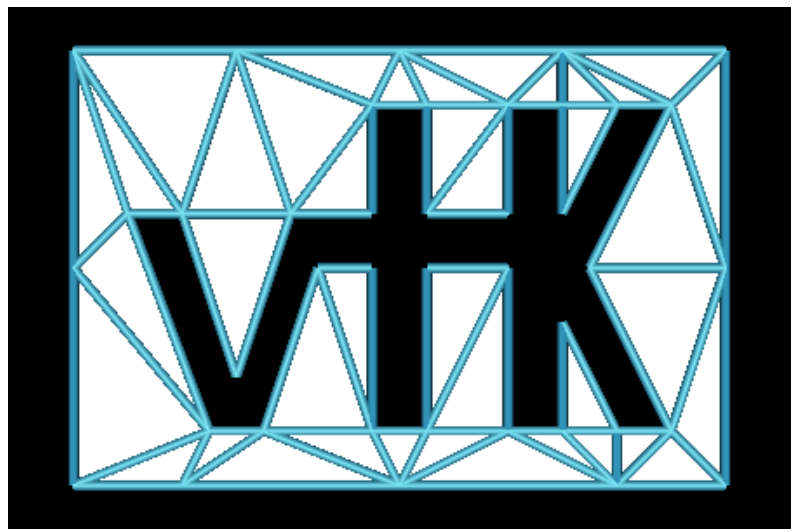


VTK: Multidimensional solution

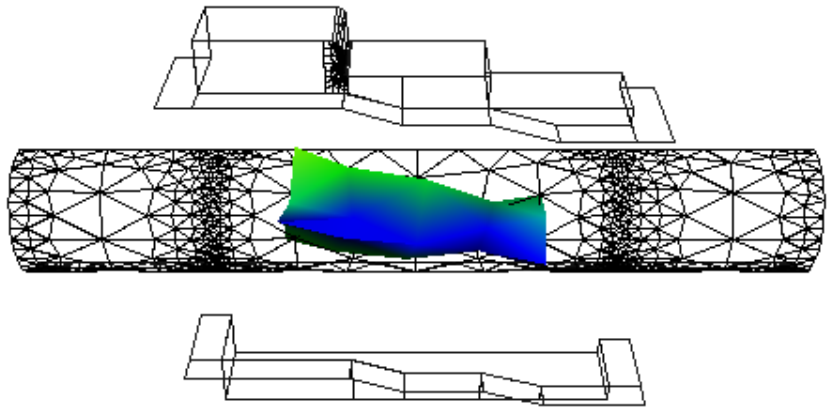


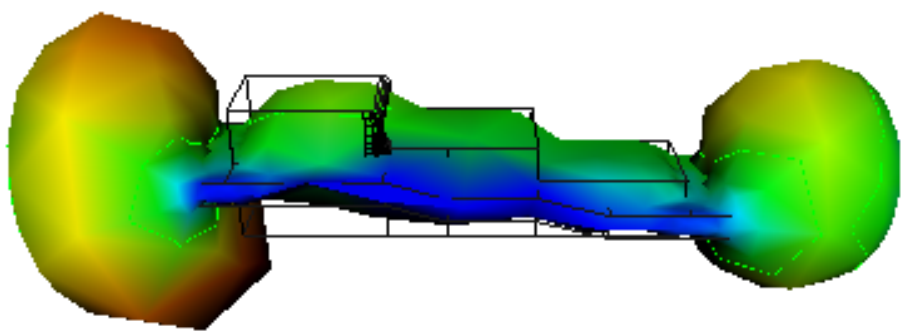
VTK: Modeling Algorithms



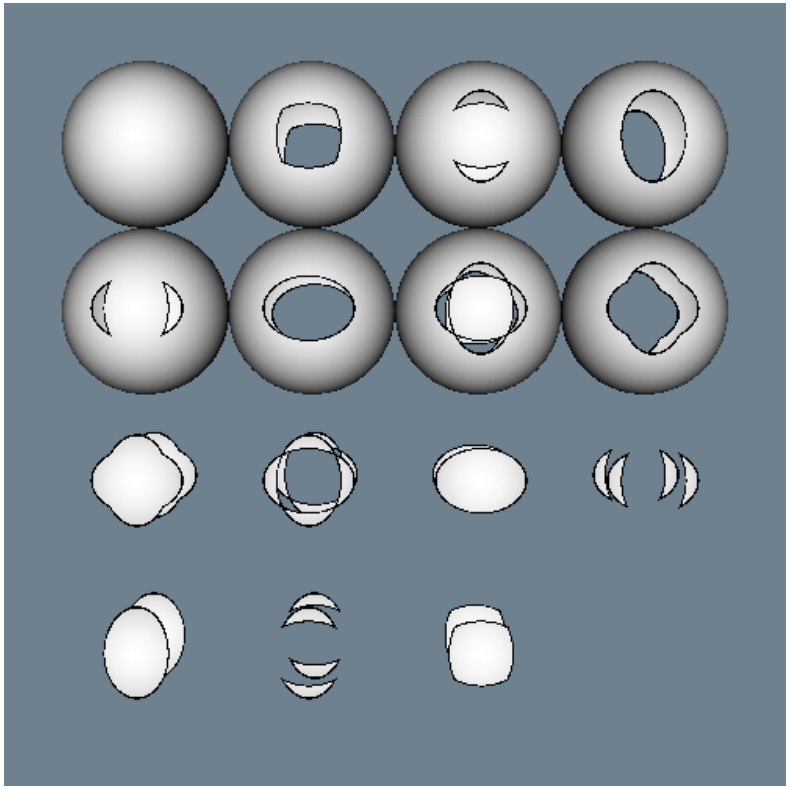


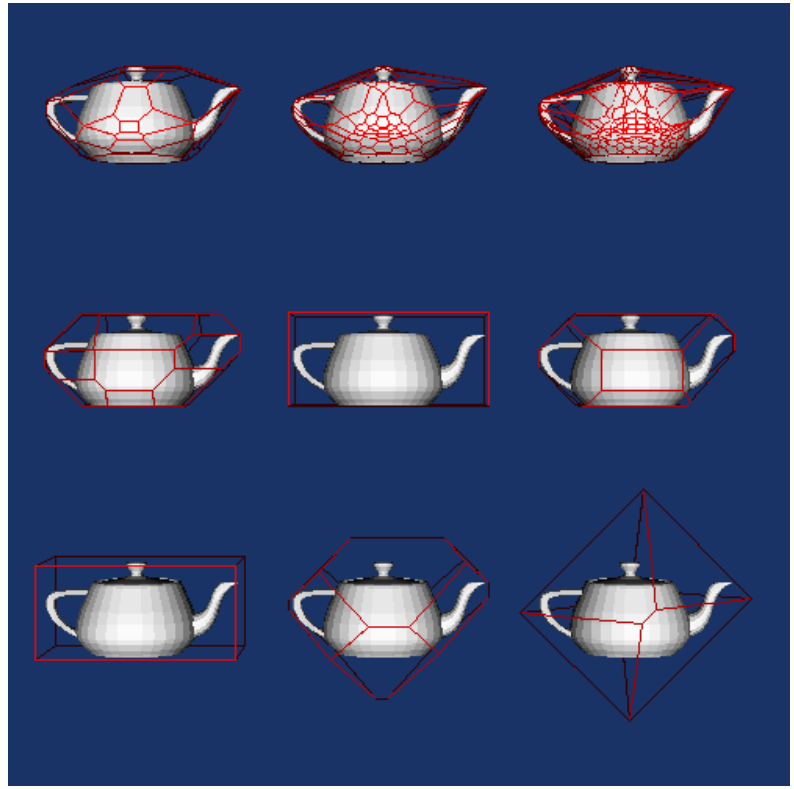
VTK: Modeling Algorithms – cont.



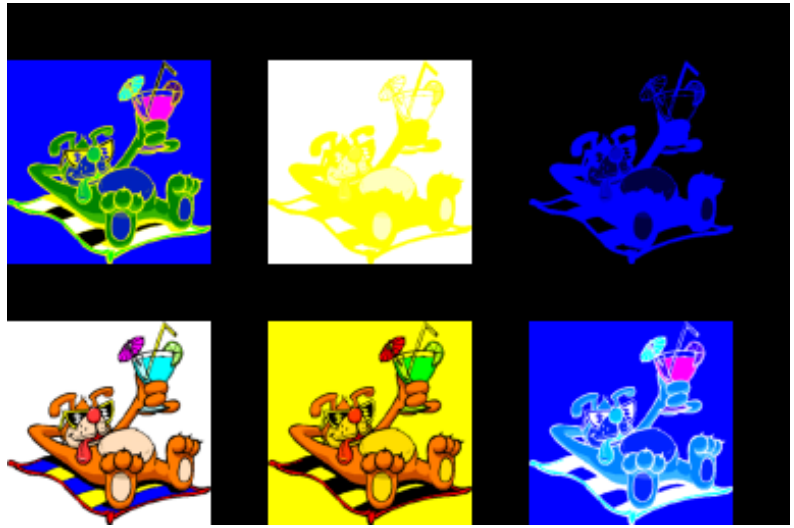


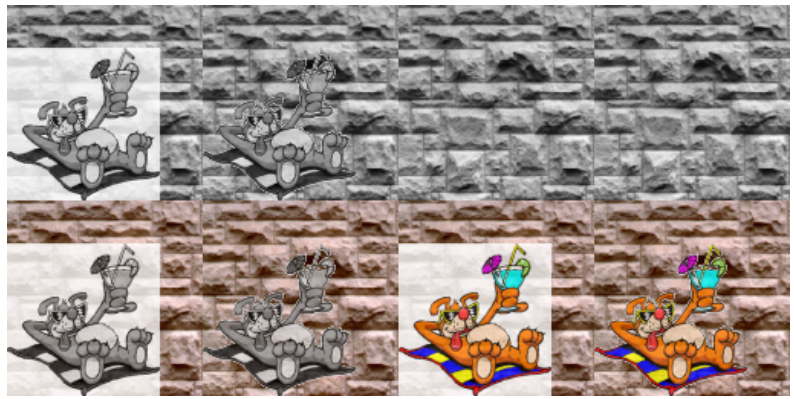
VTK: Modeling Algorithms – cont.





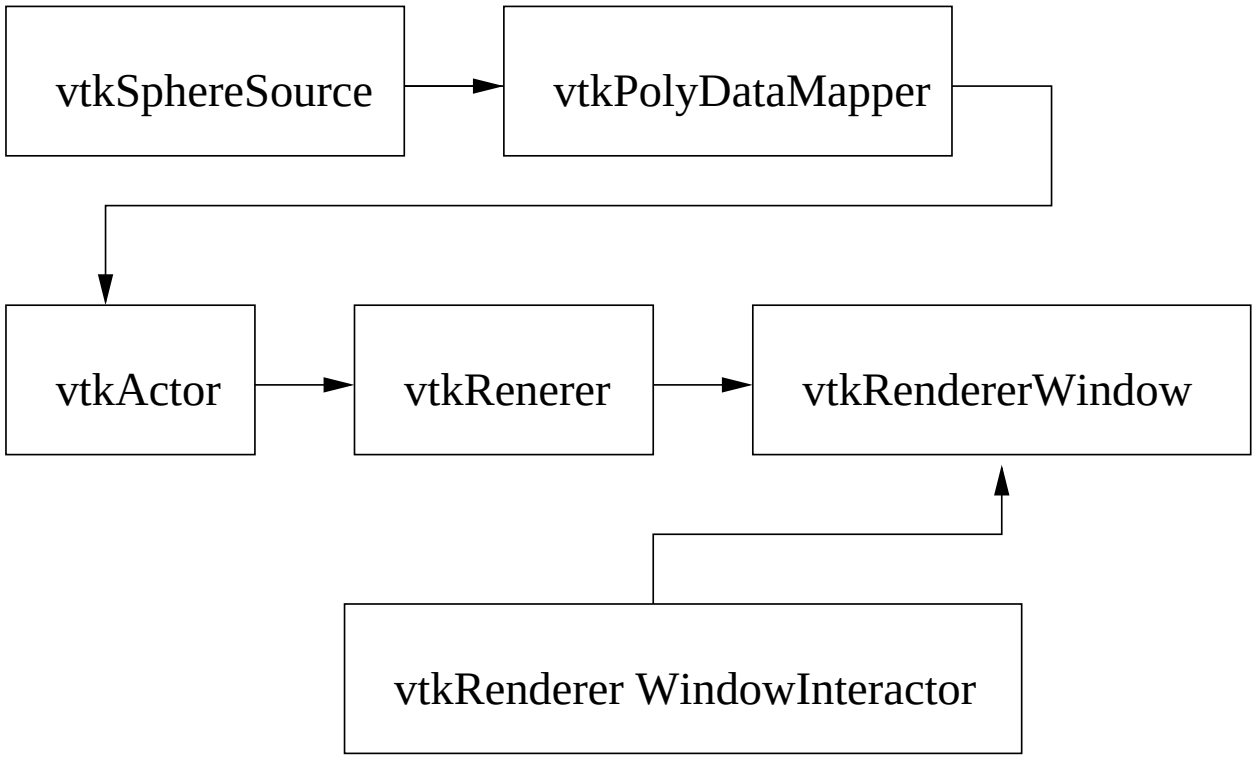
VTK:Imaging





VTK: Visualization Pipeline

- Demand-driven data-flow with automatic network updates
- Network looping and feedback supported
- Supports multiple input/multiple output filters



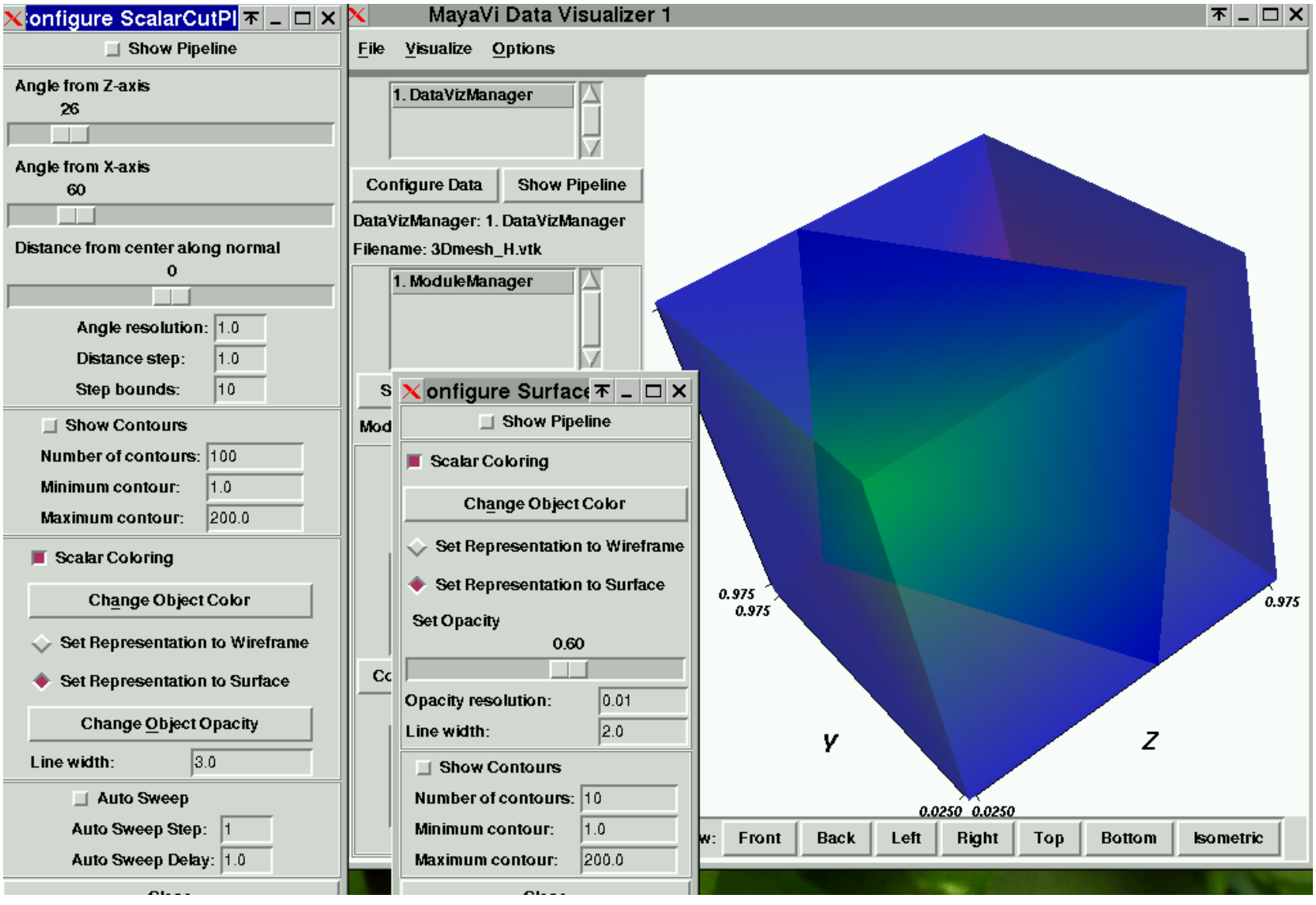
VTK: Programming - Tcl/Tk

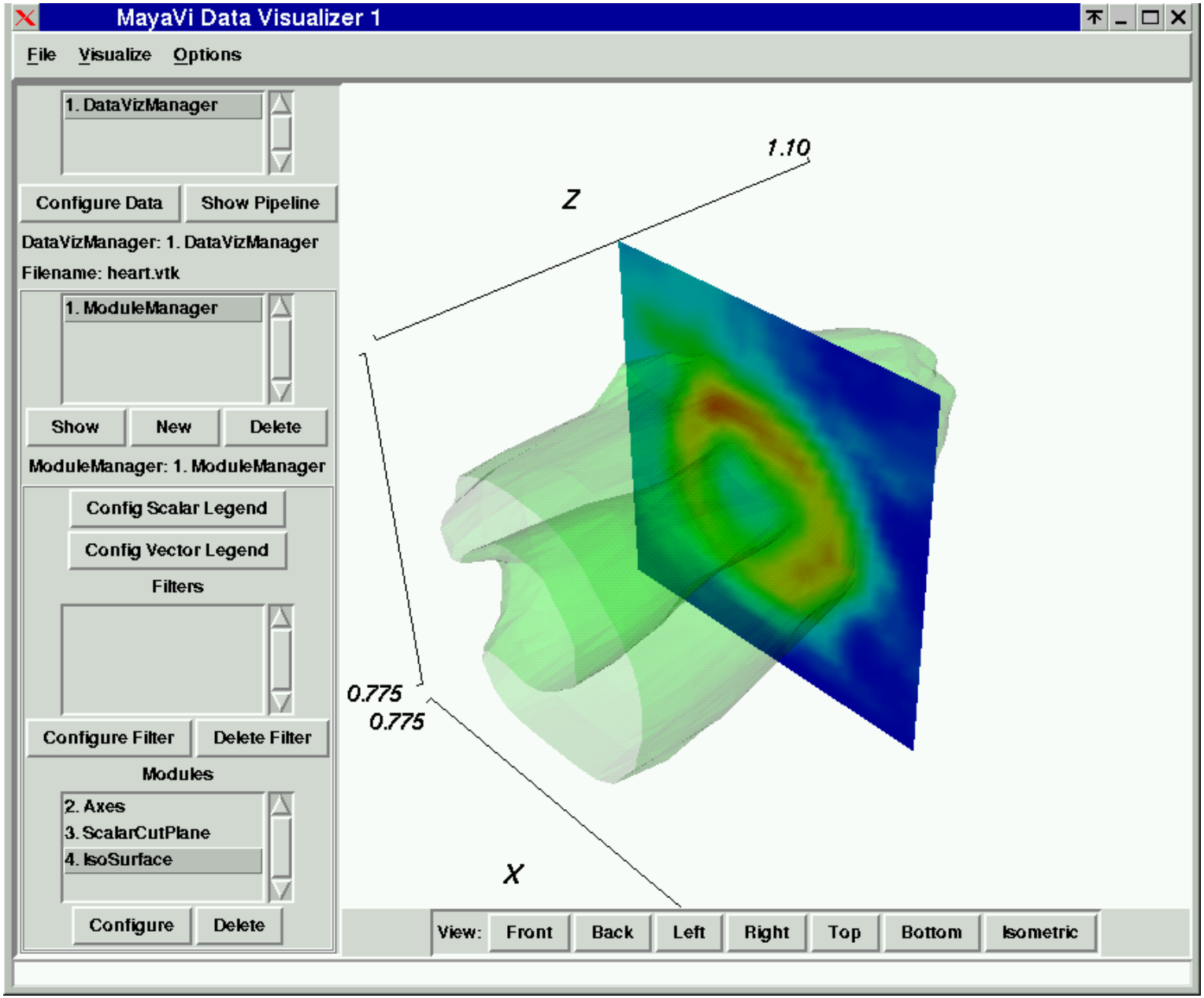
```
1  vtkSphereSource sphere
2  sphere SetRadius 1.0
3  sphere SetThetaResolution 18
4  sphere SetPhiResolution 18
5  vtkPolyDataMapper map
6  map SetInput [sphere GetOutput]
7  vtkActor aSphere
8  aSphere SetMapper map
9  [aSphere GetProperty] SetColor 0 0 1
10 vtkRenderWindow renWin
11 vtkRenderer ren1
12 renWin AddRenderer ren1
13 vtkRenderWindowInteractor iren
14 iren SetRenderWindow renWin
15 ren1 AddActor aSphere
16 ren1 SetBackground 1 1 1
17 renWin Render
18 wm withdraw .
```

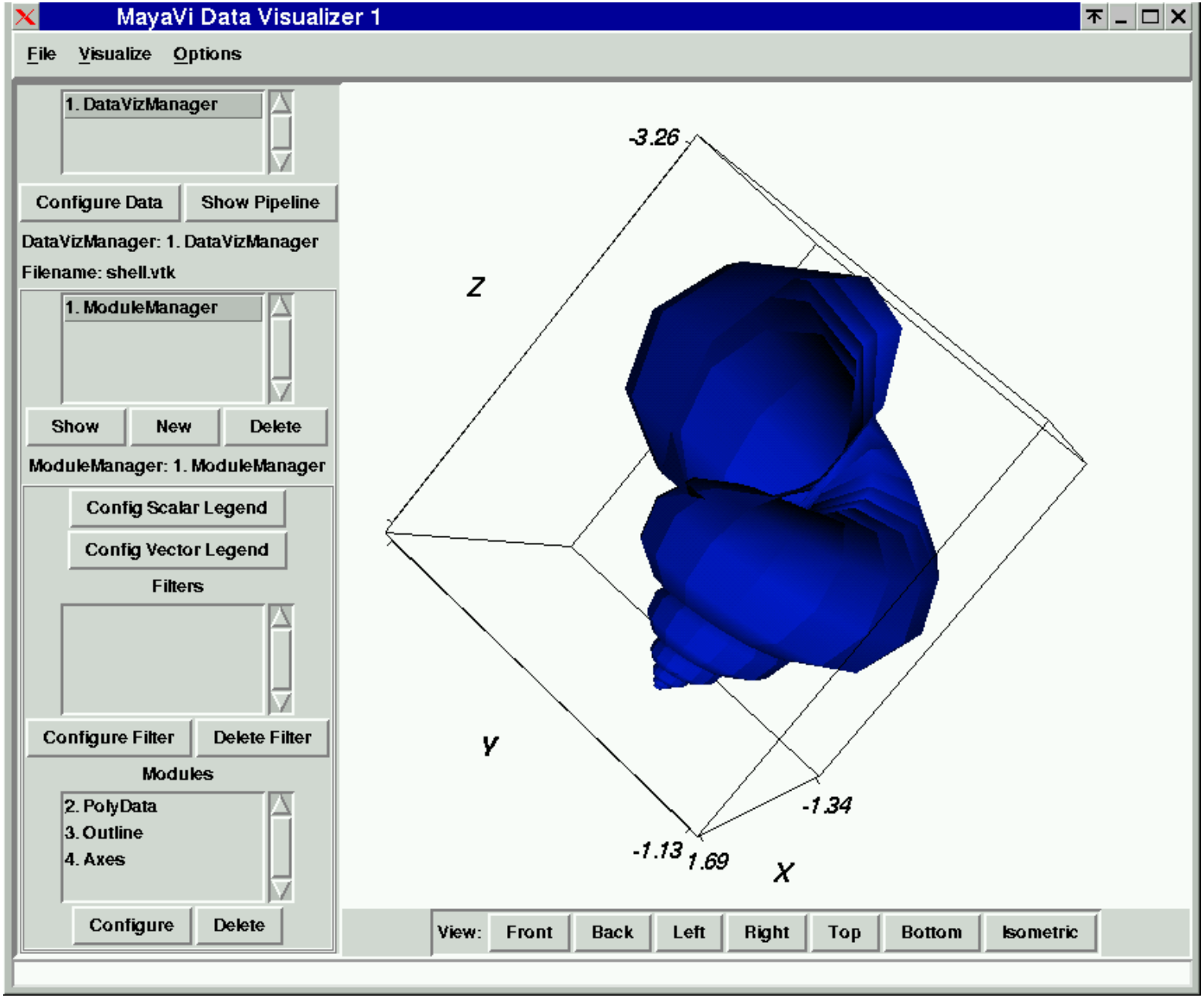
Mayavi

- free scientific data visualizer
- implemented in Python + Tkinter + VTK
- A pipeline browser can browse and edit objects in the VTK pipeline.
- Visualize computational grids.
- Visualize scalar, vector and tensor data.
- A modular design so you can add your own modules and filters.
- Output to: PostScript, PPM/BMP/TIFF/JPEG/PNG, Open Inventor, VRML or RenderMan.

Mayavi - examples




MayaVi Data Visualizer 1
File
Visualize
Options
1. DataVizManager
Configure Data
Show Pipeline
DataVizManager: 1. DataVizManager
Filename: heart.vtk
1. ModuleManager
Show
New
Delete
ModuleManager: 1. ModuleManager
Config Scalar Legend
Config Vector Legend
Filters
Configure Filter
Delete Filter
Modules
2. Axes
3. ScalarCutPlane
4. IsoSurface
Configure
Delete
1.10
Z
0.775
0.775
X
View:
Front
Back
Left
Right
Top
Bottom
Isometric


MayaVi Data Visualizer 1
File
Visualize
Options
1. DataVizManager
Configure Data
Show Pipeline
DataVizManager: 1. DataVizManager
Filename: shell.vtk
1. ModuleManager
Show
New
Delete
ModuleManager: 1. ModuleManager
Config Scalar Legend
Config Vector Legend
Filters
Configure Filter
Delete Filter
Modules
2. PolyData
3. Outline
4. Axes
Configure
Delete
-3.26
Z
Y
-1.34
-1.13
1.69
X
View:
Front
Back
Left
Right
Top
Bottom
Isometric

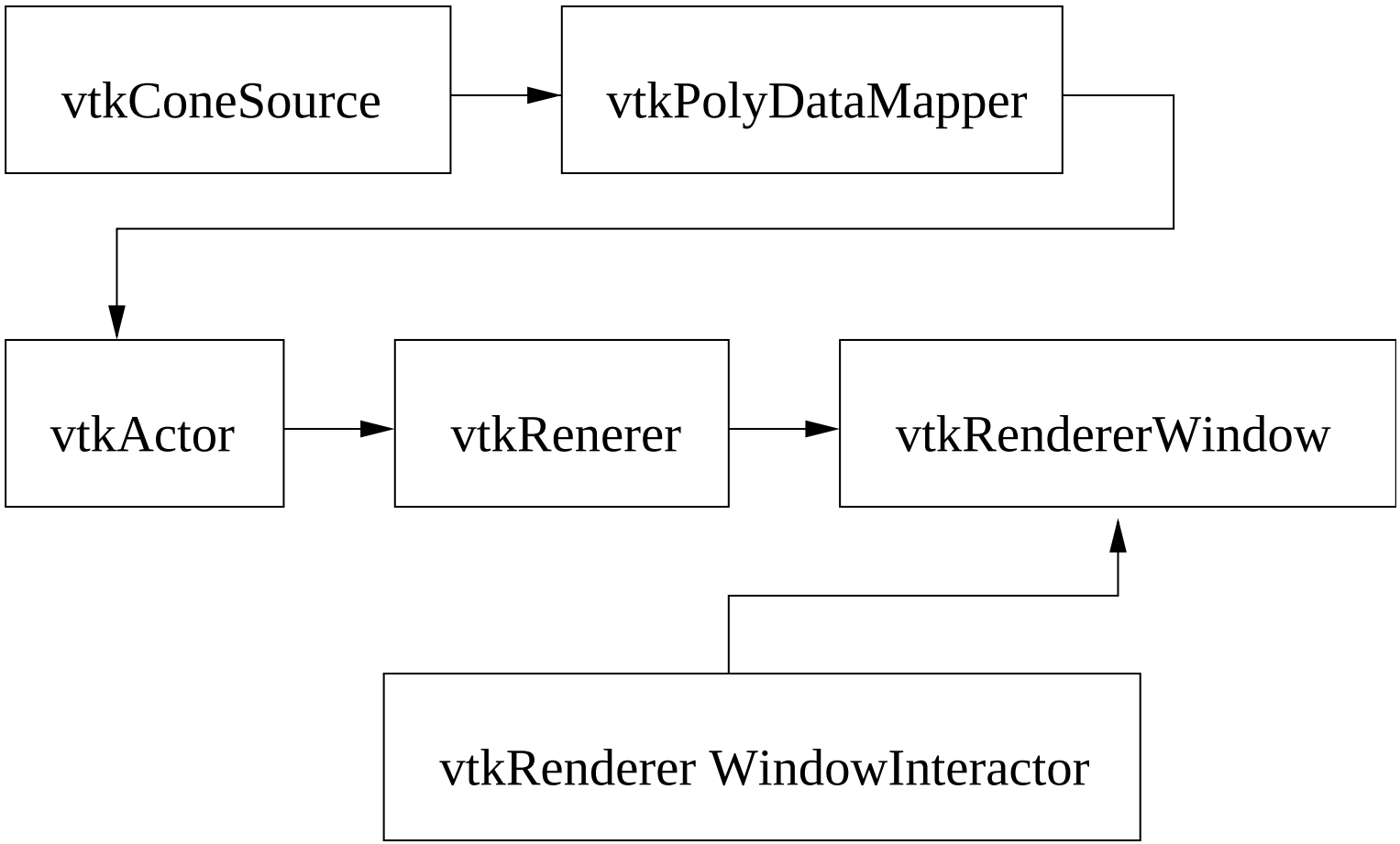
Credits

Most of the figures in this part and Tcl/Tk example come from VTK distribution.

Part III

VTK Programming

VTK: Visualization Pipe



VTK: Programming - C++

```
1   #include "vtkConeSource.h"
2   #include "vtkPolyDataMapper.h"
3   #include "vtkRenderWindow.h"
4   #include "vtkRenderWindowInteractor.h"
5   #include "vtkActor.h"
6   #include "vtkRenderer.h"
7
8   int main( int argc, char *argv[] ) {
9
10    vtkConeSource *cone =
11                    vtkConeSource::New();
12    cone->SetHeight( 3.0 );
13    cone->SetRadius( 1.0 );
14    cone->SetResolution( 5 );
15    vtkPolyDataMapper *coneMapper =
16                vtkPolyDataMapper::New();
17    coneMapper->SetInput(
18                     cone->GetOutput() );
19
20    vtkActor *coneActor =
21                          vtkActor::New();
22    coneActor->SetMapper( coneMapper );
23    vtkRenderer *ren1=
24                          vtkRenderer::New();
25    ren1->AddActor( coneActor );
26    ren1->SetBackground(0.1, 0.2, 0.4);
27    vtkRenderWindow *renWin =
28                    vtkRenderWindow::New();
29    renWin->AddRenderer( ren1 );
30    renWin->SetSize( 400, 400 );
31
32    vtkRenderWindowInteractor *iren =
33      vtkRenderWindowInteractor::New();
34    iren->SetRenderWindow(renWin);
35
36    iren->Initialize();
37    iren->Start();
38
39    return 0;
40  }
```

VTK: Programming - Makefile

```
1   CXX = g++ -g -Wall
2
3   # for queen VTK 4.2
4   #INCLUDES = -I. -I/home/pracow/putanowr/include/vtk
5   #LDFLAGS = -L/home/pracow/putanowr/lib/vtk \
6                                   -L/usr/X11R6/lib
7   #LDADD = -lvtkCommon -lvtkIO -lvtkGraphics \
8            -lvtkRendering -lGL -lX11 -lm
9
10  #for jinx VTK 3.2
11  INCLUDES = -I. -I/usr/include/vtk
12  LDFLAGS = -L/usr/lib/vtk -L/usr/X11R6/lib
13  LDADD = -lVTKCommon -lVTKGraphics -lGL \
14          -lXt -lX11 -lm
15
16  #for foo VTK 4.2
17  #INCLUDES = -I. -I/opt/include/vtk
18  #LDFLAGS = -L/opt/lib/vtk -L/usr/X11R6/lib
19  #LDADD = -lvtkCommon -lvtkIO -lvtkGraphics \
20           -lvtkRendering -lGL -lX11 -lm
21
22  %.o : %.cxx
23  $(CXX) -c $(INCLUDES) $< -o $@
24
25  Cone1 : Cone1.o
26  $(CXX) $^ $(LDFLAGS) $(LDADD) -o $@
27
28  .PHONY : clean distclean
29  clean :
30  /bin/rm -rf *.o
31  distclean :
32  rm -f Cone1
```

VTK: Programming - Tcl/Tk

```
1   package require vtk
2   package require vtkinteraction
3
4   vtkConeSource cone
5      cone SetHeight 3.0
6      cone SetRadius 1.0
7      cone SetResolution 5
8
9   vtkPolyDataMapper coneMapper
10     coneMapper SetInput [cone GetOutput]
11
12  vtkActor coneActor
13     coneActor SetMapper coneMapper
14
15  vtkRenderer ren1
16     ren1 AddActor coneActor
17     ren1 SetBackground 0.1 0.2 0.4
18
19  vtkRenderWindow renWin
20     renWin AddRenderer ren1
21     renWin SetSize 400 400
22
23  vtkRenderWindowInteractor iren
24     iren SetRenderWindow renWin
25     iren Initialize
26
27  wm withdraw .
```

Part IV

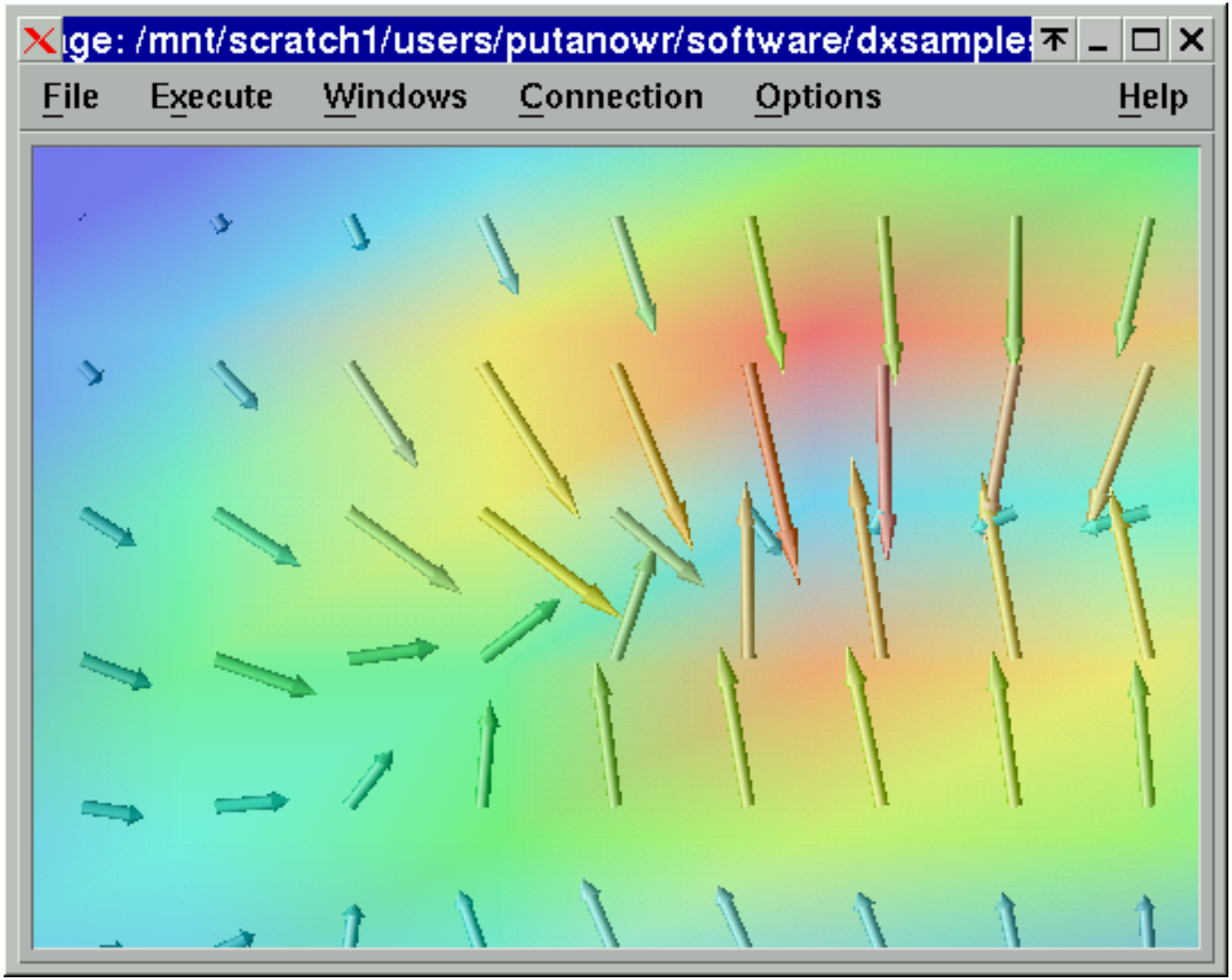
OpenDX Overview

OpenDX

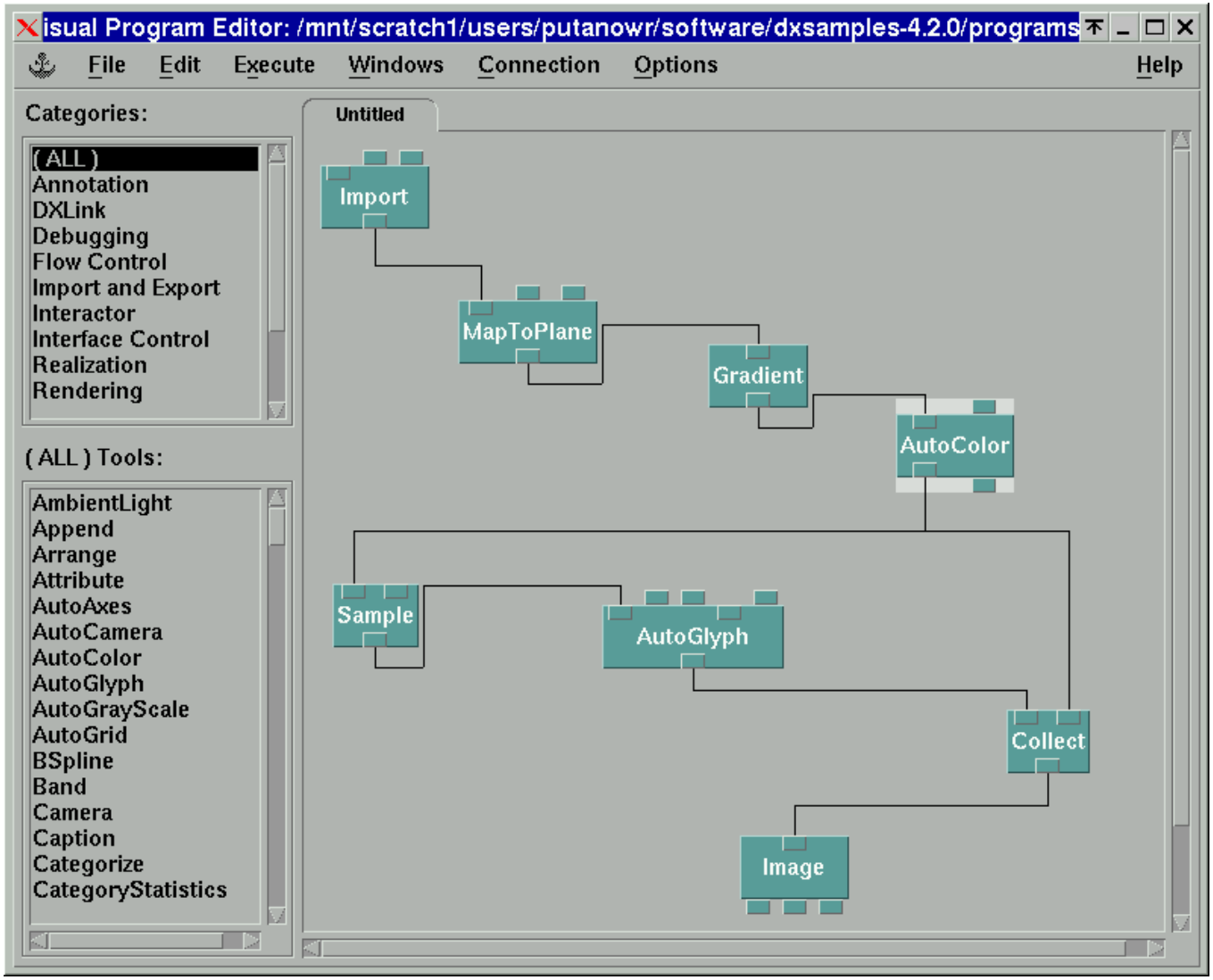
Open source software project based on IBM's Visualization Data Explored (DX)

- Powerful, full-featured visualization system
- Visual programming, advanced GUI
- Modular design, object-oriented, self-describing data model
- C implementation
- Scripting
- Python and Tcl/Tk wrappers

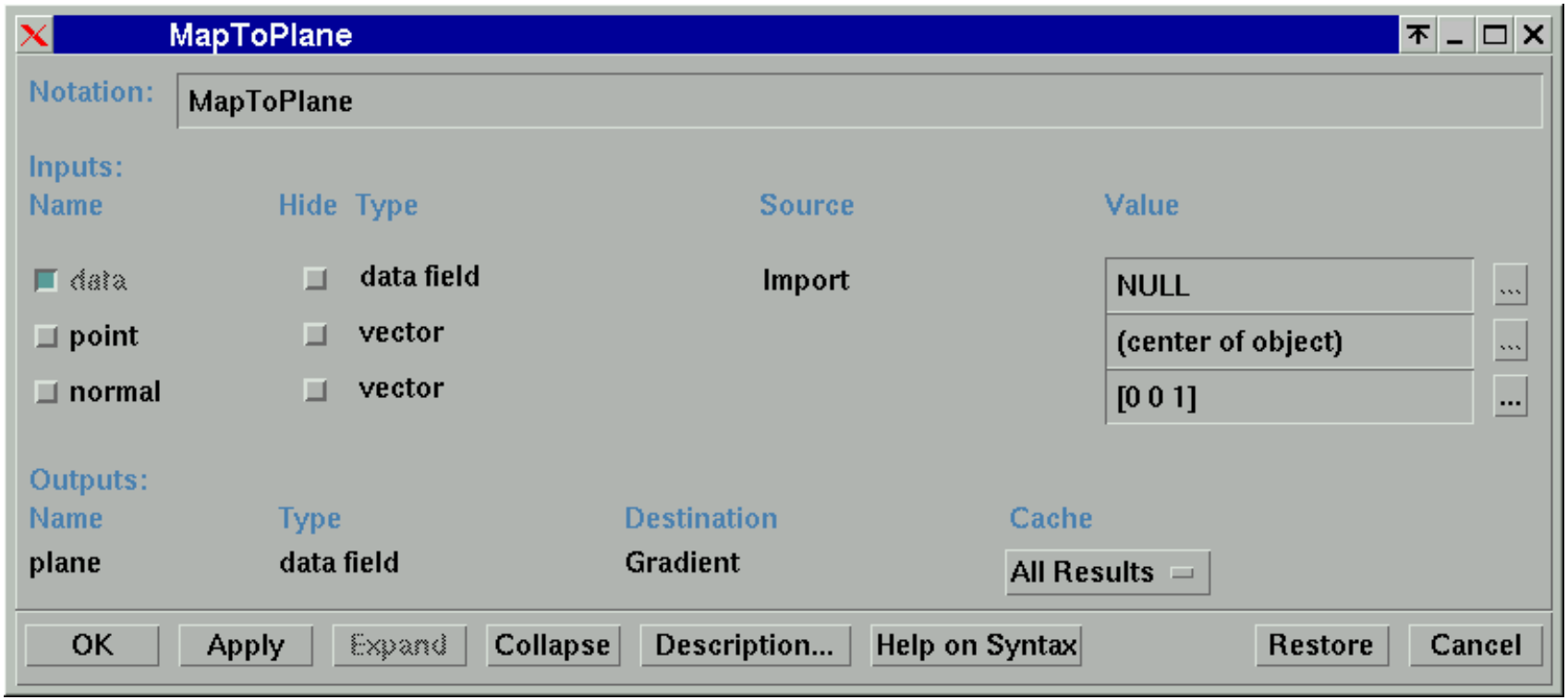
Visual Programs – Gradient



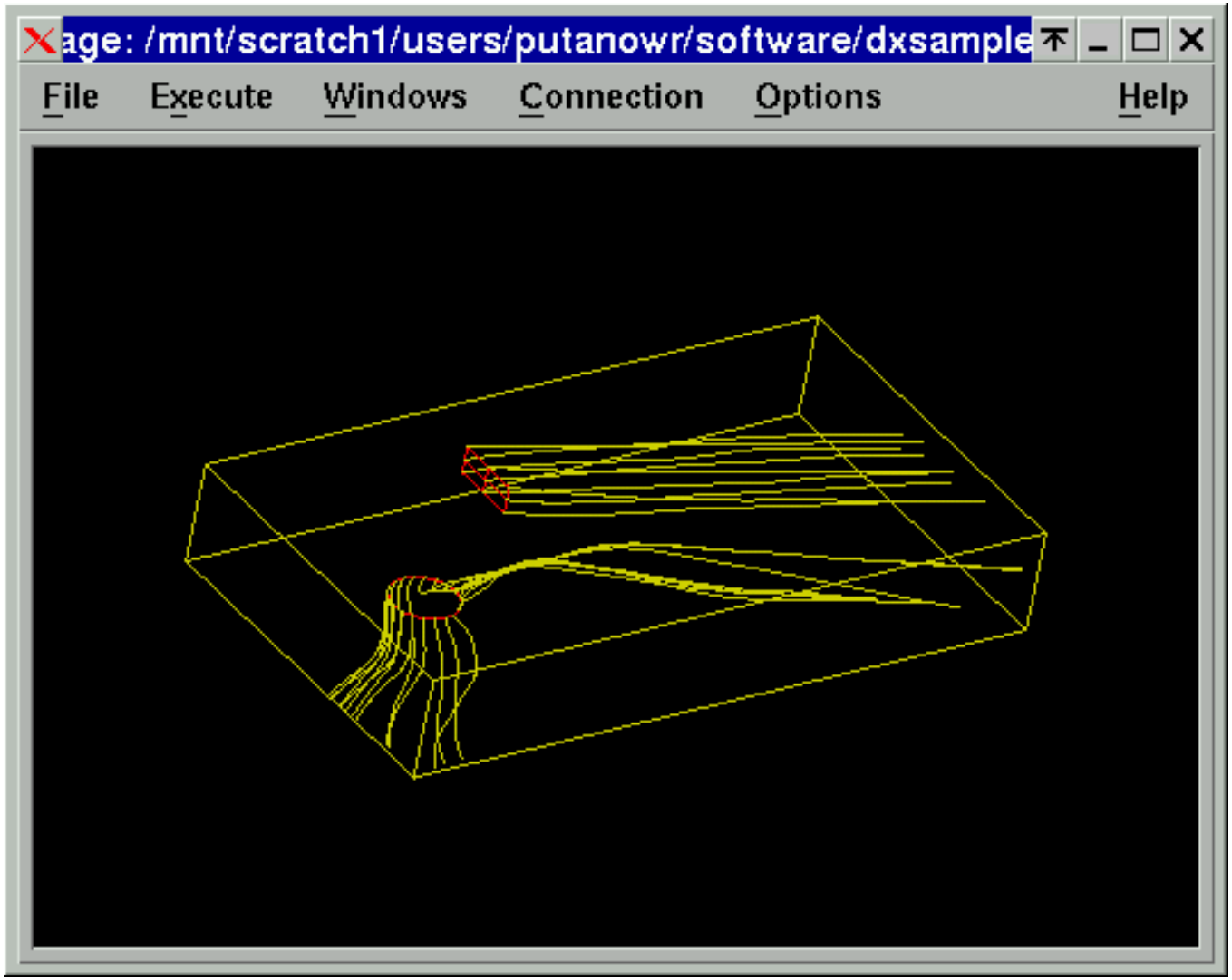
Visual Programs – Gradient



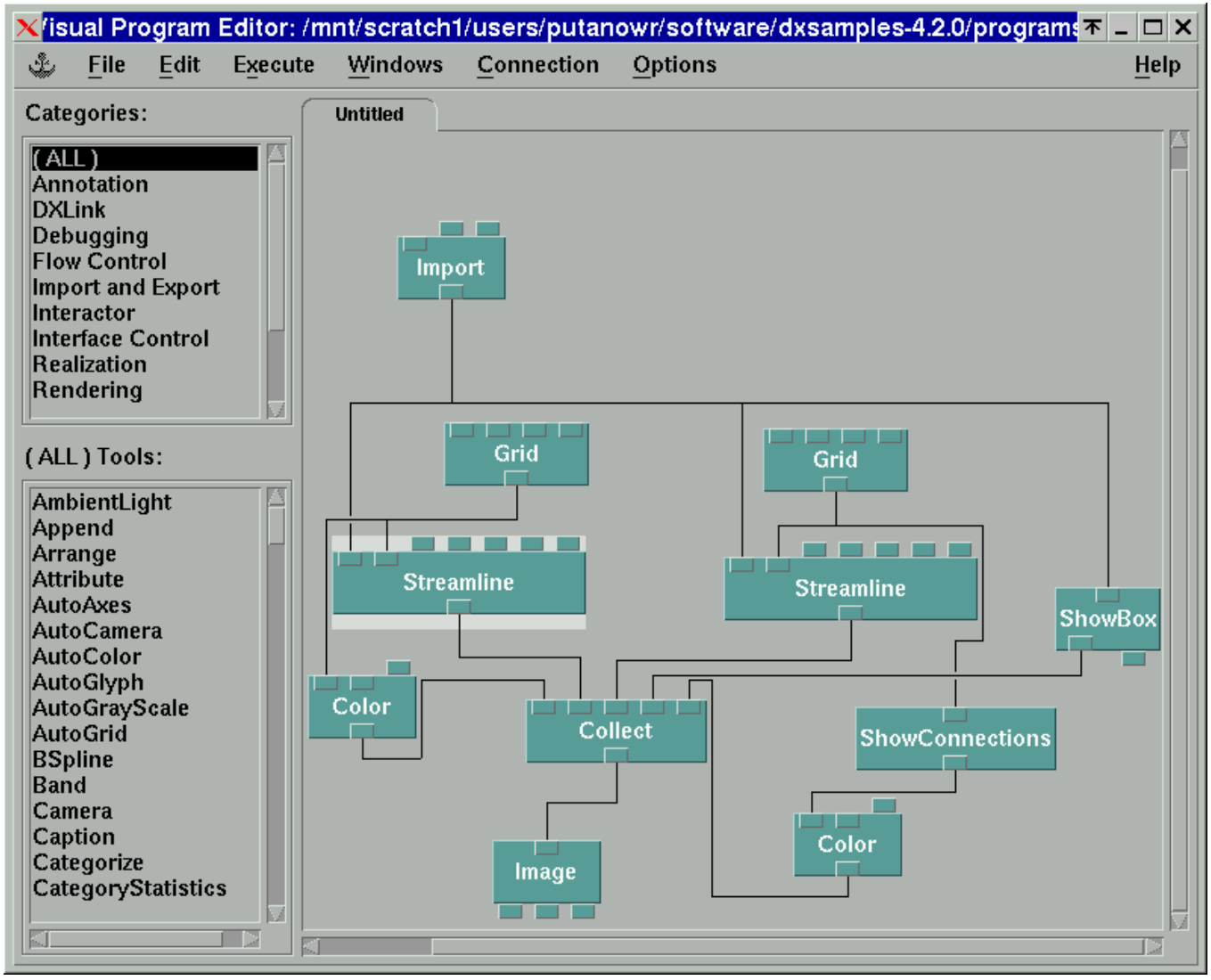
Visual Programs – Gradient



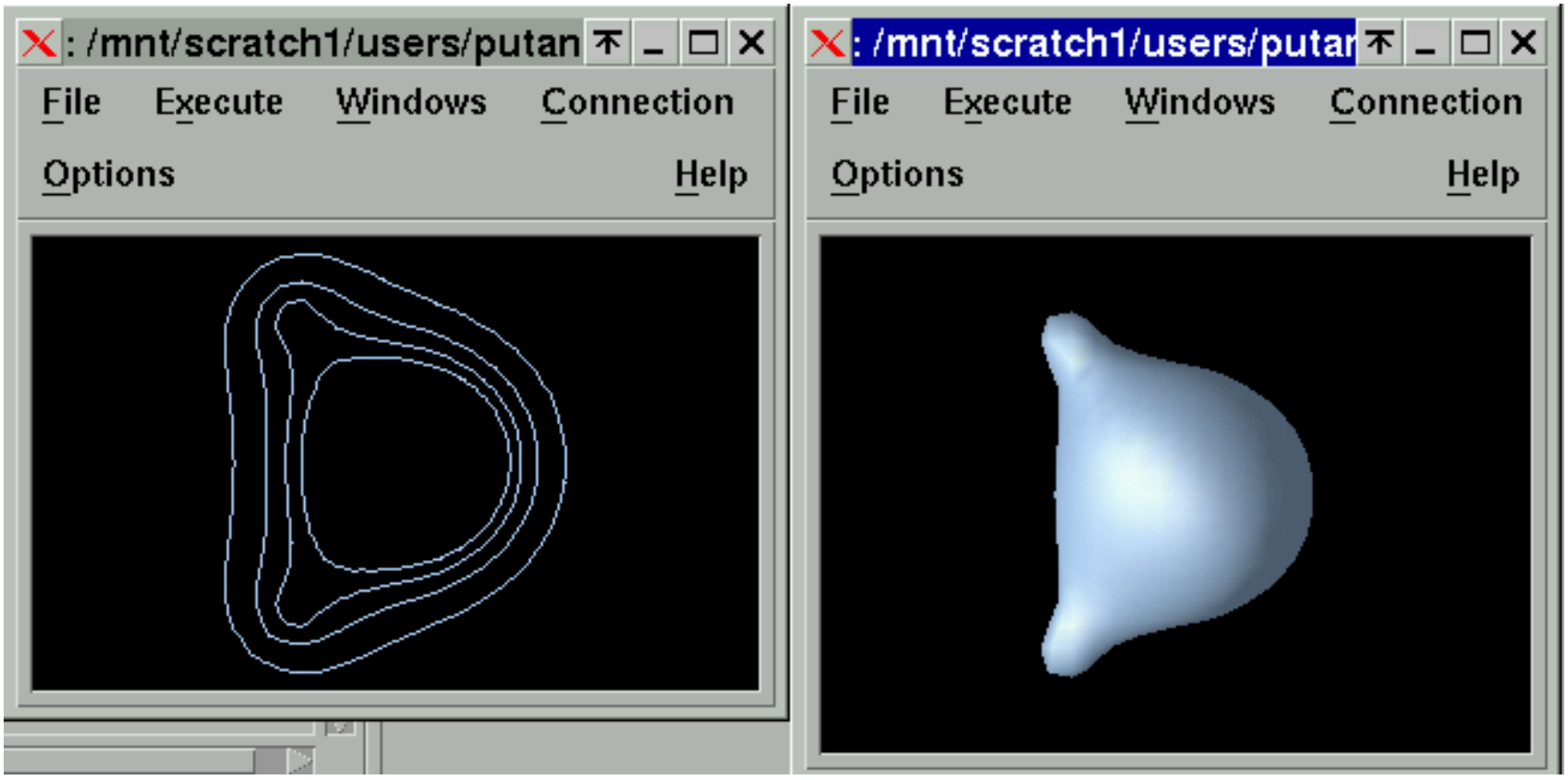
Visual Programs – Streamlines



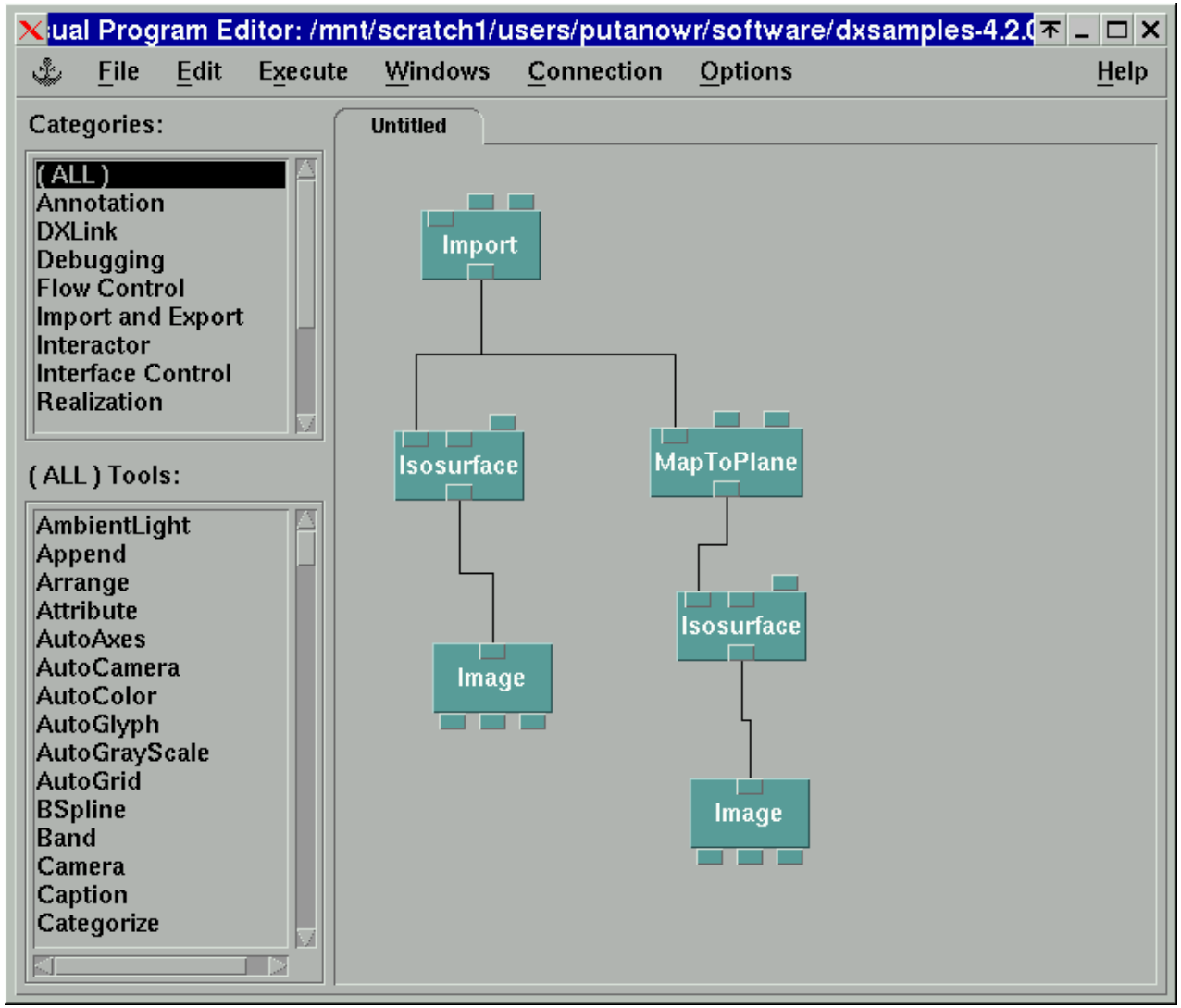
Visual Programs – Streamlines



Visual Programs – isosurfaces



Visual Programs – Isosurfaces



Credits

All figures in this presentation come from OpenDx distribution.

References

1. Gnuplot home page `http://www.gnuplot.info`
2. Gnuplot demo page `http://gnuplot.sourceforge.net/demo`
3. GNU Octave `http://www.gnu.org`
4. Octaviz `http://octaviz.sourceforge.net/`
5. VTK - The Visualization Toolkit `http://www.vtk.org`
6. OpenDX `http://www.opendx.org`
7. Own materials

Thank you for your attention

License