Mayavi2 tutorial

Prabhu Ramachandran and Gaël Varoquaux
1 Introduction to Mlab
1 Introduction to Mlab
1 Mlab: simple scripting for Mayavi2

from enthought.mayavi import mlab

- Simple problems should have simple solutions.
- Work interactively in IPython (-wthread).
- Work with numpy arrays.
- People know the pylab/matlab plotting API.

```python
x, y = ogrid[-10:10:100j, -10:10:100j]
r = sqrt(x**2 + y**2)
from enthought.mayavi import mlab
mlab.surf(sin(r)/r)
```
1 Mlab: plotting functions

0D data
mlab.points3d(x, y, z)

1D data
mlab.plot3d(x, y, z)

2D data
mlab.surf(z)
mlab.mesh(x, y, z)

3D data
mlab.contour3d(z)

Vector fields
mlab.quiver3d(x, y, z)
List of plotting functions in user guide.

Every plotting function has a test/demo function.

`mlab.surf` vs. `mlab.mesh`:
- `mlab.surf` = carpet plot:
  - image-like data + elevation
- `mlab.mesh` = orthogonal-grid, but general shape
Mlab: sprucing up the plot

```python
mlab.title('A title')
mlab.colorbar()
```

+ axes, outlines, text, ...

- Many keyword arguments for plotting functions.

**Gotcha: extents**

In VTK the extents are given by the data extents.

```python
x, y = mgrid[-10:10:100j, -10:10:100j]
r = sqrt(x**2 + y**2)
s = mlab.mesh(x, y, sin(r)/r, extent=(0,1,0,1,0,1))
mlab.outline(s, extent=(0,1,0,1,0,1))
mlab.axes(s, extent=(0,1,0,1,0,1))
```
Figures

- `mlab.figure()`: create a new figure or retrieve an existing figure.
- `mlab.clf()`: clear the current figure.
- `mlab.gcf()`: return the current figure.

**show and the event loop**

- GUI event loop needs to be running:
  - `mlab.show()` to display the visualization (after creating it).
- `@mlab.show`: decorator to make sure a function runs in the event loop.
mlab.show_pipeline()
We visualize the same data with two different methods:

**Visualizations = data sources + visualization modules**
Introduction to Mlab

Mlab: a more complex pipeline

30 000 points
1. Mlab: a more complex pipeline

30,000 points $\xrightarrow{\text{Delaunay2D}}$ Mesh
**Introduction to Mlab**

1. **Mlab: a more complex pipeline**

30,000 points → Delaunay2D → Mesh → Decimation → Irregular mesh

![Points](image1.png) → ![Mesh](image2.png) → ![Decimation](image3.png) → ![Irregular mesh](image4.png)
1 Mlab: a more complex pipeline

30,000 points → Delaunay2D → Mesh → Decimation → Irregular mesh

Complete pipeline
x, y, z = ogrid [-5:5:100j, -5:5:100j, -5:5:100j]

scalars = x*x*0.5 + y*y + z*z*2.0

obj = mlab.contour3d(scalars)
Adding a cut plane

```python
x, y, z = ogrid [-5:5:100j, -5:5:100j, -5:5:100j]

scalars = x*x*0.5 + y*y + z*z*2.0

obj = mlab.contour3d(scalars, opacity=0.5)

mlab.pipeline.scalar_cut_plane(obj)
```
mlab.options.backend = 'envisage'
⇒ mlab commands open up a full blown mayavi application.

By default, mlab uses a full blown application if open.
⇒ Use in interactive shell and mayavi2 -x foo.py