8th Advanced School on SCIENTIFIC VISUALIZATION

VTK Tutorial
Data structures, filtering and rendering

Stefano Perticoni – s.perticoni@scsitaly.com

Prerequisites

The following Python 2.7 and vtk 5.10 execution environment for Windows is recommended:

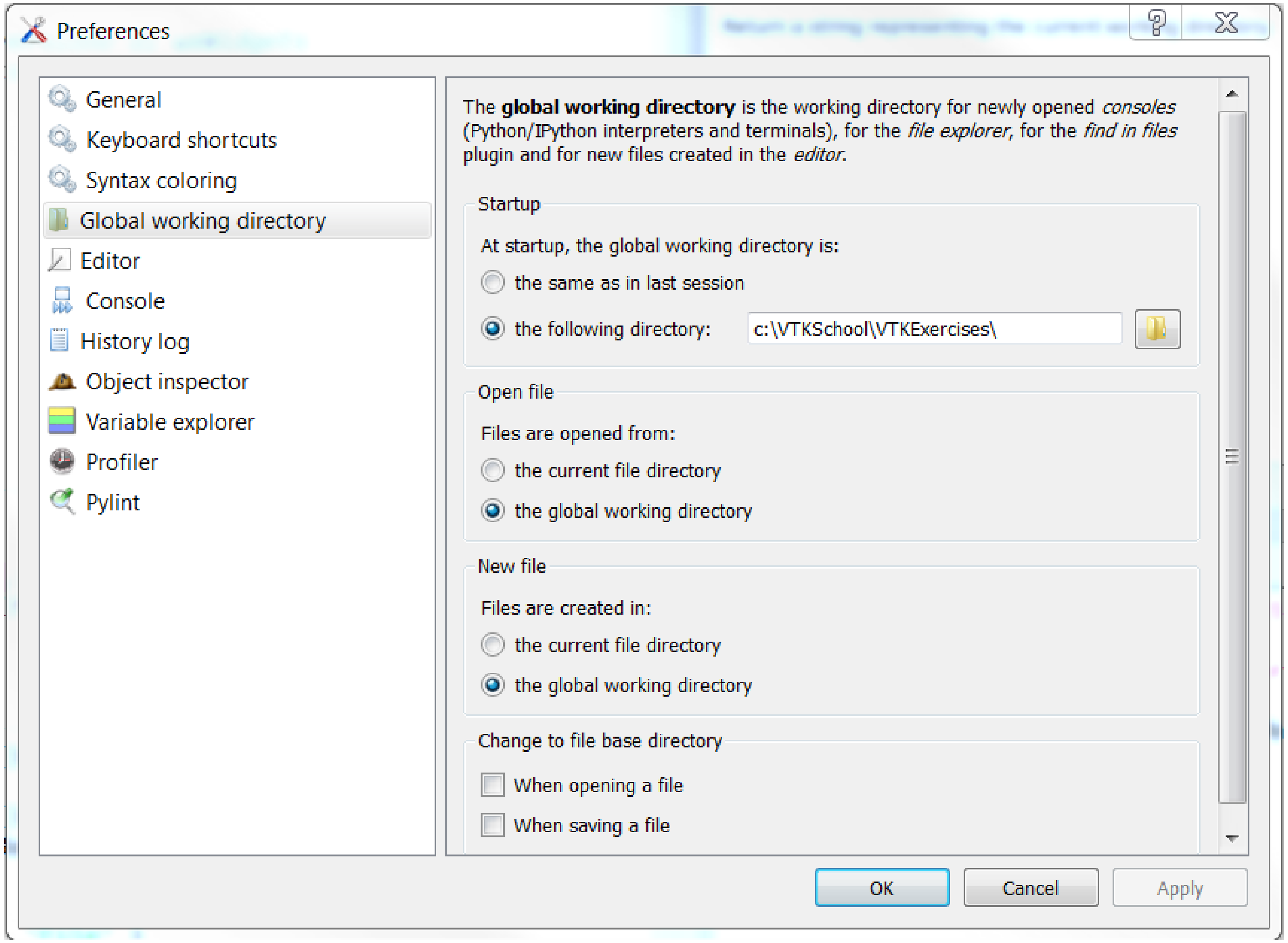
pythonxy
Scientific-oriented Python Distribution based on Qt and Spyder

http://ftp.ntua.gr/pub/devel/pythonxy/Python(x,y)-2.7.3.1.exe

Otherwise download a pure Python 2.7/ VTK 5.8 execution environment for your Win/Mac/Linux system from:

http://vtk.org/files/release/5.8/tmp/

Spyder set up



The **global working directory** is the working directory for newly opened *consoles* (Python/IPython interpreters and terminals), for the *file explorer*, for the *find in files* plugin and for new files created in the *editor*.

Startup

At startup, the global working directory is:
- the same as in last session
- [✓ the following directory:] c:\VTKSchool\VTKExercises\

Open file

Files are opened from:
- the current file directory
- [✓ the global working directory]

New file

Files are created in:
- the current file directory
- [✓ the global working directory]

Change to file base directory
- ☐ When opening a file
- ☐ When saving a file

Spyder set up

1. Open Spyder: Just write Spyder in Windows search box

   Spyder

2. Tools → Preferences → Global working directory:
   Set the global working directory to c:\YourUnzipDir\

3. Restart Spyder

4. Open a python shell: Interpreters → Open Interpreter
and check the path:

```
>>> os.getcwd()
'c:\\YourUnzipDir\\'
```

If this is the result your environment is OK

Exercise: learn vtkArray

1 Make an array

```
myArray = vtk.vtkDoubleArray()
list_dir(myArray)
help(myArray.SetValue)
print(myArray)
myArray.SetName('my first array')
myArray.SetNumberOfComponents(1)
myArray.SetNumberOfTuples(500*500) #going to make a 500x500 picture
```

2 Fill it with data

```
from math import sin, cos
for x in range(0,500):
    for y in range(0,500):
        myArray.SetValue(x*500+y, 127.5+(1.0+sin(x/25.0)*cos(y/25.0)))
```

Exercise: learn vtkArray

1. Create the Data structure

```
id = vtk.vtkImageData()
```

2. Define its Geometry

```
id.SetOrigin(0,0,0)
id.SetSpacing(1,1,1)
```

3. Define its Topology

```
id.SetDimensions(500,500,1)
```

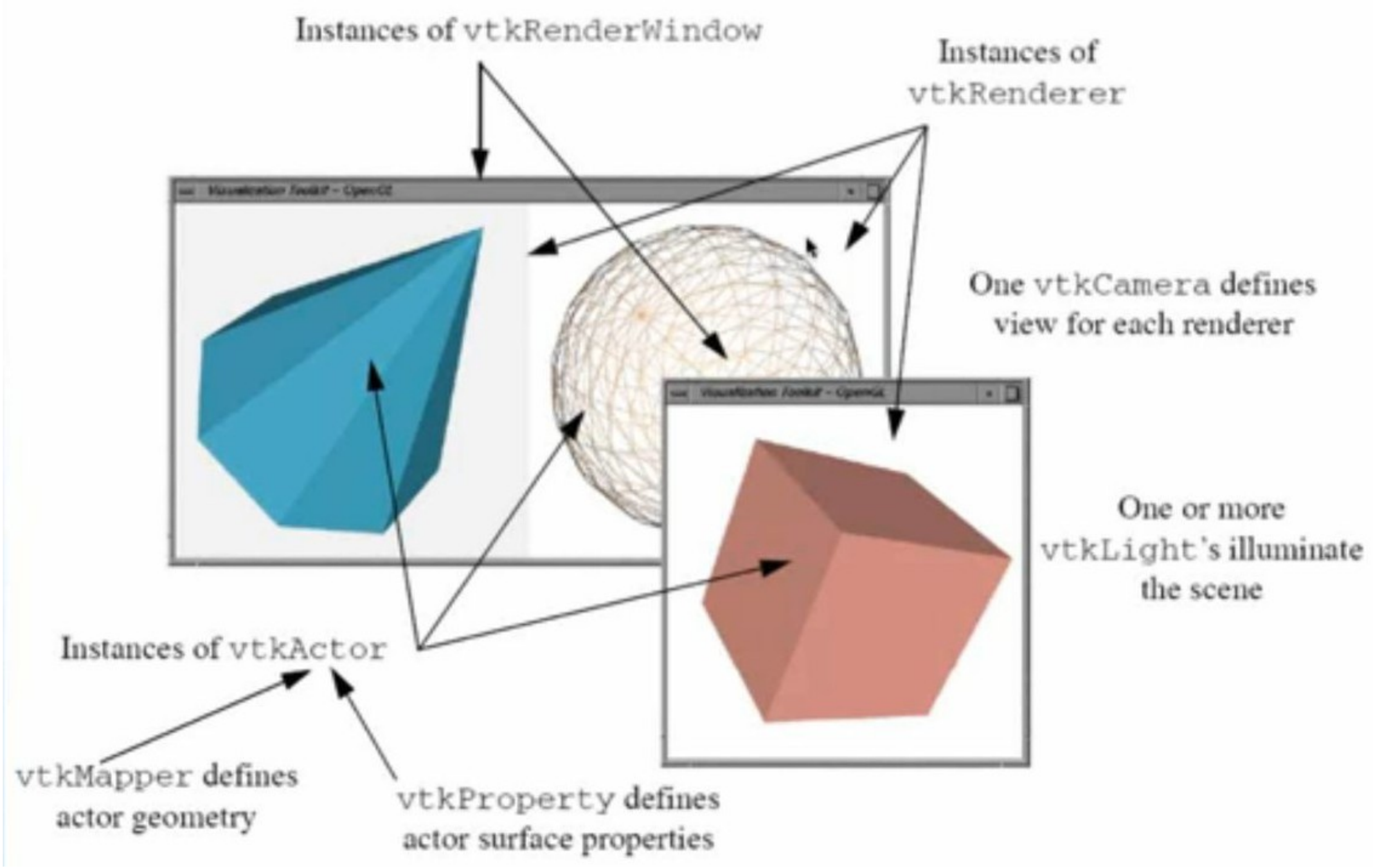
4. Assign Data to the Structure, Geometry and/or Topology

```
id.SetScalarType(vtk.VTK_DOUBLE)
id.GetPointData().SetScalars(myArray)
```

5. Inspect it

```
print(id)
print(id.GetPointData())
array = id.GetPointData().GetArray('my first array')
array.GetRange()
```

The VTK Graphics Subsystem


Instances of vtkRenderWindow
Instances of vtkRenderer
One vtkCamera defines view for each renderer
One or more vtkLight's illuminate the scene
Instances of vtkActor
vtkMapper defines actor geometry
vtkProperty defines actor surface properties

vtkRenderWindow

- SetSize() — set the size of the window
- AddRenderer() — add another renderer which draws into this
- SetInteractor() — set class to handles mouse/key events
  - vtkRenderWindowInteractor->SetInteractorStyle()
- Render() — updates pipeline and draws scene

vtkRenderer

- SetViewport() - specify where to draw in the render window
- SetLayer() - set pane/depth in render window to draw on

- AddViewProp() - add objects to be rendered
- AddLight() - add a light to illuminate the scene
- SetAmbient() - set the intensity of the ambient lighting
- SetBackground() - set background color

- SetActiveCamera() - specify the camera to use to render the scene
- ResetCamera() - reset the camera so that all actors are visible

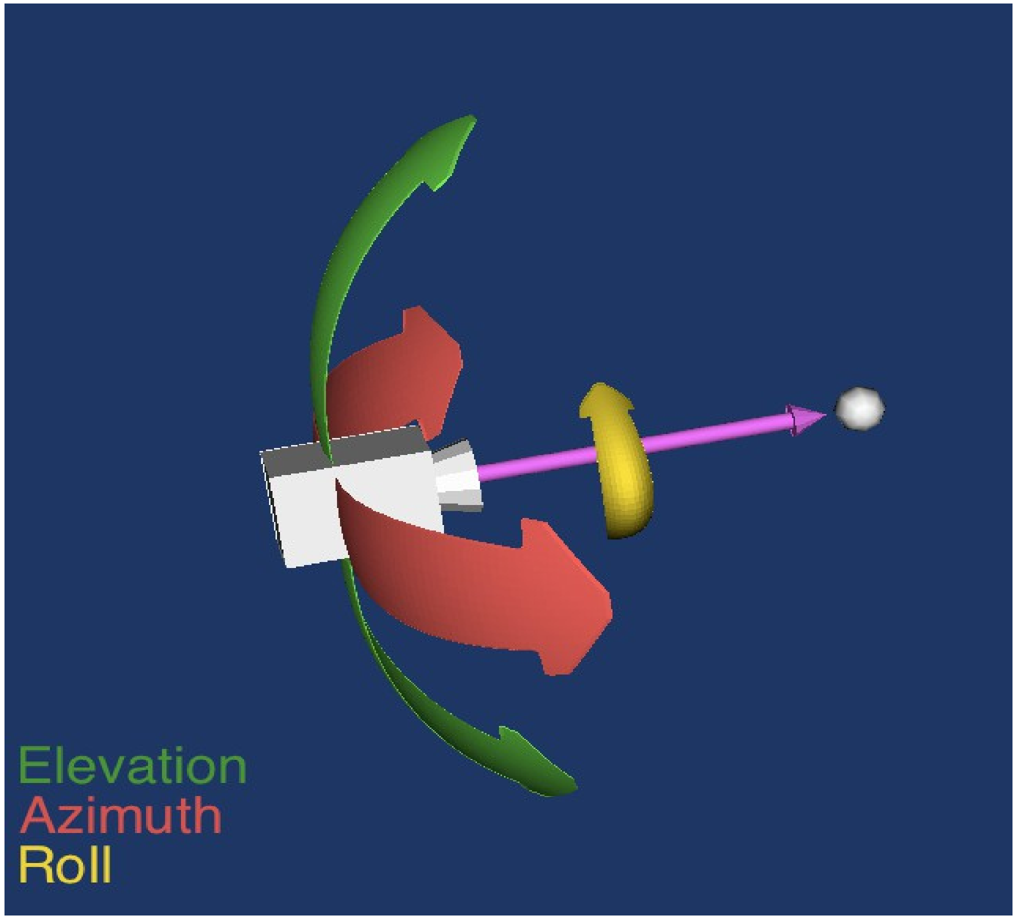
vtkCamera

- Position - where the camera is located
- FocalPoint - where the camera is pointing
- ViewUp - which direction is "up"
- ClippingRange - data outside of this range is clipped
- ViewAngle - the camera view angle controls perspective effects
- ParallelProjection - turn parallel projection on/off (no perspective effects)

- Roll, Pitch, Yaw, Elevation, Azimuth
  move the camera in a variety of ways

- Zoom, Dolly - changes view angle (Zoom);
  move camera closer (Dolly)

vtkCamera



vtkActor (subclass of vtkProp)

- Visibility - is the actor visible?
- Pickable - is the actor pickable?

- Texture - a texture map associated with the actor
- SetOrigin/Scale/UserTransform - control where it is drawn
- GetBounds

- vtkProperty - surface lighting properties

Exercise: make a window

1. Make a window

```
renwin = vtk.vtkRenderWindow()
renwin.SetSize(500,500)
```

2. Make a renderer for that window

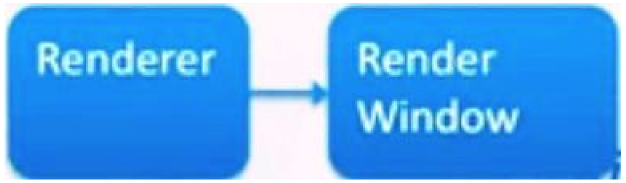
```
renderer = vtk.vtkRenderer()
renwin.AddRenderer(renderer)
```

3. Control how it all looks

```
renderer.SetBackground2(1,1,1)
renderer.SetGradientBackground(1)
```

4. Show it

```
renwin.Render()
```



Exercise: show some data

1. Access the data processing pipeline that has your data

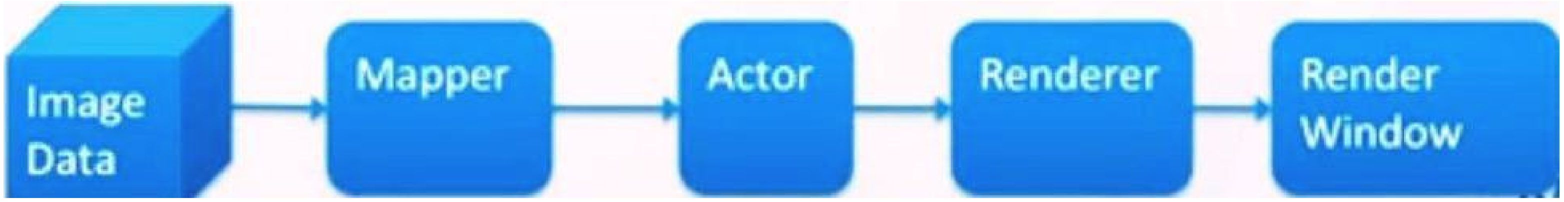
```
mapper = vtk.vtkDataSetMapper()
mapper.SetInput(id)
mapper.ScalarVisibilityOff() # we'll talk about this soon
```

2. Link that to the display system

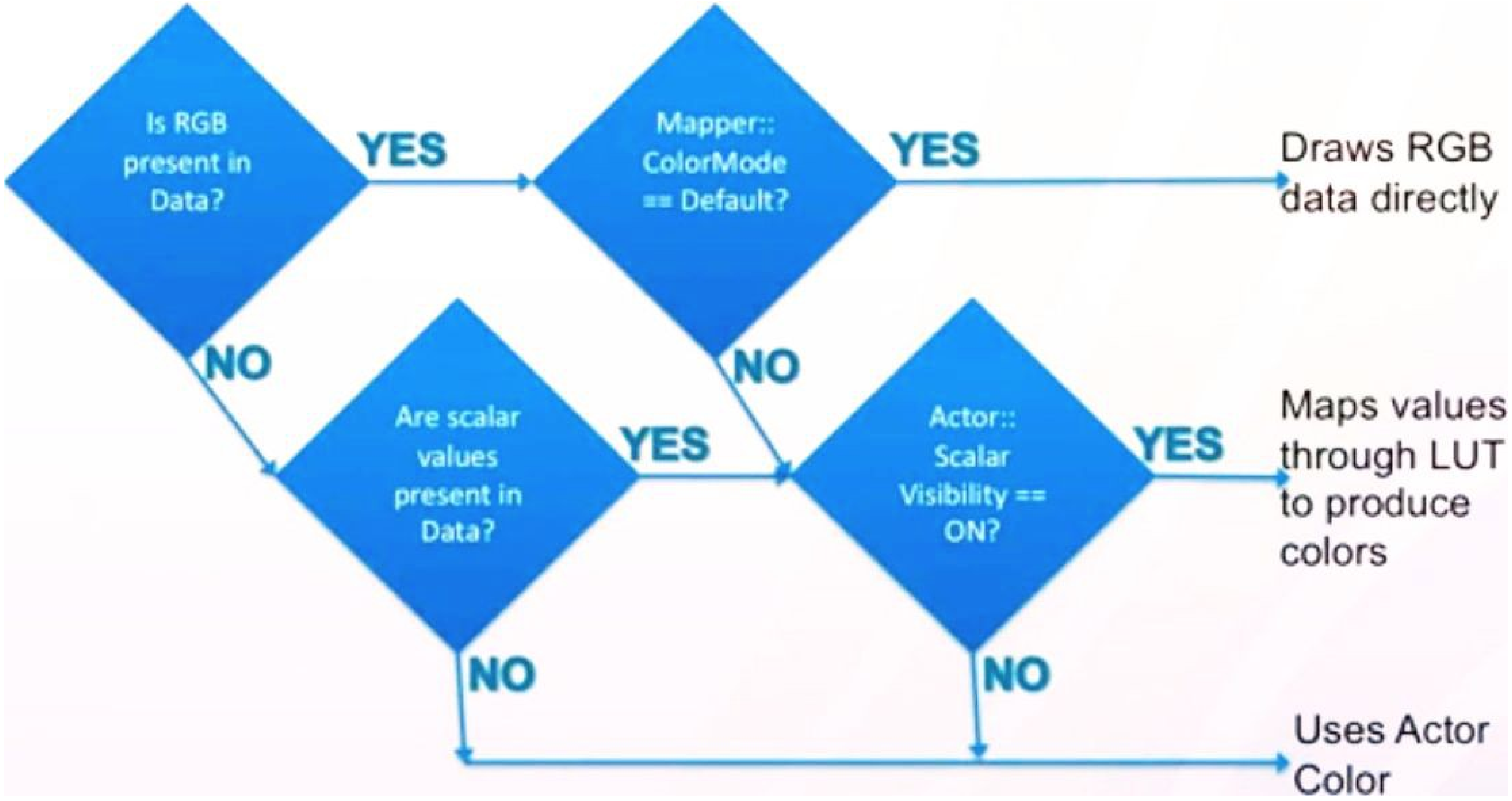
```
actor = vtk.vtkActor()
actor.SetMapper(mapper)
renderer.AddViewProp(actor)
renwin.Render()
```

3. Adjust the camera for a better view

```
renderer.ResetCamera()
renwin.Render()
```



Color control by vtkActor and vtkMapper


Is RGB present in Data?
YES
Mapper:: ColorMode == Default?
YES
Draws RGB data directly
NO
Are scalar values present in Data?
YES
NO
Actor:: Scalar Visibility == ON?
YES
Maps values through LUT to produce colors
NO
NO
Uses Actor Color

vtkProperty (Actor has)

- AmbientColor, DiffuseColor, SpecularColor — a different color for ambient, diffuse, and specular lighting
- Color — sets the three colors above to the same
- Interpolation - shading interpolation method (Flat, Gouraud)

- Representation — how to represent itself (Points, Wireframe, Surface)

- Opacity — control transparency

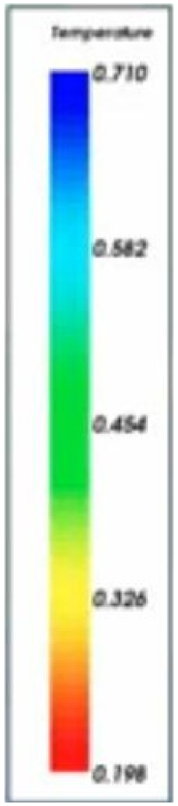
vtkMapper (Actor also has)

- ScalarVisibilityOn()/Off()
  - Color cells/points by data values or entire object by actor color
- Choose which array to color by
  - SetScalarModeToDefault()
  - SetScalarModeToUsePointData()
  - SetScalarModeToUseCellData()
  - SelectColorArray(array name)
- SetLookupTable(lut)
- SetScalarRange(min, max)
  - range of data values for lut
- InterpolateScalarBeforeMappingOn()/Off()
  - whether to interpolate colors across cells in color or data space

vtkLookupTable (Mapper has)

- NumberOfColors - number of colors in the table
- TableRange - the min/max scalar value range to map
- If building a table from linear HSVA ramp:
  - HueRange - mm/max hue range
  - SaturationRange - min/max saturation range
  - ValueRange - min/max value range
  - AlphaRange - min/max transparency range

- If manually building a table
  - Build (after setting NumberOfColors)
  - SetTableValue(idx, rgba) for each NumberOfColors entries



Exercise : Visualize the topology

1. Specify whole Prop color

```
actorProperty = actor.GetProperty()
actorProperty.SetDiffuseColor(0,1,1)
renwin.Render()
```

2. Change from surface to edges rendering

```
actorProperty.SetRepresentationToWireframe()
renwin.Render()
renderer.GetActiveCamera().Zoom(10)
renwin.Render()
```

3. Reset

```
actorProperty.SetRepresentationToSurface()
renderer.ResetCamera()
```

Exercise : Visualize the topology

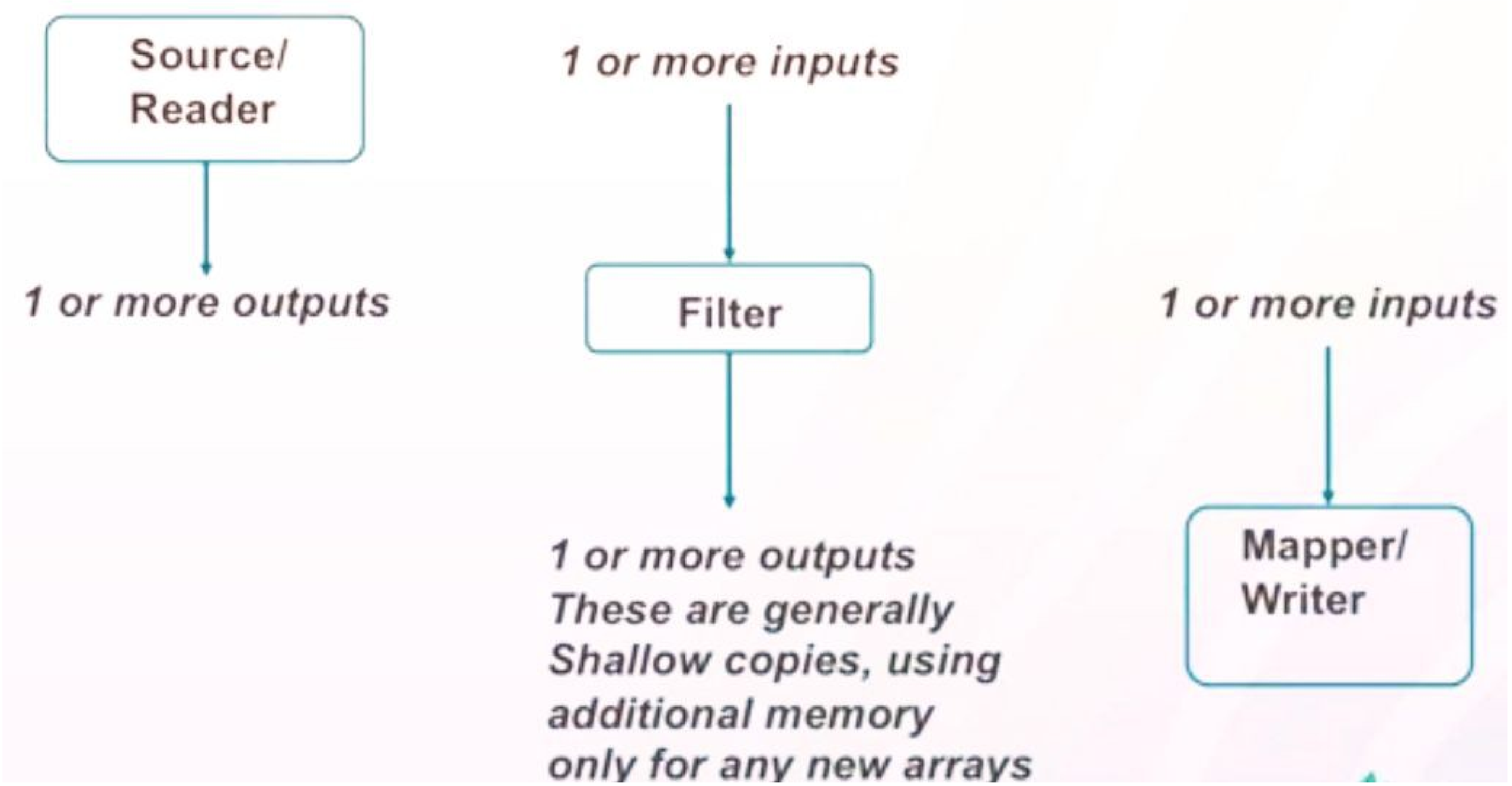
1. Turn on color from values

```
mapper.ScalarVisibilityOn()
renwin.Render()
```

2. Match up lookuptable range

```
myArray.GetRange()
mapper.SetScalarRange(127,129)
renwin.Render()
```

Algorithms


Source/ Reader
1 or more outputs
1 or more inputs
Filter
1 or more outputs
These are generally
Shallow copies, using
additional memory
only for any new arrays
1 or more inputs
Mapper/ Writer

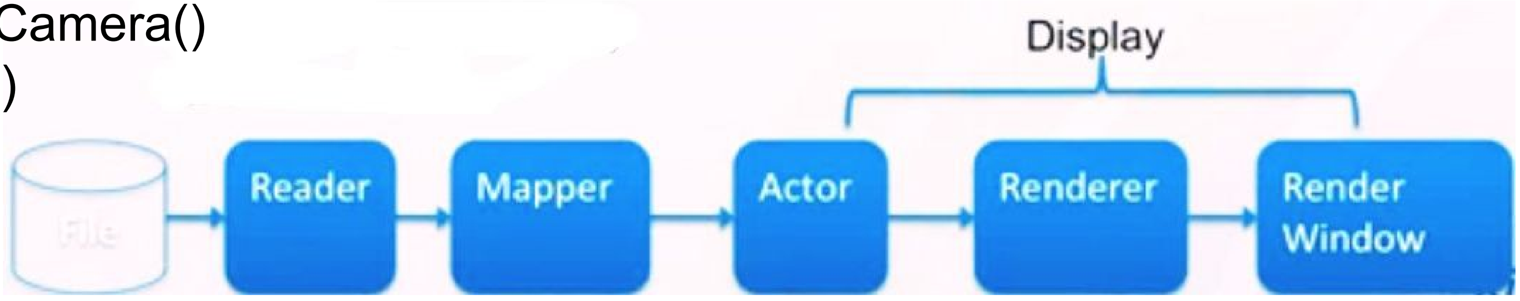
Read a data file, inspect and visualize

1. Create a reader, tell it what file and run it

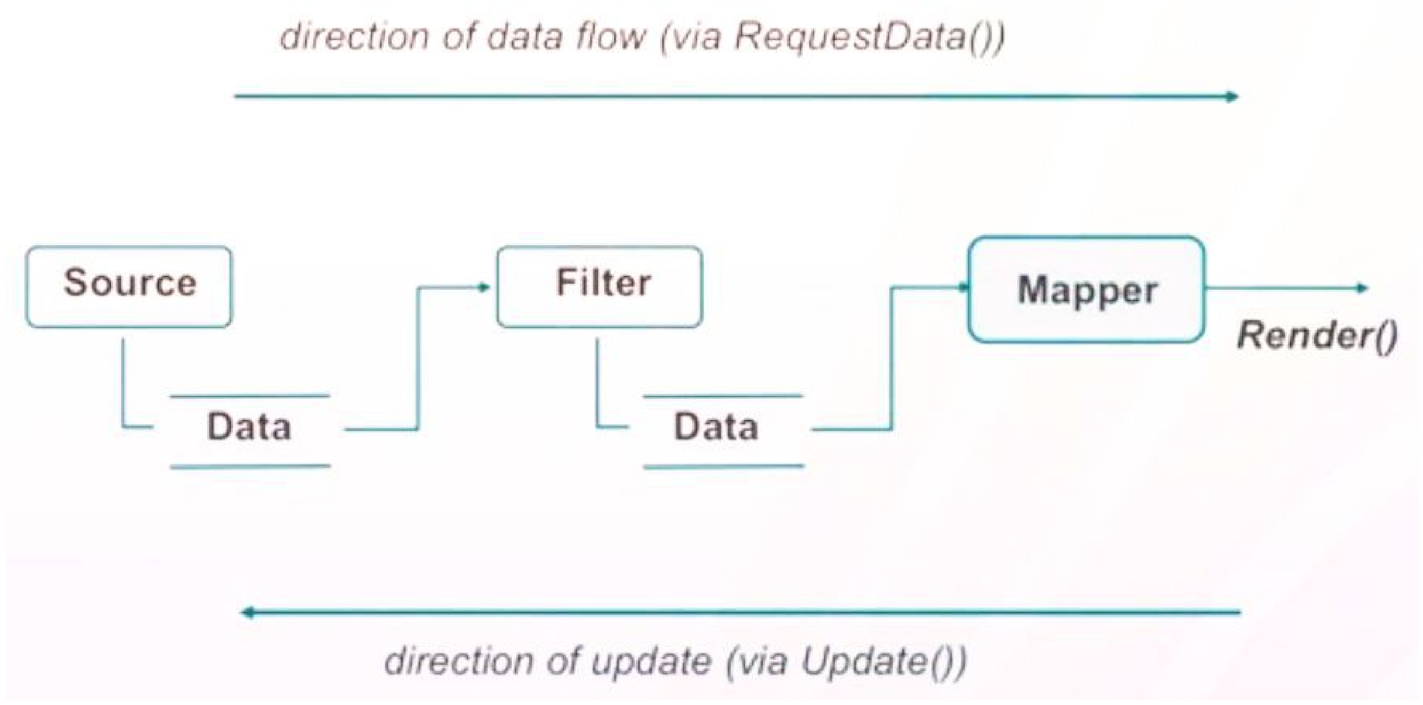
```
reader = vtk.vtkDataSetReader()
reader.SetFileName("c:/Devel/VTKSchool/codeAndData/SaintHelens.vtk")
```

2. Examine the result

```
id = reader.GetOutput()
print id.GetPointData().GetArray(0)
reader.Update()
print id.GetPointData().GetArray(0).GetRange()
mapper.SetInputConnection(reader.GetOutputPort())
mapper.SetScalarRange(682.0, 2543.0)
renwin.Render()
renderer.ResetCamera()
renwin.Render()
```



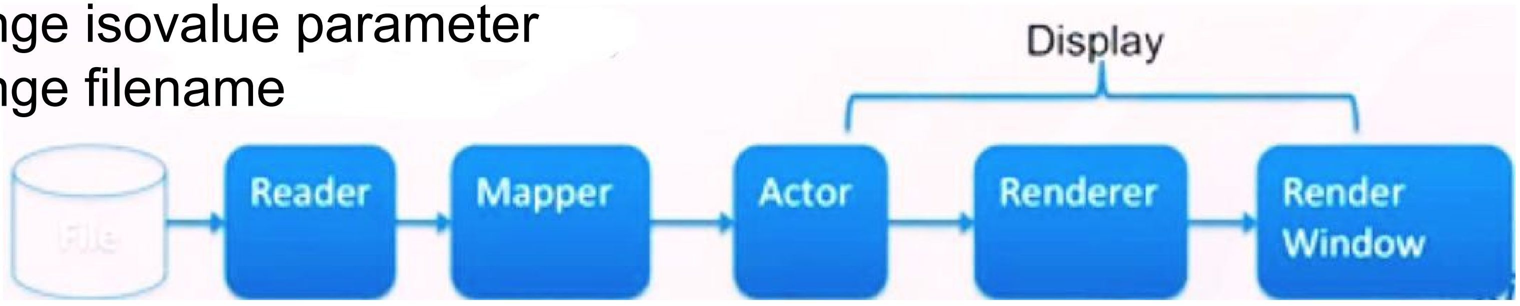
Pipeline execution model


direction of data flow (via RequestData())
Source
Data
Filter
Data
Mapper
Render()
direction of update (via Update())

Demand Driven Pipeline

- Lazy evaluation
  - Pipeline only produces results when you ask it to Update or Render()
  - Changing a parameter or rearranging the pipeline doesn't do that.
  - Each filter caches its most recent output

- Modified time
  - Each filter keeps track of when it last produced data, and when its parameters were last changed
  - Pipeline only updates as far back as it has to
  - Examples:
    - Camera motion - data isn't reread, only mapper has to execute
    - Change isovalue parameter
    - Change filename



Exercise : manipulate the read in data

1. Make filter to convert to a less constrained data structure

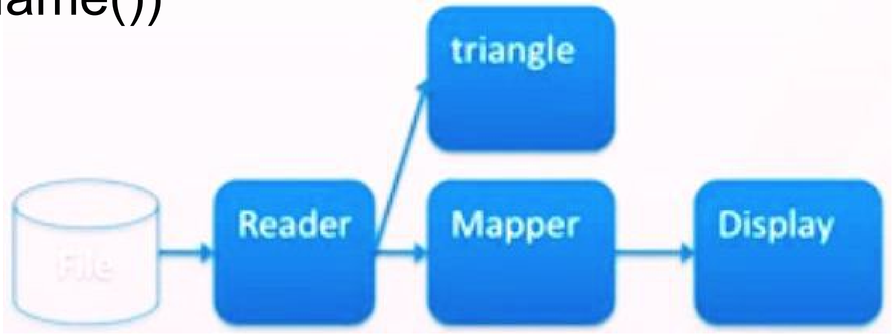
```
triangles = vtk.vtkDataSetTriangleFilter()
```

2. Connect it

```
triangles.SetInputConnection(reader.GetOutputPort())
```

3. Run it

```
triangles.Update()
print(reader.GetOutput().GetClassName())
print(triangles.GetOutput().GetClassName())
```



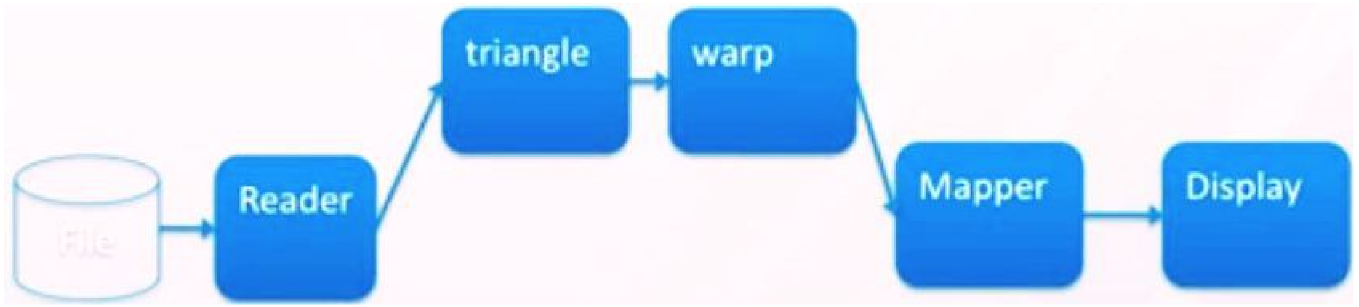
Exercise: manipulate the read in data

1. Make and use a filter to change the geometry

```
warp = vtk.vtkWarpScalar()
warp.SetInputConnection(triangles.GetOutputPort())
warp.Update()
print(triangles.GetOutput().GetBounds())
print(warp.GetOutput().GetBounds())
```

2. Show it

```
mapper.SetInputConnection(warp.GetOutputPort())
renwin.Render()
```



Exercise: manipulate the data

1. Make a clip filter and put it in pipeline

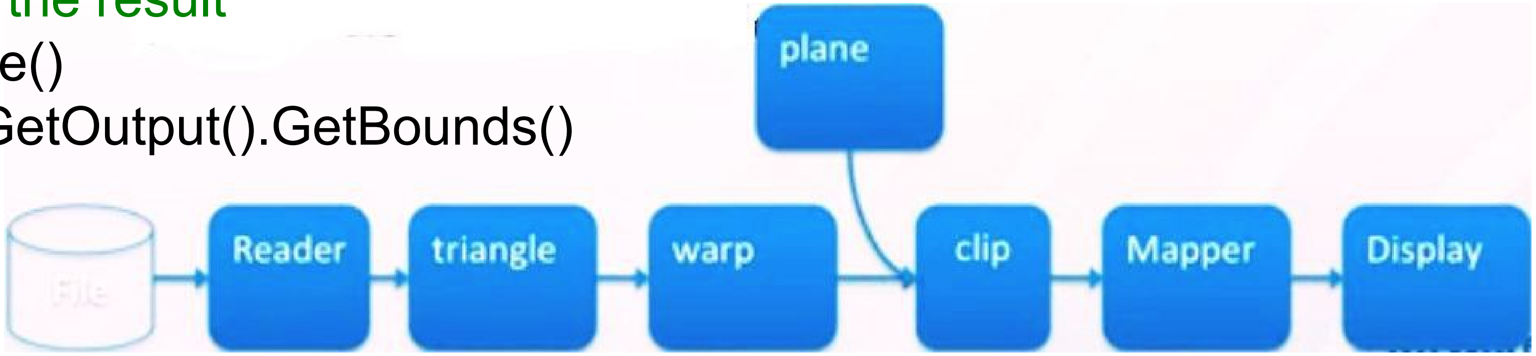
```
clip = vtk.vtkClipDataSet()
clip.SetInputConnection(warp.GetOutputPort())
mapper.SetInputConnection(clip.GetOutputPort())
```

2. Make a source to orient clip filter with

```
plane = vtk.vtkPlane()
clip.SetClipFunction(plane)
plane.SetOrigin(560000,5120000,2000)
```

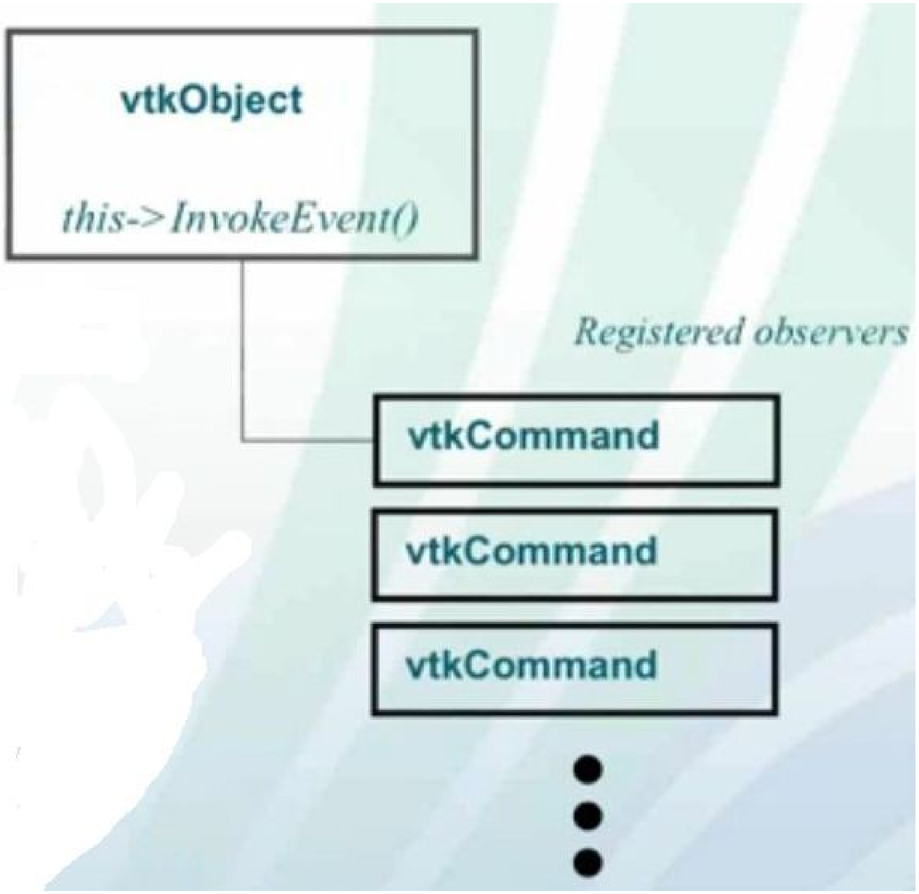
3. Inspect the result

```
clip.Update()
print clip.GetOutput().GetBounds()
```



Interaction

- Events
  - Instances of vtk classes can fire events and watch events fired by others
  - watcher executes some code whenever the event occurs
- Interactors
  - Watch mouse, keyboard, window system events to move camera call render etc
- Widgets
  - Special purpose classes that are drawn in scene and watch events



Exercise: use a widget to interact with the data

1. Get a hold of window events

```
iren = vtk.vtkRenderWindowInteractor()
renwin.SetInteractor(iren)
```

2. Make and initially place the widget

```
widget = vtk.vtkImplicitPlaneWidget()
widget.PlaceWidget(warp.GetOutput().GetBounds())
widget.SetOrigin([plane.GetOrigin()[x] for x in 0,1,2])
widget.SetNormal([plane.GetNormal()[x] for x in 0,1,2])
```

3. Connect it to the renderwindow's events

```
widget.SetInteractor(iren)
```

Exercise: use a widget to interact with the data

1. Connect the widget's events to our pipeline

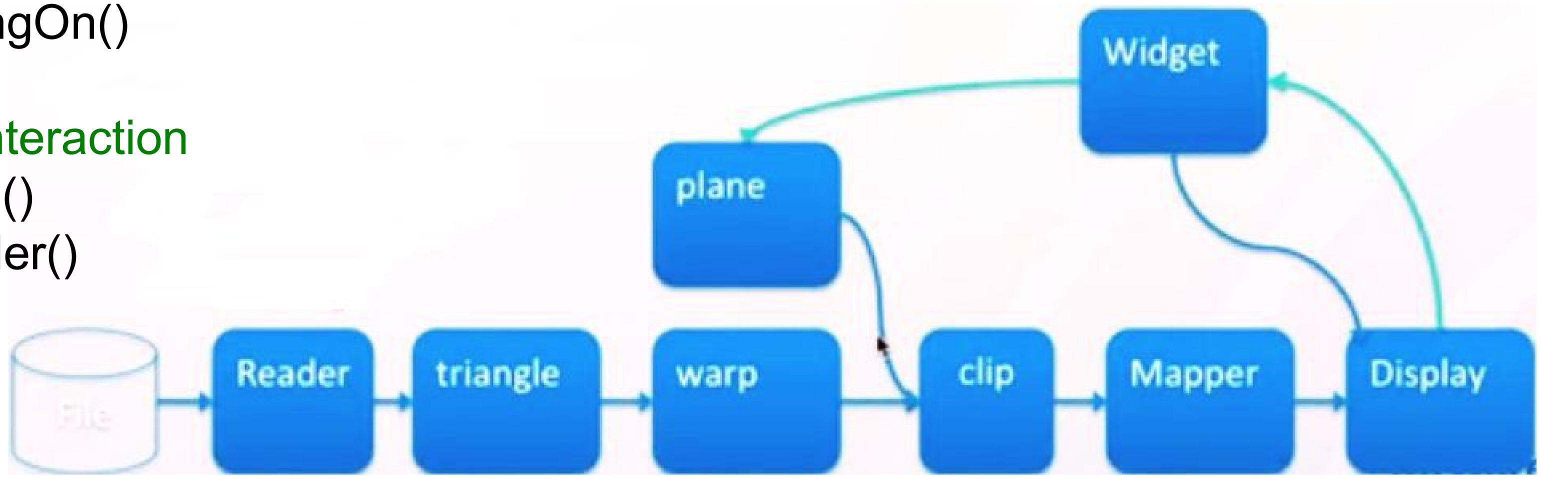
```
def eventhandler(obj , event):
        global plane
        obj.GetPlane(plane)

widget.AddObserver("InteractionEvent", eventhandler)
```

2. Configure the widget

```
widget.SetEnabled(1)
widget.DrawPlaneOn()
widget.TubingOn()
```

3. Turn on interaction

```
iren.Initialize()
renwin.Render()
iren.Start()
```



Tutorial: Import timevarying data, visualize it and export as Unstructured Grid

Advanced School on SCIENTIFIC VISUALIZATION

Tutorial: Create a cone and scale it with a 3D widget

CINECA

Advanced School on SCIENTIFIC VISUALIZATION

Exercise: Import timevarying data, visualize it and export as Structured Points

CINECA

Summer School on SCIENTIFIC VISUALIZATION

CINECA