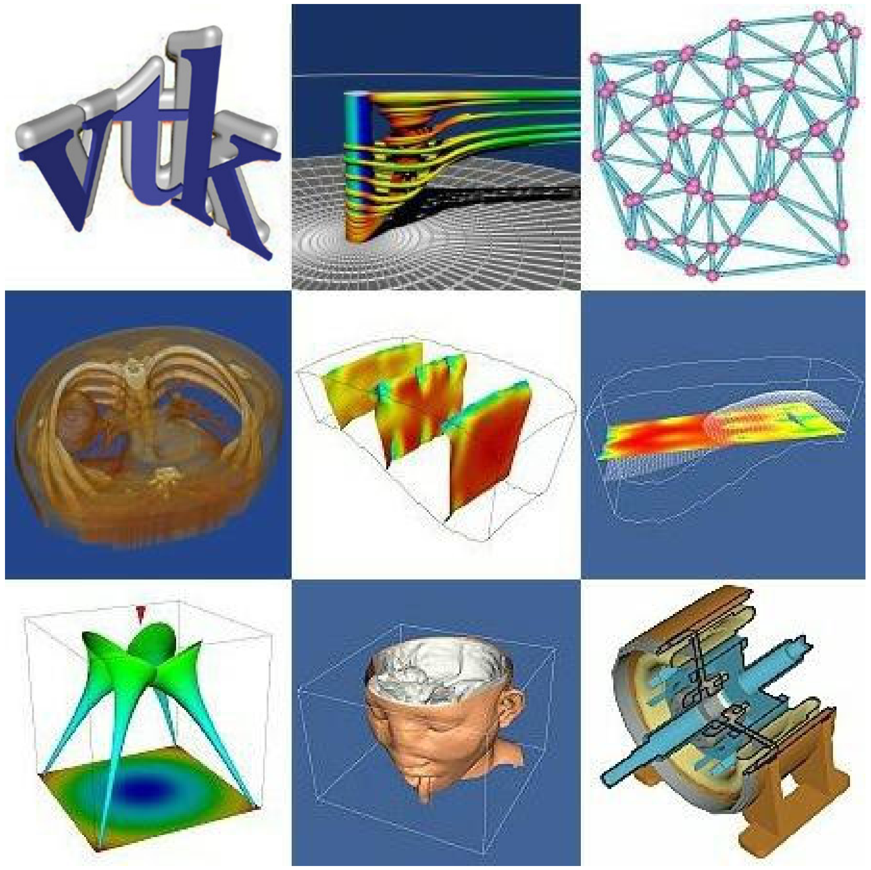
VTK Introduction

Goal: get familiar with VTK visualization pipeline; set up python+vtk environment for the coming programming assignments

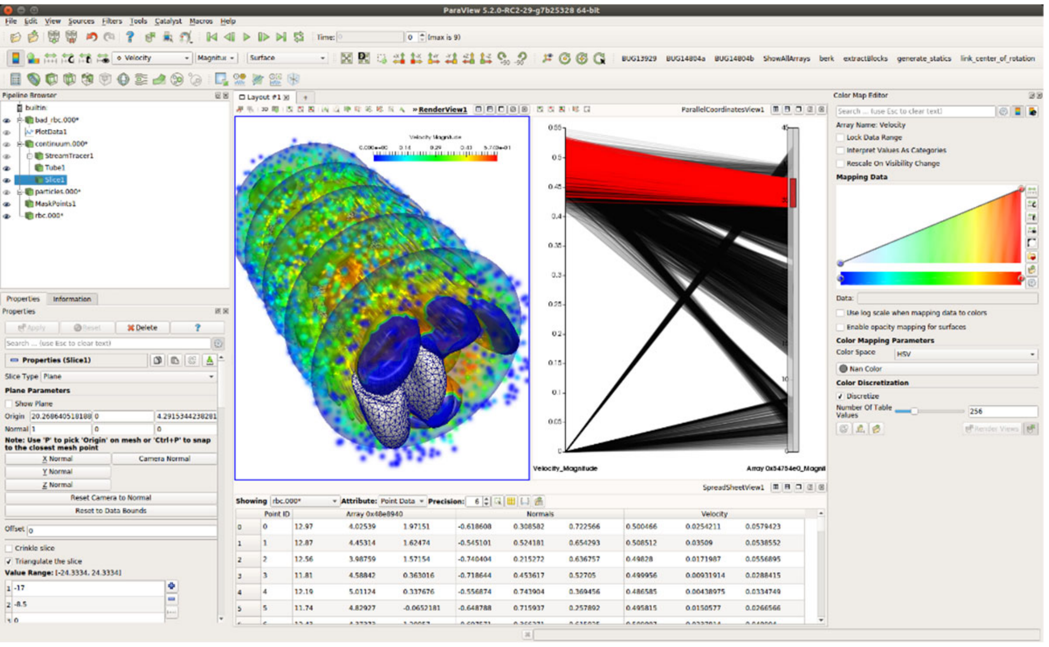
https://vtk.org/

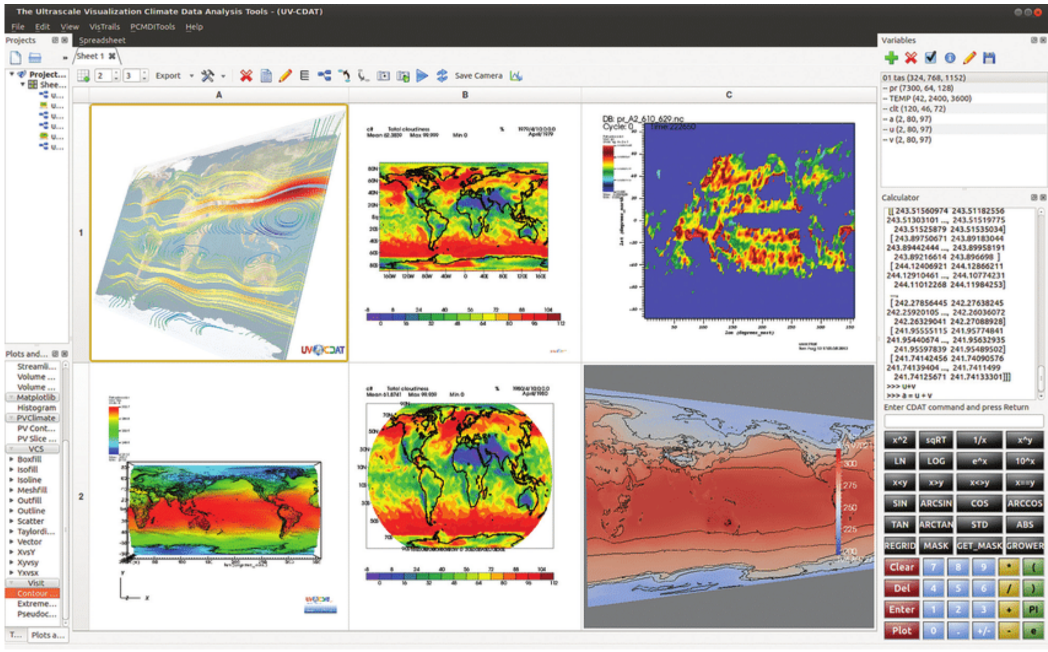
What is VTK (visualization toolkit)?

- An open source, freely available software system for 3D graphics, image processing, and visualization.
- Support for hundreds of algorithms in visualization and image processing.
- Was created by 3 GE researchers, now is maintained by Kitware, but actively developed and improved by researchers across the world.
- Has been applied widely in many real-world applications and research works.



https://vtk.org/gallery/

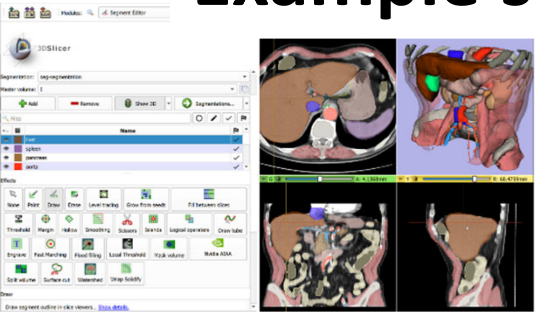




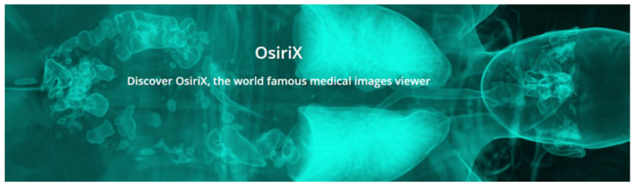
Paraview
https://www.paraview.org/

VisIt
https://visit-dav.github.io/visit-website/index.html

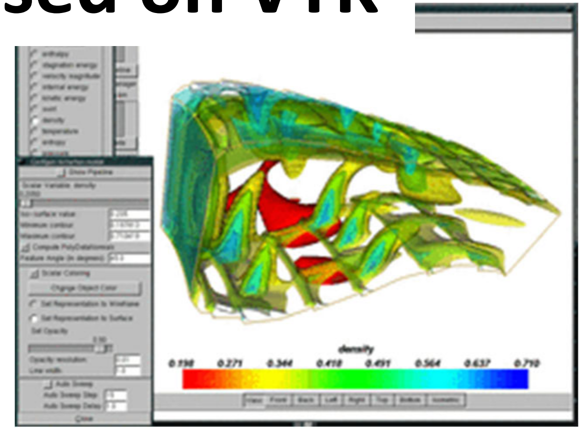
Example systems developed based on VTK



[3D Slicer]



[OSIRIX : 3D DICOM Viewer]



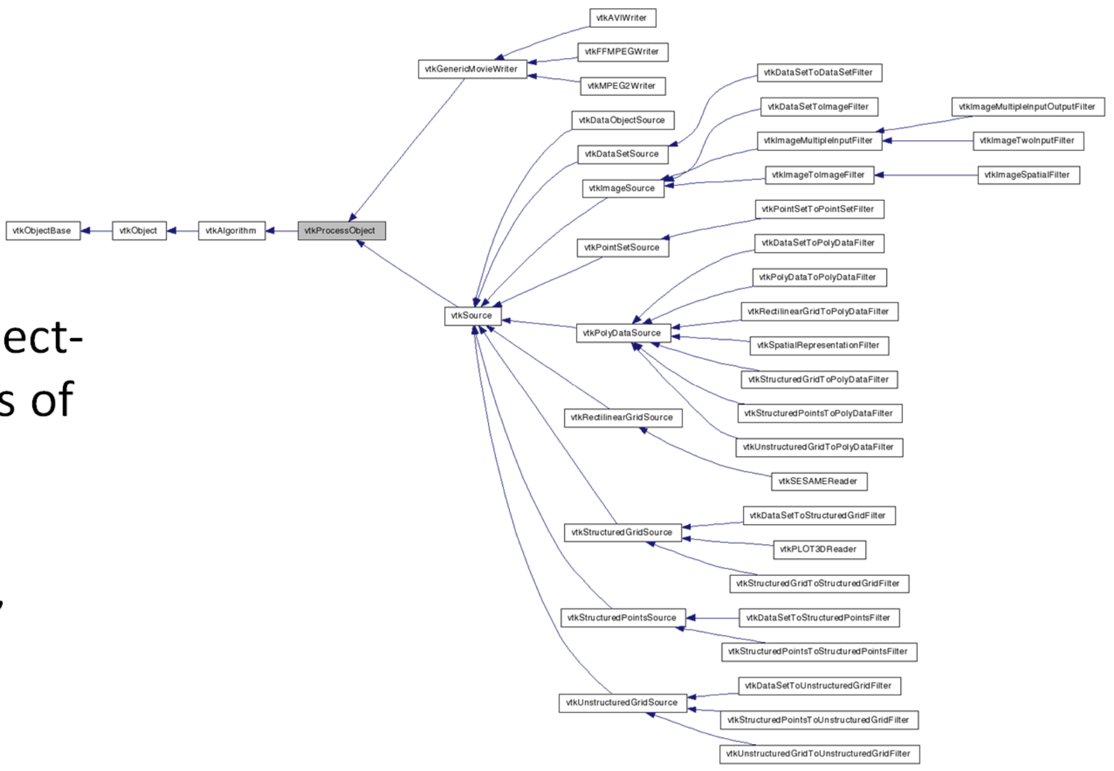
[MayaVi]

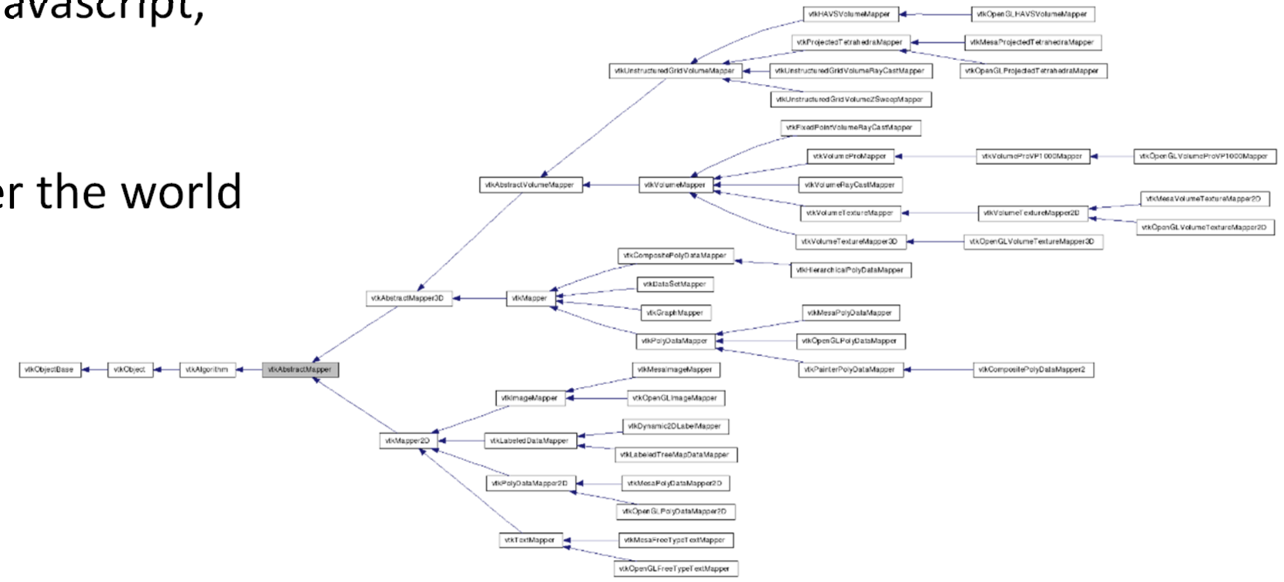
The core of VTK is written in C++ with object-oriented paradigm and contains hundreds of classes

VTK compiles and runs on Windows , MacOS , Linux

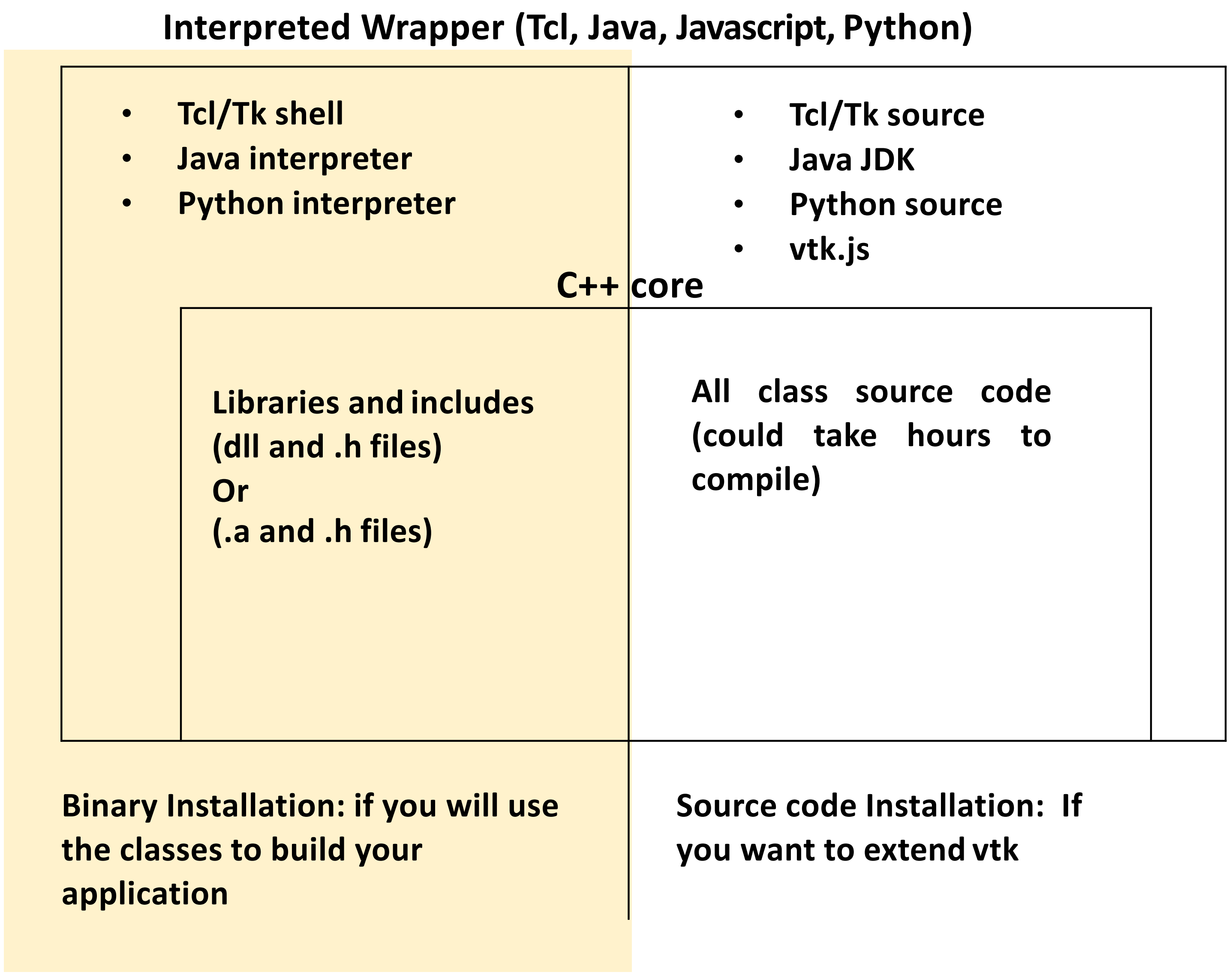
It provides different interfaces for a few languages, including Tcl/Tk, Java, Javascript, and Python

It has users and developers all over the world


vtkObjectBase
vtkObject
vtkAlgorithm
vtkProcessObject
vtkGenericMovieWriter
vtkAVIWriter
vtkFFMPEGWriter
vtkMPEG2Writer
vtkSource
vtkDataObjectSource
vtkDataSetSource
vtkImageSource
vtkPointSetSource
vtkPolyDataSource
vtkRectilinearGridSource
vtkStructuredGridSource
vtkStructuredPointsSource
vtkUnstructuredGridSource
vtkDataSetToDataSetFilter
vtkDataSetToImageFilter
vtkImageMultipleInputFilter
vtkImageToImageFilter
vtkImageMultipleInputOutputFilter
vtkImageTwoInputFilter
vtkImageSpatialFilter
vtkPointSetToPointSetFilter
vtkDataSetToPolyDataFilter
vtkPolyDataToPolyDataFilter
vtkRectilinearGridToPolyDataFilter
vtkSpatialRepresentationFilter
vtkStructuredGridToPolyDataFilter
vtkStructuredPointsToPolyDataFilter
vtkUnstructuredGridToPolyDataFilter
vtkSESAMEReader
vtkDataSetToStructuredGridFilter
vtkPLOT3DReader
vtkStructuredGridToStructuredGridFilter
vtkDataSetToStructuredPointsFilter
vtkStructuredPointsToStructuredPointsFilter
vtkDataSetToUnstructuredGridFilter
vtkStructuredPointsToUnstructuredGridFilter
vtkUnstructuredGridToUnstructuredGridFilter


vtkObjectBase
vtkObject
vtkAlgorithm
vtkAbstractMapper
vtkAbstractMapper3D
vtkAbstractVolumeMapper
vtkUnstructuredGridVolumeMapper
vtkHAVSVolumeMapper
vtkOpenGLHAVSVolumeMapper
vtkProjectedTetrahedraMapper
vtkMesaProjectedTetrahedraMapper
vtkOpenGLProjectedTetrahedraMapper
vtkUnstructuredGridVolumeRayCastMapper
vtkUnstructuredGridVolumeZSweepMapper
vtkVolumeMapper
vtkFixedPointVolumeRayCastMapper
vtkVolumeProMapper
vtkVolumeProVP1000Mapper
vtkOpenGLVolumeProVP1000Mapper
vtkVolumeRayCastMapper
vtkVolumeTextureMapper
vtkVolumeTextureMapper2D
vtkMesaVolumeTextureMapper2D
vtkOpenGLVolumeTextureMapper2D
vtkVolumeTextureMapper3D
vtkOpenGLVolumeTextureMapper3D
vtkMapper
vtkCompositePolyDataMapper
vtkHierarchicalPolyDataMapper
vtkDataSetMapper
vtkGraphMapper
vtkPolyDataMapper
vtkMesaPolyDataMapper
vtkOpenGLPolyDataMapper
vtkPainterPolyDataMapper
vtkCompositePolyDataMapper2
vtkMapper2D
vtkImageMapper
vtkMesaImageMapper
vtkOpenGLImageMapper
vtkLabeledDataMapper
vtkDynamic2DLabelMapper
vtkLabeledTreeMapDataMapper
vtkPolyDataMapper2D
vtkMesaPolyDataMapper2D
vtkOpenGLPolyDataMapper2D
vtkTextMapper
vtkMesaFreeTypeTextMapper
vtkOpenGLFreeTypeTextMapper

System Architecture



Interpreted Wrapper (Tcl, Java, Javascript, Python)

- **Tcl/Tk shell**
- **Java interpreter**
- **Python interpreter**

- **Tcl/Tk source**
- **Java JDK**
- **Python source**
- **vtk.js**

C++ core

Libraries and includes (dll and .h files) Or (.a and .h files)

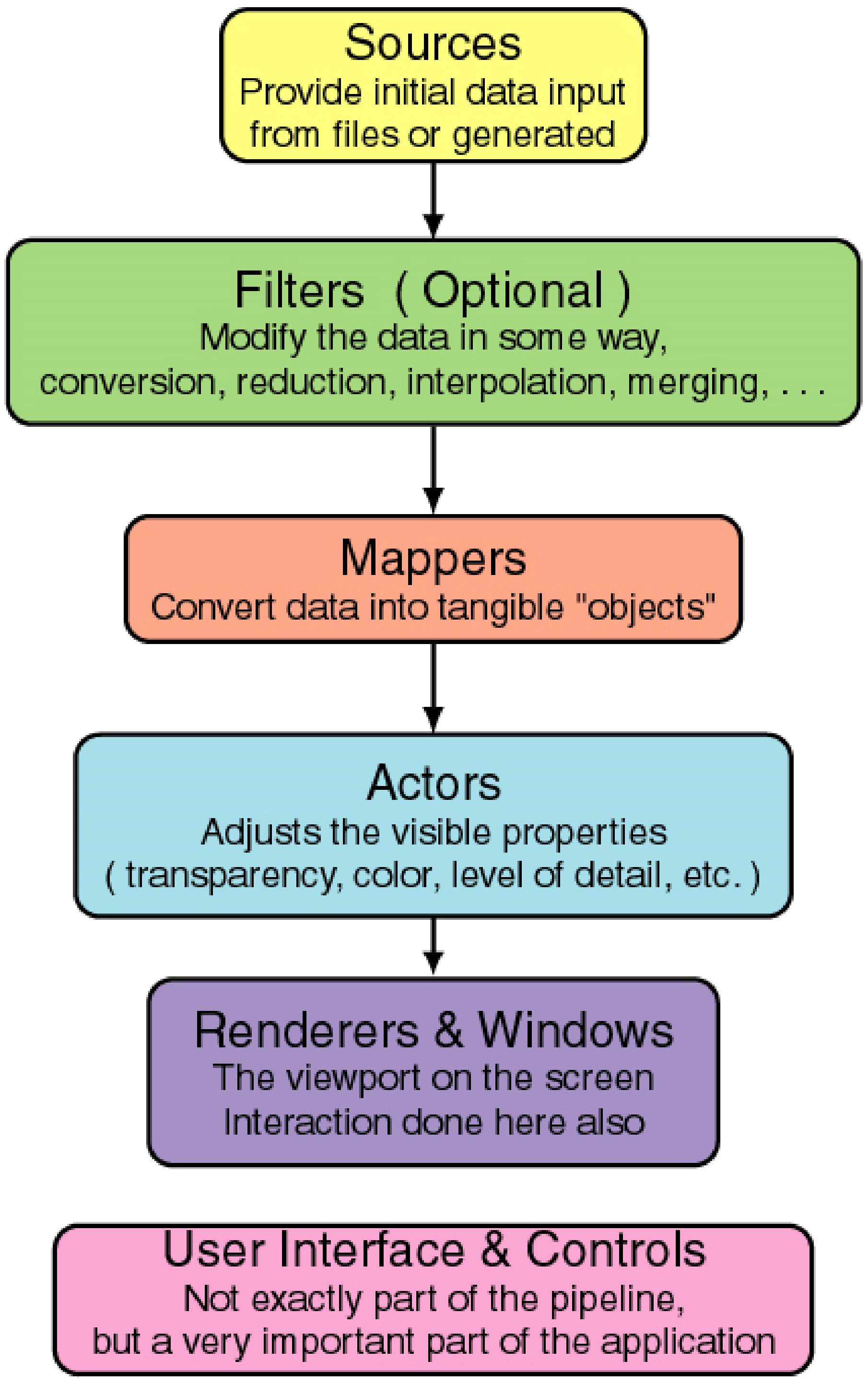
All class source code (could take hours to compile)

Binary Installation: if you will use the classes to build your application

Source code Installation: If you want to extend vtk

VTK Visualization Pipeline

In VTK, visualizations are created via a pipeline as shown to the right.


Sources
Provide initial data input from files or generated
Filters (Optional)
Modify the data in some way, conversion, reduction, interpolation, merging, . . .
Mappers
Convert data into tangible "objects"
Actors
Adjusts the visible properties (transparency, color, level of detail, etc.)
Renderers & Windows
The viewport on the screen
Interaction done here also
User Interface & Controls
Not exactly part of the pipeline, but a very important part of the application

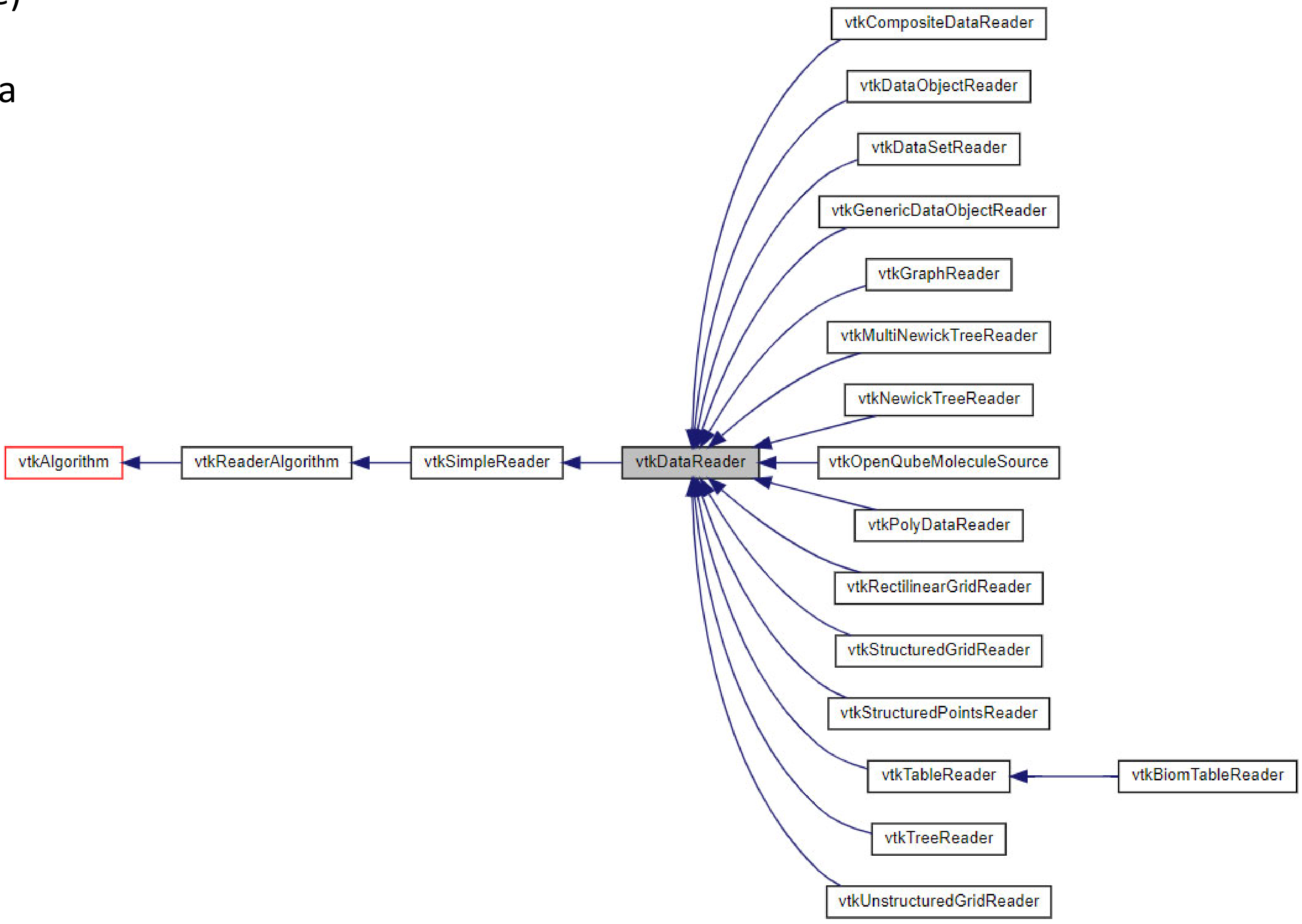
https://www.cs.uic.edu/~jbell/CS526/Tutorial/Tutorial.html

VTK pipeline

```
''' Step 1: Read a vtk data file '''
vtk_reader = vtk.vtkDataSetReader()
vtk_reader.SetFileName(input_file_name)
```

Source

The **source** imports (from file) or creates (e.g., function) the data

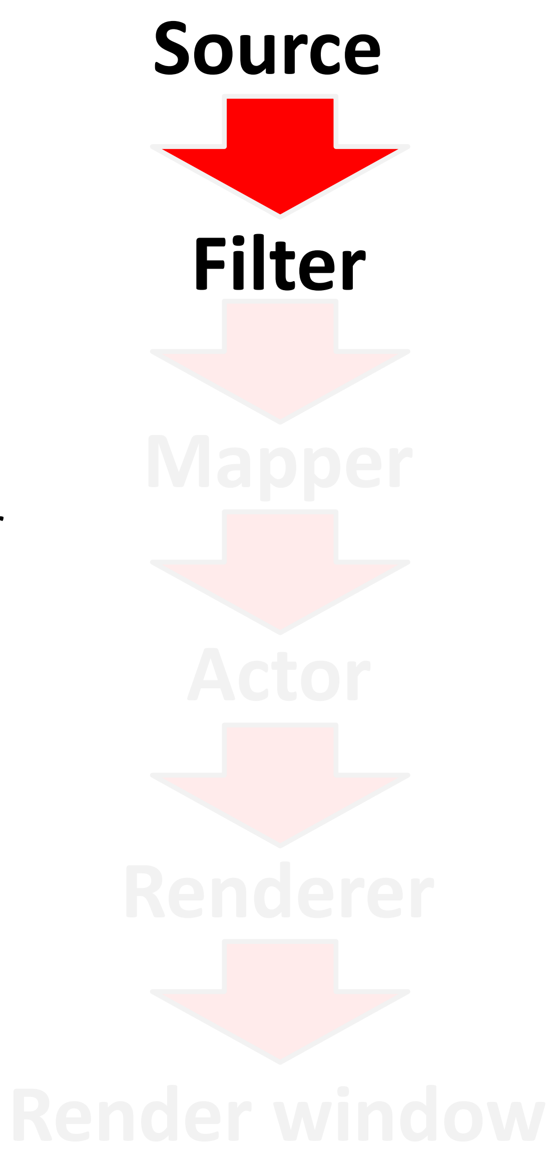


VTK pipeline

```
''' Step 1: Read a vtk data file '''
vtk_reader = vtk.vtkDataSetReader()
vtk_reader.SetFileName(input_file_name)

'''Step 2: Get geometry using a filter '''
vtk_geometry = vtk.vtkExtractEdges()
```

One or more **filters** process the data (from source) to create geometric objects (lines or surfaces)

Extract the edges from the loaded grid. This can be any filter, like the ***contour*** *filter that you will need later*



VTK pipeline

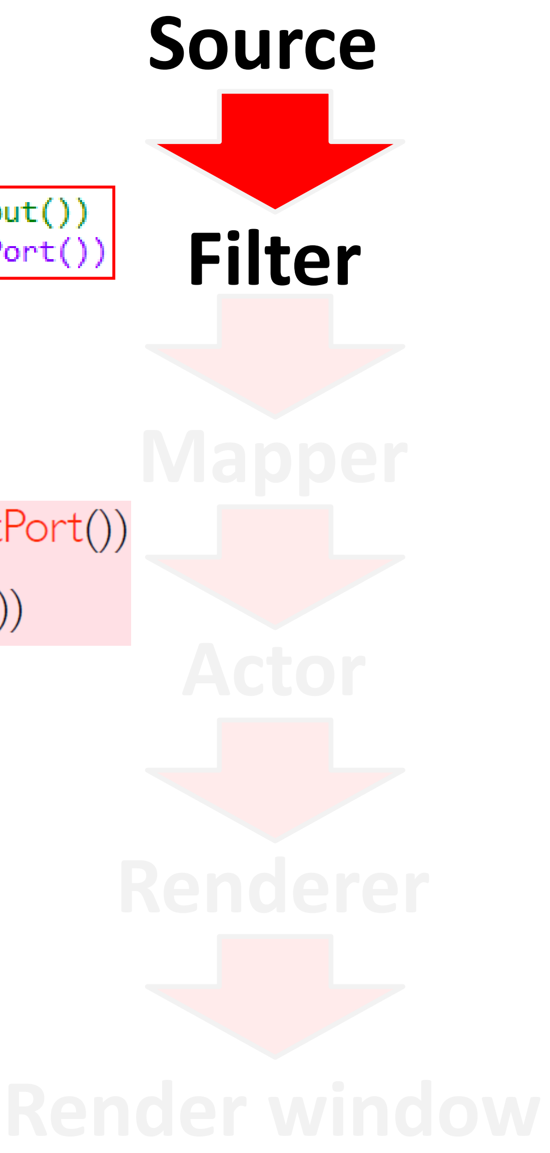
```
''' Step 1: Read a vtk data file '''
vtk_reader = vtk.vtkDataSetReader()
vtk_reader.SetFileName(input_file_name)

'''Step 2: Get geometry using a filter '''
vtk_geometry = vtk.vtkExtractEdges()
#vtk_geometry.SetInputData(vtk_reader.GetPolyDataOutput())
vtk_geometry.SetInputConnection(vtk_reader.GetOutputPort())
```

One or more **filters** process the data (from source) to create geometric objects (lines or surfaces)

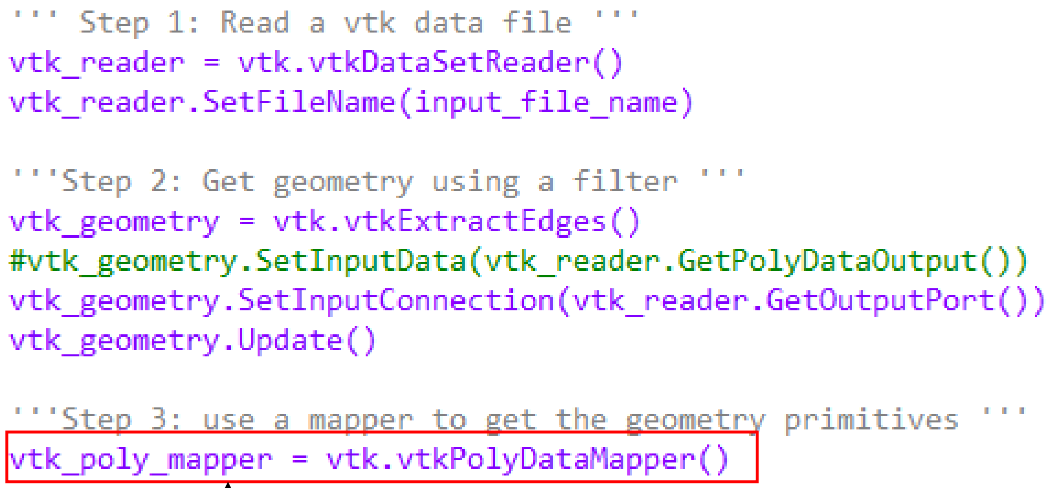
VTK pipeline connection

(1) *Receiver*.SetInputConnection(*Supplier*.GetOutputPort())

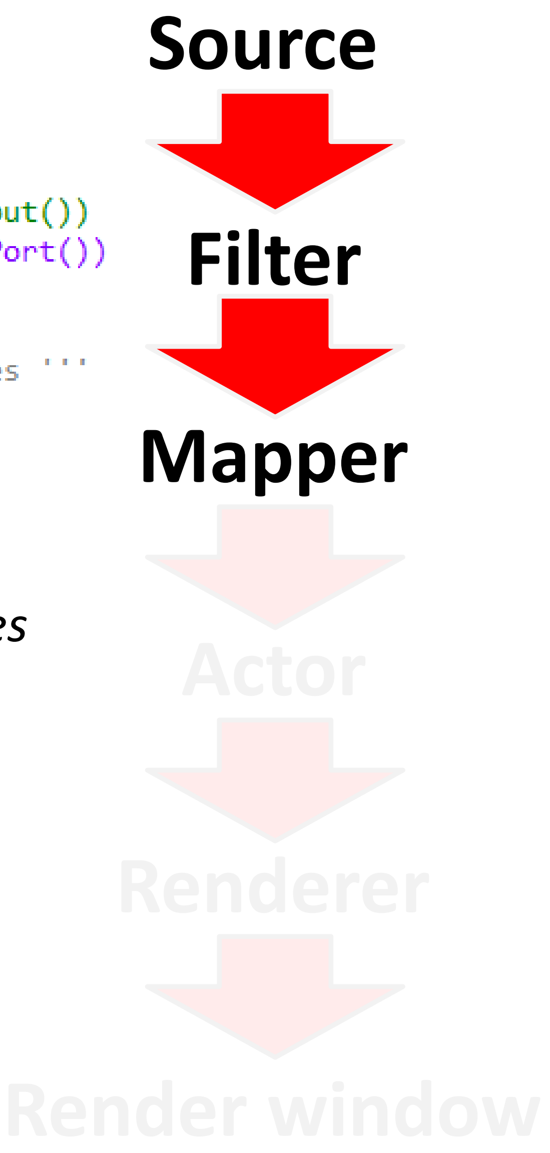
(2) *Receiver*.SetInputData(*Supplier*.GetOutput())



VTK pipeline



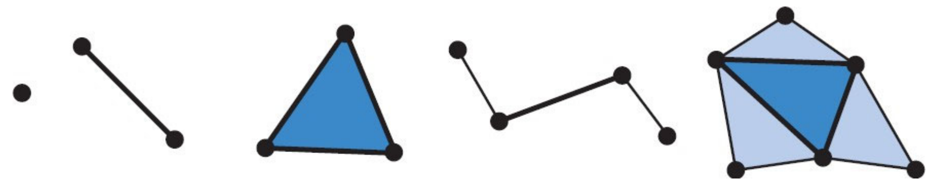
```
''' Step 1: Read a vtk data file '''
vtk_reader = vtk.vtkDataSetReader()
vtk_reader.SetFileName(input_file_name)

'''Step 2: Get geometry using a filter '''
vtk_geometry = vtk.vtkExtractEdges()
#vtk_geometry.SetInputData(vtk_reader.GetPolyDataOutput())
vtk_geometry.SetInputConnection(vtk_reader.GetOutputPort())
vtk_geometry.Update()

'''Step 3: use a mapper to get the geometry primitives '''
vtk_poly_mapper = vtk.vtkPolyDataMapper()
```



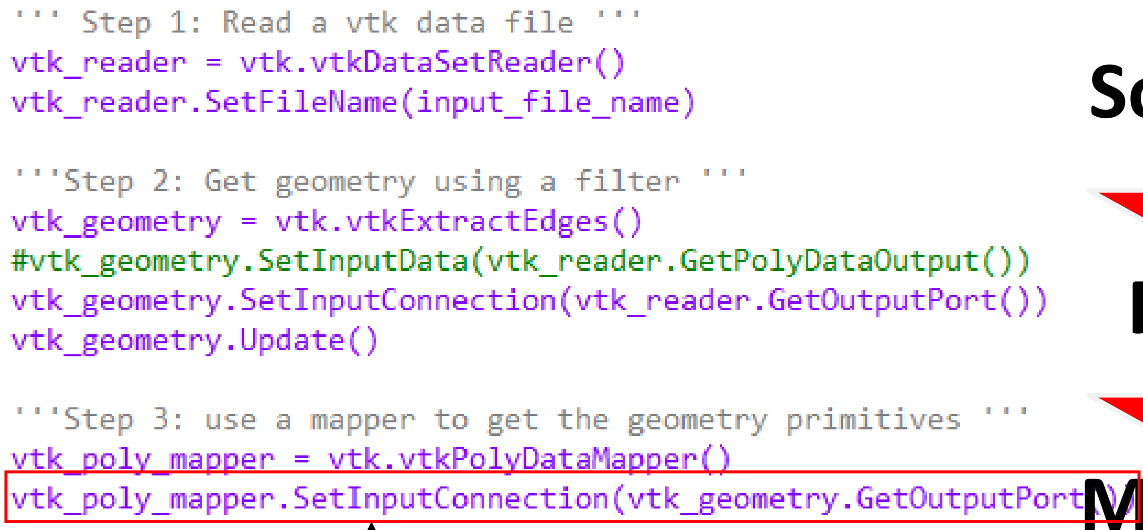
The **mapper** converts geometry to graphical primitives (points, line segments, triangles,...)

Create a graphical mapper

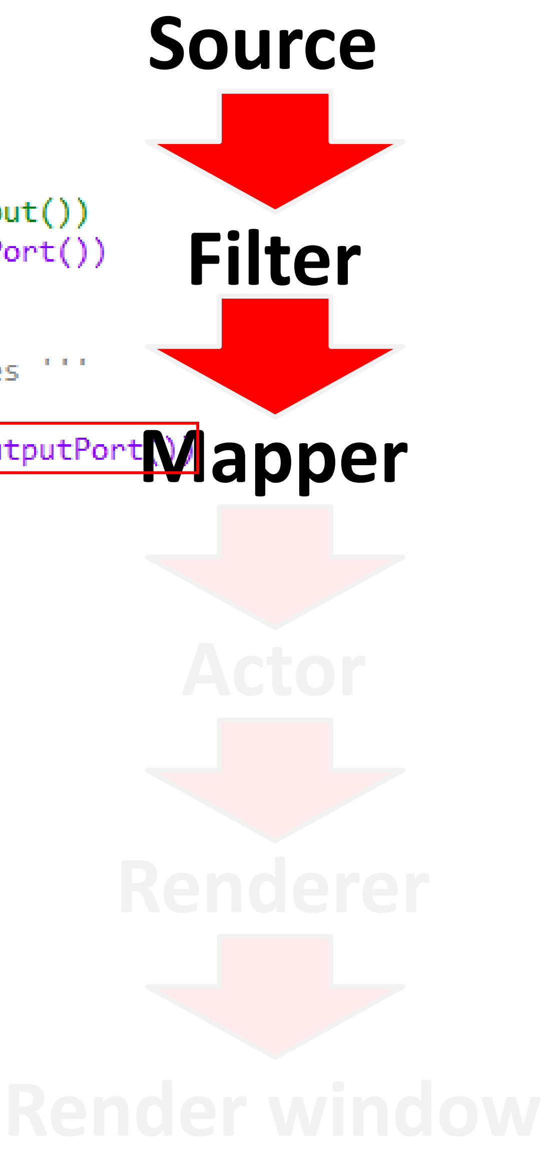
Similar to filter, there are different types of mapper



VTK pipeline



```
''' Step 1: Read a vtk data file '''
vtk_reader = vtk.vtkDataSetReader()
vtk_reader.SetFileName(input_file_name)

'''Step 2: Get geometry using a filter '''
vtk_geometry = vtk.vtkExtractEdges()
#vtk_geometry.SetInputData(vtk_reader.GetPolyDataOutput())
vtk_geometry.SetInputConnection(vtk_reader.GetOutputPort())
vtk_geometry.Update()

'''Step 3: use a mapper to get the geometry primitives '''
vtk_poly_mapper = vtk.vtkPolyDataMapper()
vtk_poly_mapper.SetInputConnection(vtk_geometry.GetOutputPort())
```



The **mapper** converts geometry to graphical primitives (points, line segments, triangles,...)

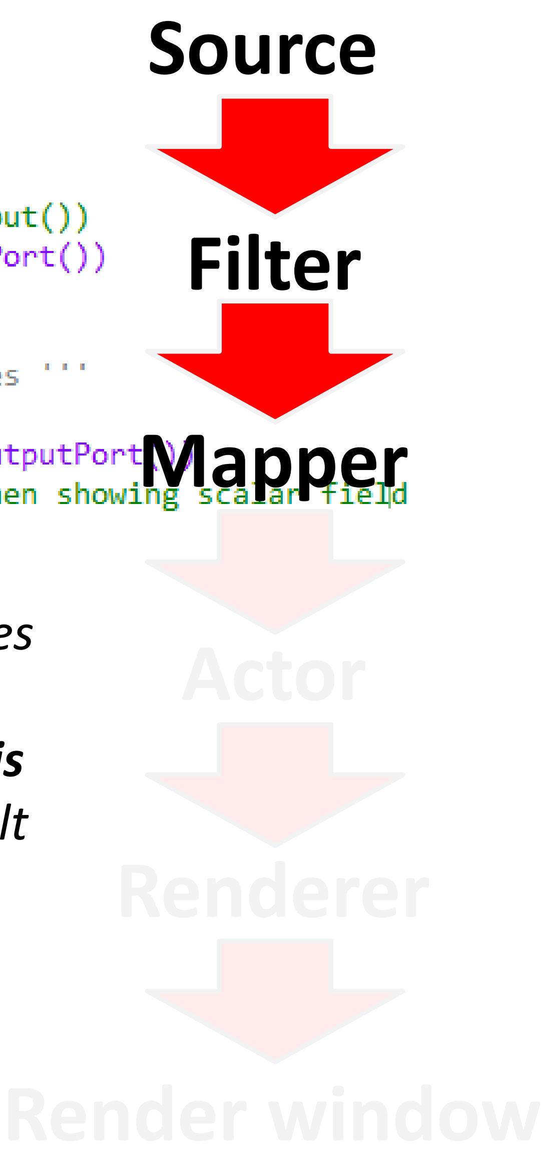
Apply it to the geometry created by the above filter

VTK pipeline

```
''' Step 1: Read a vtk data file '''
vtk_reader = vtk.vtkDataSetReader()
vtk_reader.SetFileName(input_file_name)

'''Step 2: Get geometry using a filter '''
vtk_geometry = vtk.vtkExtractEdges()
#vtk_geometry.SetInputData(vtk_reader.GetPolyDataOutput())
vtk_geometry.SetInputConnection(vtk_reader.GetOutputPort())
vtk_geometry.Update()

'''Step 3: use a mapper to get the geometry primitives '''
vtk_poly_mapper = vtk.vtkPolyDataMapper()
vtk_poly_mapper.SetInputConnection(vtk_geometry.GetOutputPort())
vtk_poly_mapper.ScalarVisibilityOff()#Turn this on when showing scalar field
```

The **mapper** converts geometry to graphical primitives (points, line segments, triangles,...)

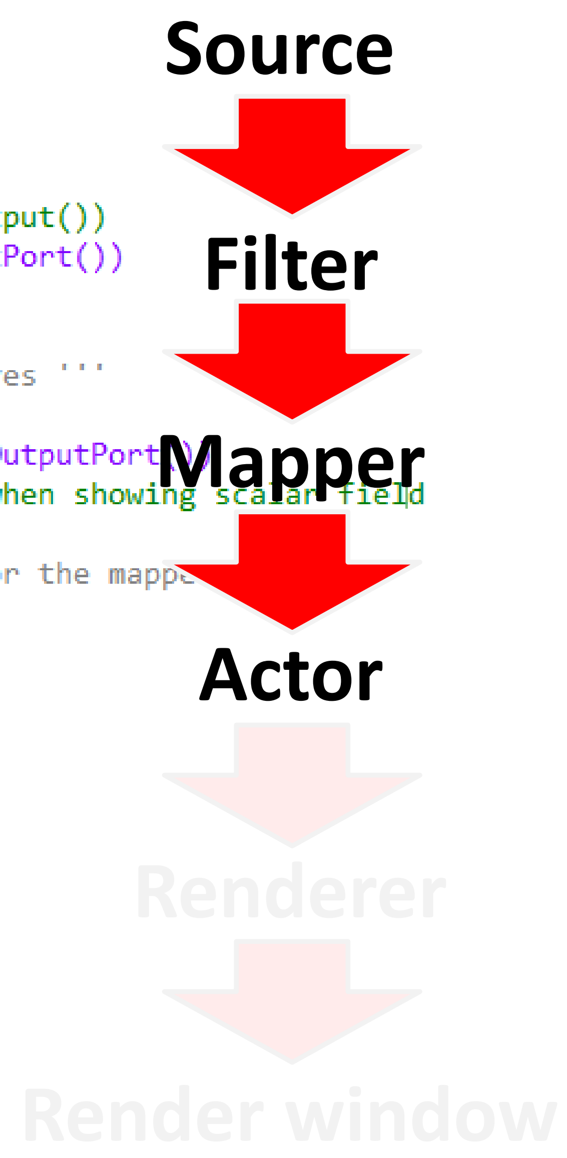
This turns OFF the use of the scalar values for color coding
In the later scalar field visualization, this needs to be ON*. Good news is the default of this setting is ON!*



VTK pipeline

```
''' Step 1: Read a vtk data file '''
vtk_reader = vtk.vtkDataSetReader()
vtk_reader.SetFileName(input_file_name)

'''Step 2: Get geometry using a filter '''
vtk_geometry = vtk.vtkExtractEdges()
#vtk_geometry.SetInputData(vtk_reader.GetPolyDataOutput())
vtk_geometry.SetInputConnection(vtk_reader.GetOutputPort())
vtk_geometry.Update()

'''Step 3: use a mapper to get the geometry primitives '''
vtk_poly_mapper = vtk.vtkPolyDataMapper()
vtk_poly_mapper.SetInputConnection(vtk_geometry.GetOutputPort())
vtk_poly_mapper.ScalarVisibilityOff()#Turn this on when showing scalar field

'''Step 4: create an actor and set the appearance for the mappe
vtk_actor = vtk.vtkActor()
```

Create an actor



The **actor** positions the primitives in the scene (e.g., transformation) and controls their appearance (colors, transparency, texture, ...)

vtkActor represents an object (geometry and properties) in a rendering scene

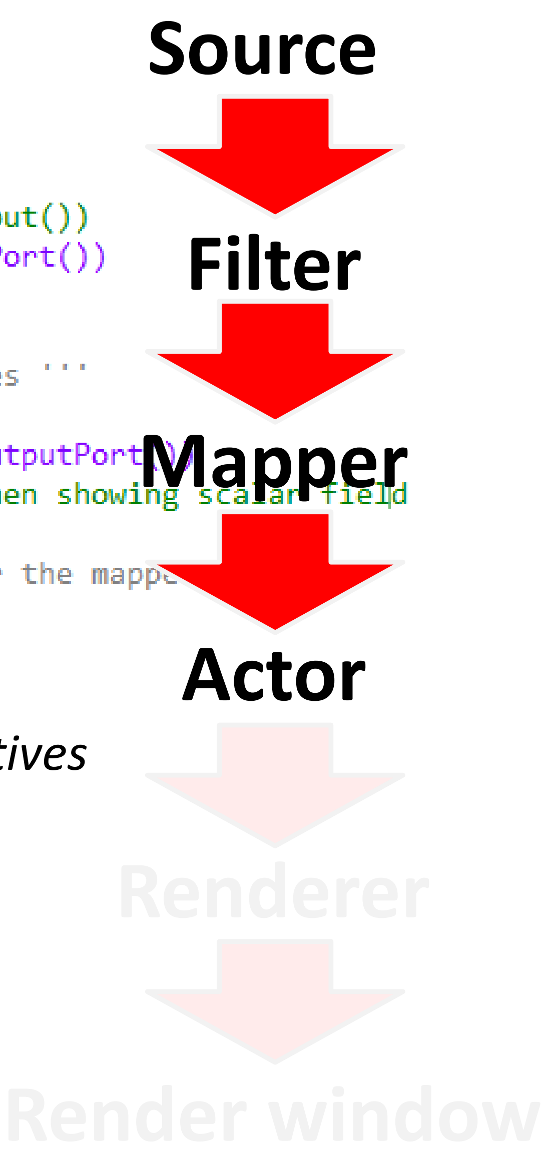
Has position, scale, orientation, various rendering properties, textures, etc. Keeps a reference to the mapper

VTK pipeline

```
''' Step 1: Read a vtk data file '''
vtk_reader = vtk.vtkDataSetReader()
vtk_reader.SetFileName(input_file_name)

'''Step 2: Get geometry using a filter '''
vtk_geometry = vtk.vtkExtractEdges()
#vtk_geometry.SetInputData(vtk_reader.GetPolyDataOutput())
vtk_geometry.SetInputConnection(vtk_reader.GetOutputPort())
vtk_geometry.Update()

'''Step 3: use a mapper to get the geometry primitives '''
vtk_poly_mapper = vtk.vtkPolyDataMapper()
vtk_poly_mapper.SetInputConnection(vtk_geometry.GetOutputPort
vtk_poly_mapper.ScalarVisibilityOff()#Turn this on when showing scalar field

'''Step 4: create an actor and set the appearance for the mappe
vtk_actor = vtk.vtkActor()
vtk_actor.SetMapper(vtk_poly_mapper)
```

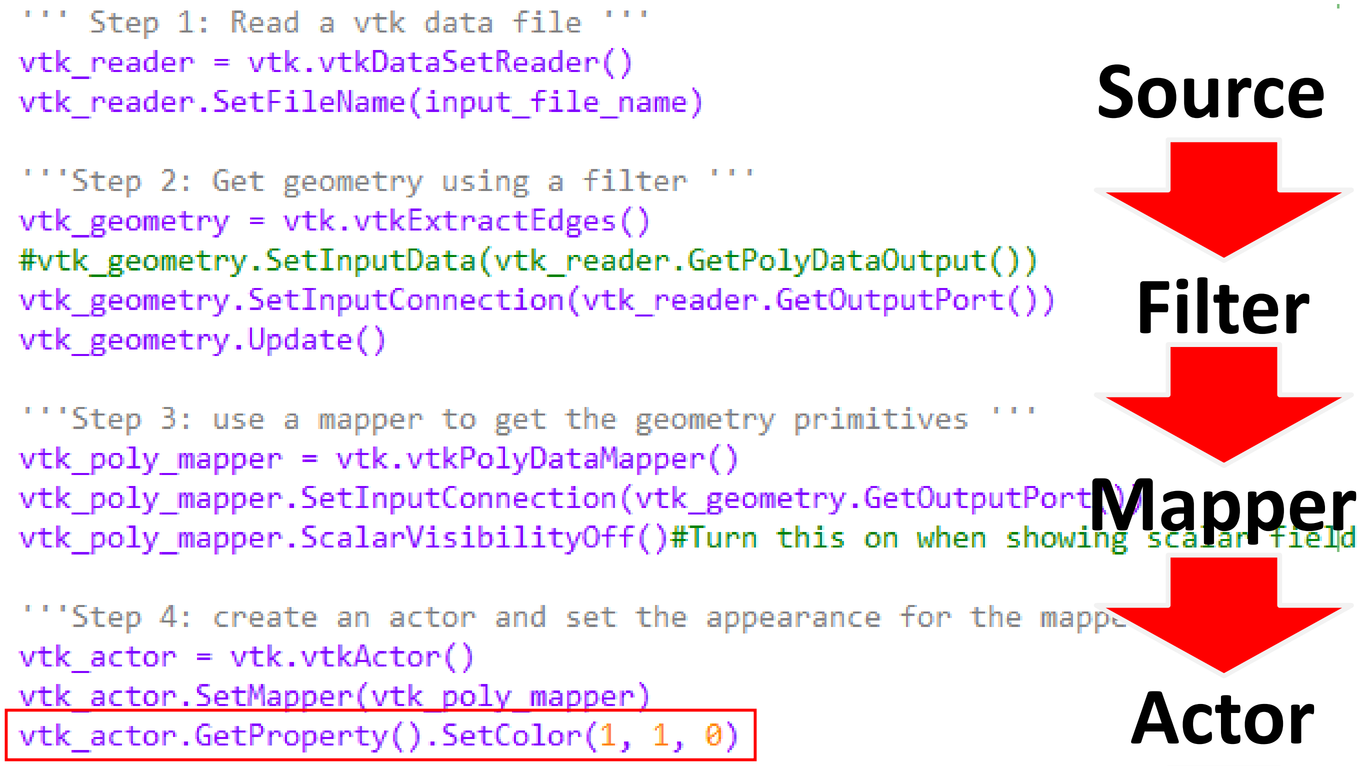
Attach it to the above graphical primitives

The **actor** positions the primitives in the scene (e.g., transformation) and controls their appearance (colors, transparency, texture, ...)



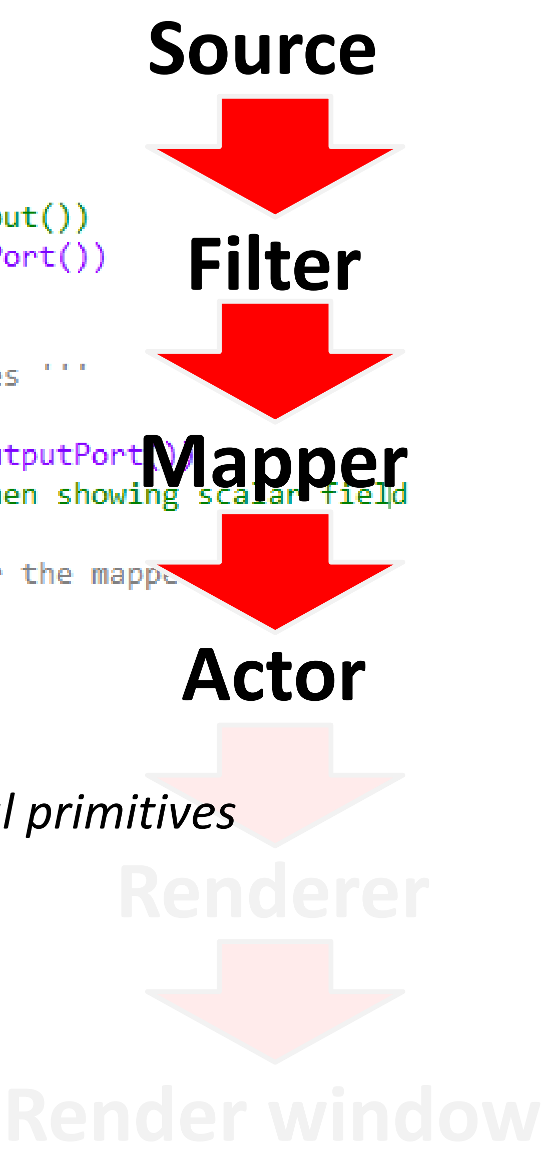
VTK pipeline

The **actor** positions the primitives in the scene (e.g., transformation) and controls their appearance (colors, transparency, texture, …)



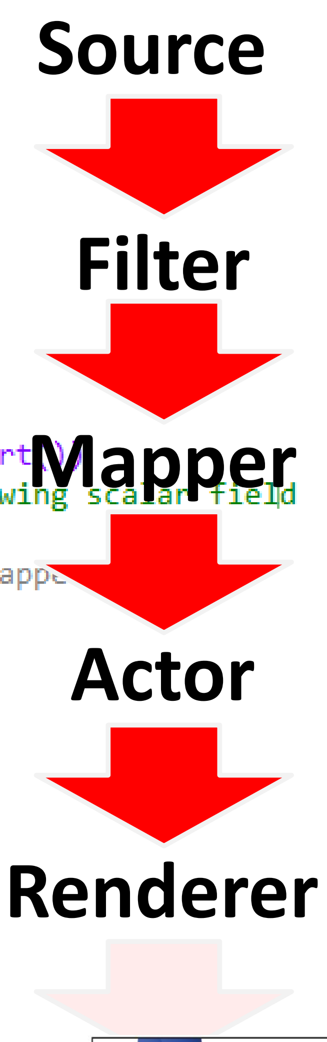
```
''' Step 1: Read a vtk data file '''
vtk_reader = vtk.vtkDataSetReader()
vtk_reader.SetFileName(input_file_name)

'''Step 2: Get geometry using a filter '''
vtk_geometry = vtk.vtkExtractEdges()
#vtk_geometry.SetInputData(vtk_reader.GetPolyDataOutput())
vtk_geometry.SetInputConnection(vtk_reader.GetOutputPort())
vtk_geometry.Update()

'''Step 3: use a mapper to get the geometry primitives '''
vtk_poly_mapper = vtk.vtkPolyDataMapper()
vtk_poly_mapper.SetInputConnection(vtk_geometry.GetOutputPort
vtk_poly_mapper.ScalarVisibilityOff()#Turn this on when showing scalar field

'''Step 4: create an actor and set the appearance for the mappe
vtk_actor = vtk.vtkActor()
vtk_actor.SetMapper(vtk_poly_mapper)
vtk_actor.GetProperty().SetColor(1, 1, 0)
```

Set a constant color for these graphical primitives



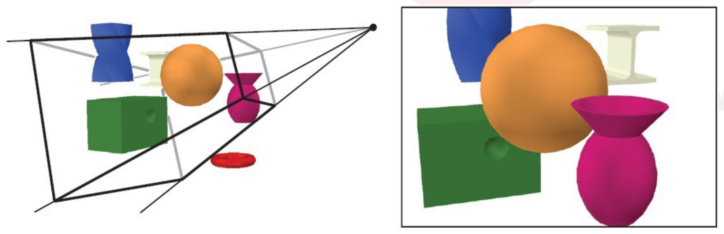
VTK pipeline

```
''' Step 1: Read a vtk data file '''
vtk_reader = vtk.vtkDataSetReader()
vtk_reader.SetFileName(input_file_name)

'''Step 2: Get geometry using a filter '''
vtk_geometry = vtk.vtkExtractEdges()
#vtk_geometry.SetInputData(vtk_reader.GetPolyDataOutput())
vtk_geometry.SetInputConnection(vtk_reader.GetOutputPort())
vtk_geometry.Update()

'''Step 3: use a mapper to get the geometry primitives '''
vtk_poly_mapper = vtk.vtkPolyDataMapper()
vtk_poly_mapper.SetInputConnection(vtk_geometry.GetOutputPort())
vtk_poly_mapper.ScalarVisibilityOff()#Turn this on when showing scalar field

'''Step 4: create an actor and set the appearance for the mappe
vtk_actor = vtk.vtkActor()
vtk_actor.SetMapper(vtk_poly_mapper)
vtk_actor.GetProperty().SetColor(1, 1, 0)

'''Step 5: create a render to set camera, lighting '''
render = vtk.vtkRenderer()
```



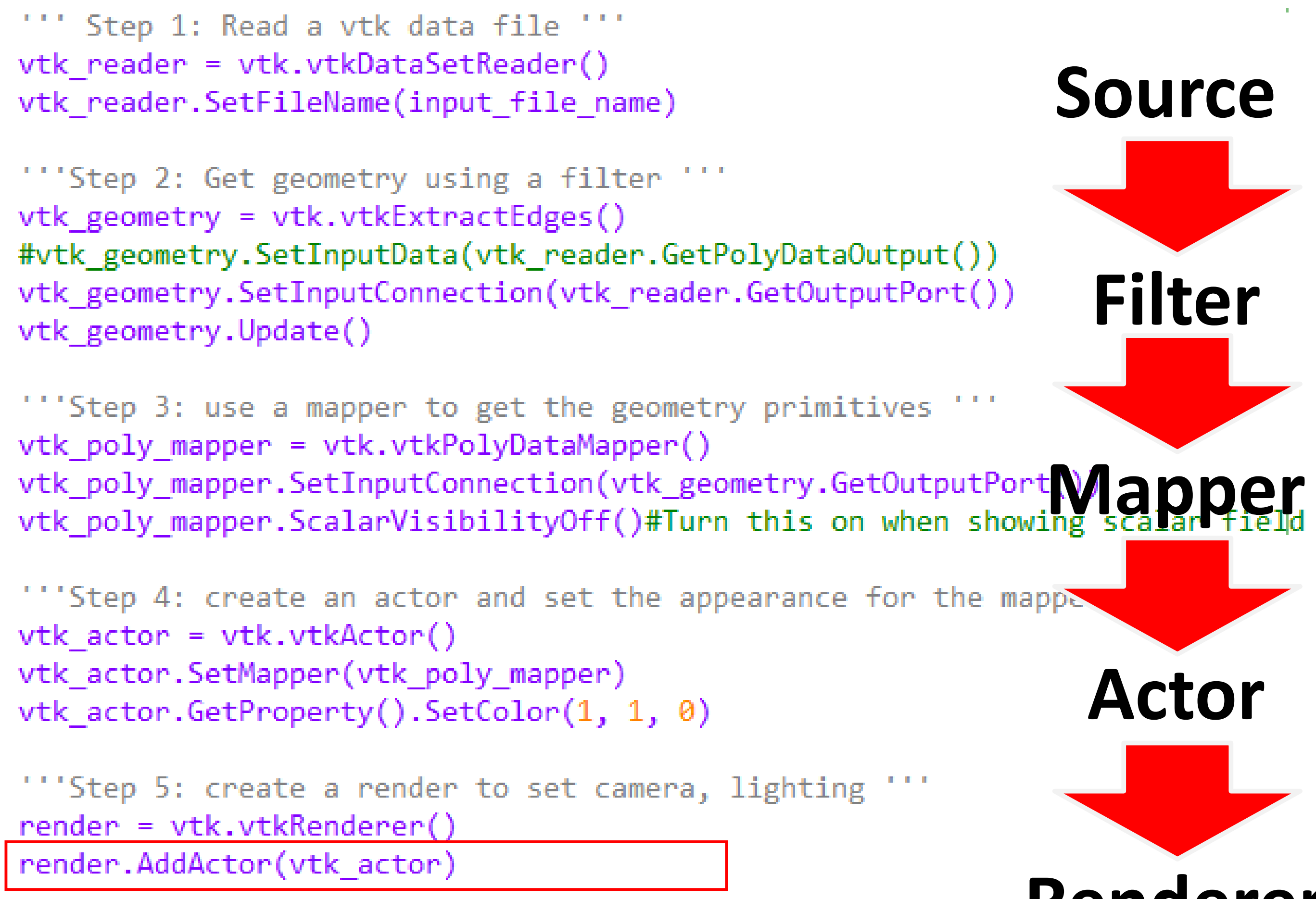
The **render** controls the camera and lighting

Create the scene render and set the camera and lighting. Here we will use the default setting.

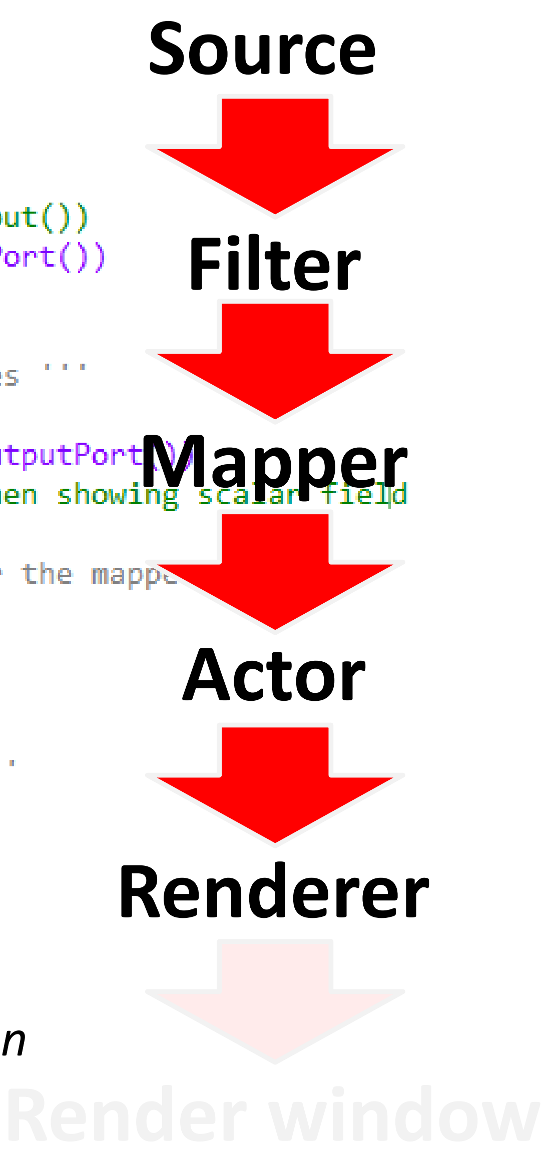


The process of converting 3D graphics primitives (points, lines, triangles, etc), a specification for lights and materials, and a camera view into an 2D image that can be displayed on the screen

VTK pipeline



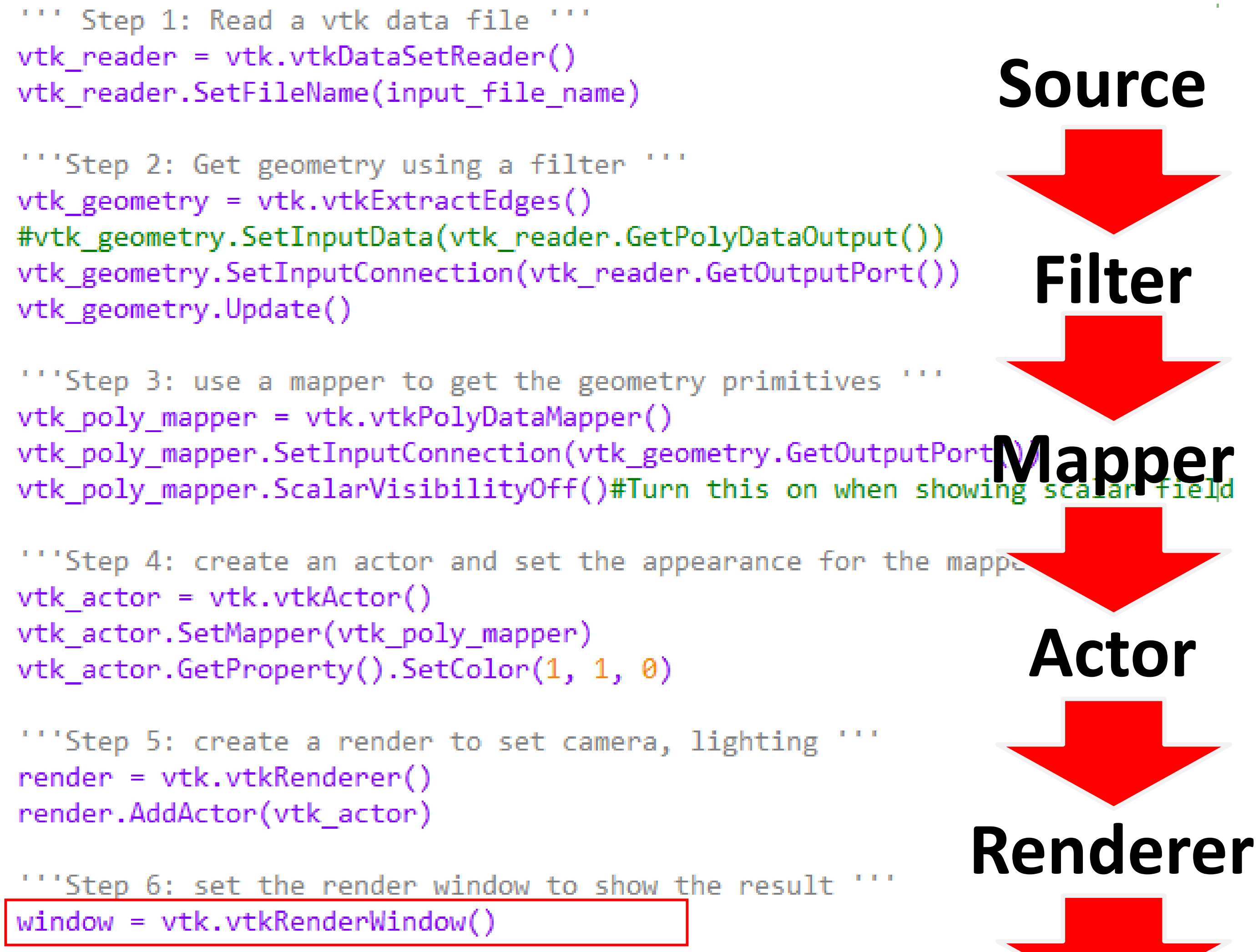
```
''' Step 1: Read a vtk data file '''
vtk_reader = vtk.vtkDataSetReader()
vtk_reader.SetFileName(input_file_name)

'''Step 2: Get geometry using a filter '''
vtk_geometry = vtk.vtkExtractEdges()
#vtk_geometry.SetInputData(vtk_reader.GetPolyDataOutput())
vtk_geometry.SetInputConnection(vtk_reader.GetOutputPort())
vtk_geometry.Update()

'''Step 3: use a mapper to get the geometry primitives '''
vtk_poly_mapper = vtk.vtkPolyDataMapper()
vtk_poly_mapper.SetInputConnection(vtk_geometry.GetOutputPort())
vtk_poly_mapper.ScalarVisibilityOff()#Turn this on when showing scalar field

'''Step 4: create an actor and set the appearance for the mappe
vtk_actor = vtk.vtkActor()
vtk_actor.SetMapper(vtk_poly_mapper)
vtk_actor.GetProperty().SetColor(1, 1, 0)

'''Step 5: create a render to set camera, lighting '''
render = vtk.vtkRenderer()
render.AddActor(vtk_actor)
```



The **render** controls the camera and lighting

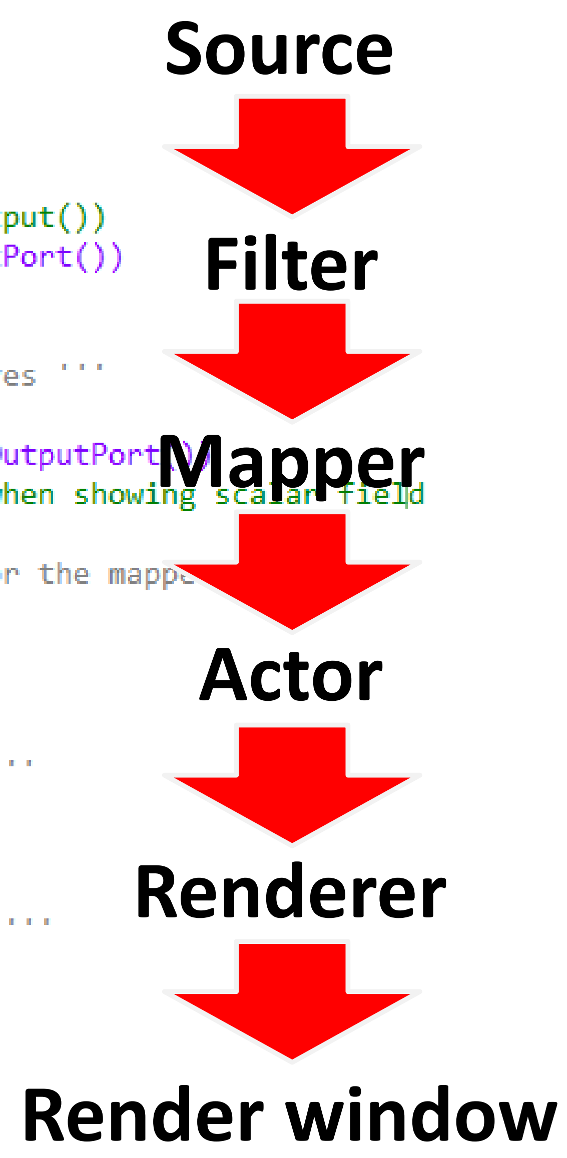
Add the above graphical objects into the scene. Multiple sets of graphical objects can be added.

VTK pipeline



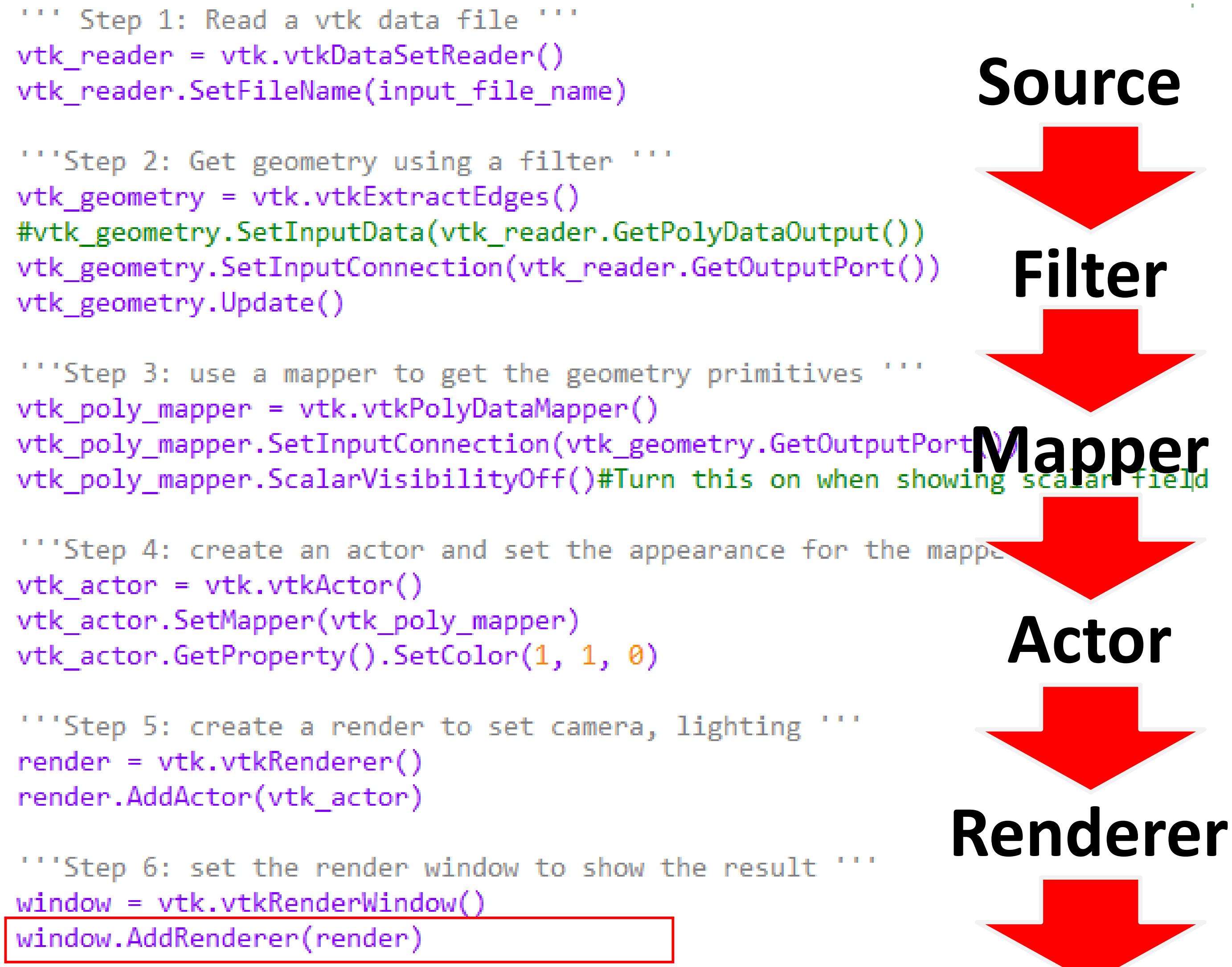
```
''' Step 1: Read a vtk data file '''
vtk_reader = vtk.vtkDataSetReader()
vtk_reader.SetFileName(input_file_name)

'''Step 2: Get geometry using a filter '''
vtk_geometry = vtk.vtkExtractEdges()
#vtk_geometry.SetInputData(vtk_reader.GetPolyDataOutput())
vtk_geometry.SetInputConnection(vtk_reader.GetOutputPort())
vtk_geometry.Update()

'''Step 3: use a mapper to get the geometry primitives '''
vtk_poly_mapper = vtk.vtkPolyDataMapper()
vtk_poly_mapper.SetInputConnection(vtk_geometry.GetOutputPort())
vtk_poly_mapper.ScalarVisibilityOff()#Turn this on when showing scalar field

'''Step 4: create an actor and set the appearance for the mappe
vtk_actor = vtk.vtkActor()
vtk_actor.SetMapper(vtk_poly_mapper)
vtk_actor.GetProperty().SetColor(1, 1, 0)

'''Step 5: create a render to set camera, lighting '''
render = vtk.vtkRenderer()
render.AddActor(vtk_actor)

'''Step 6: set the render window to show the result '''
window = vtk.vtkRenderWindow()
```

Create a window on the screen.



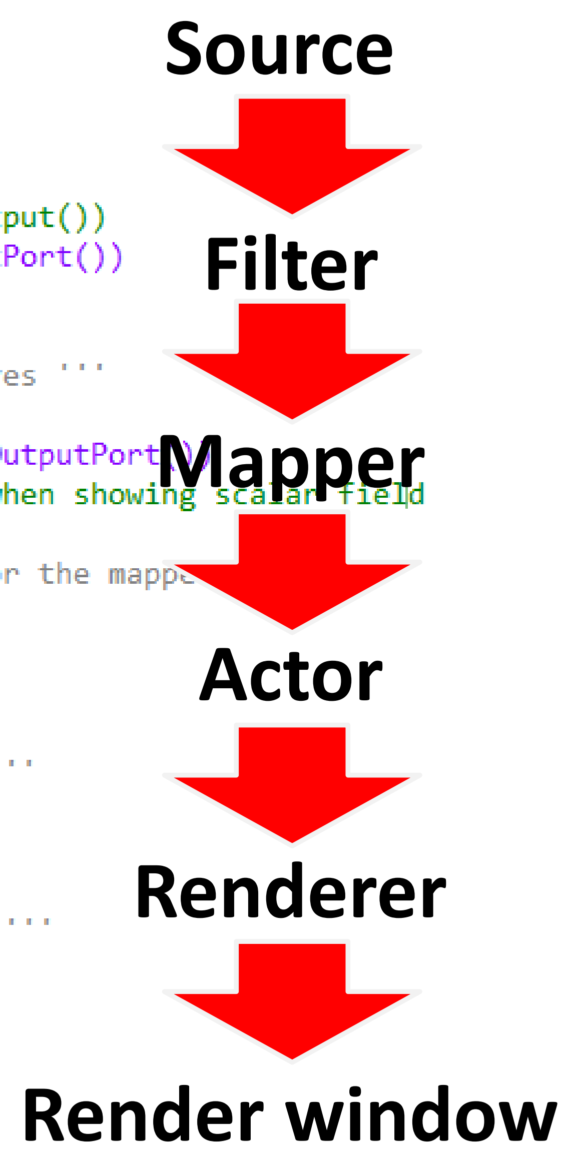
The **render window** displays the result on the screen and sets the resolution

VTK pipeline



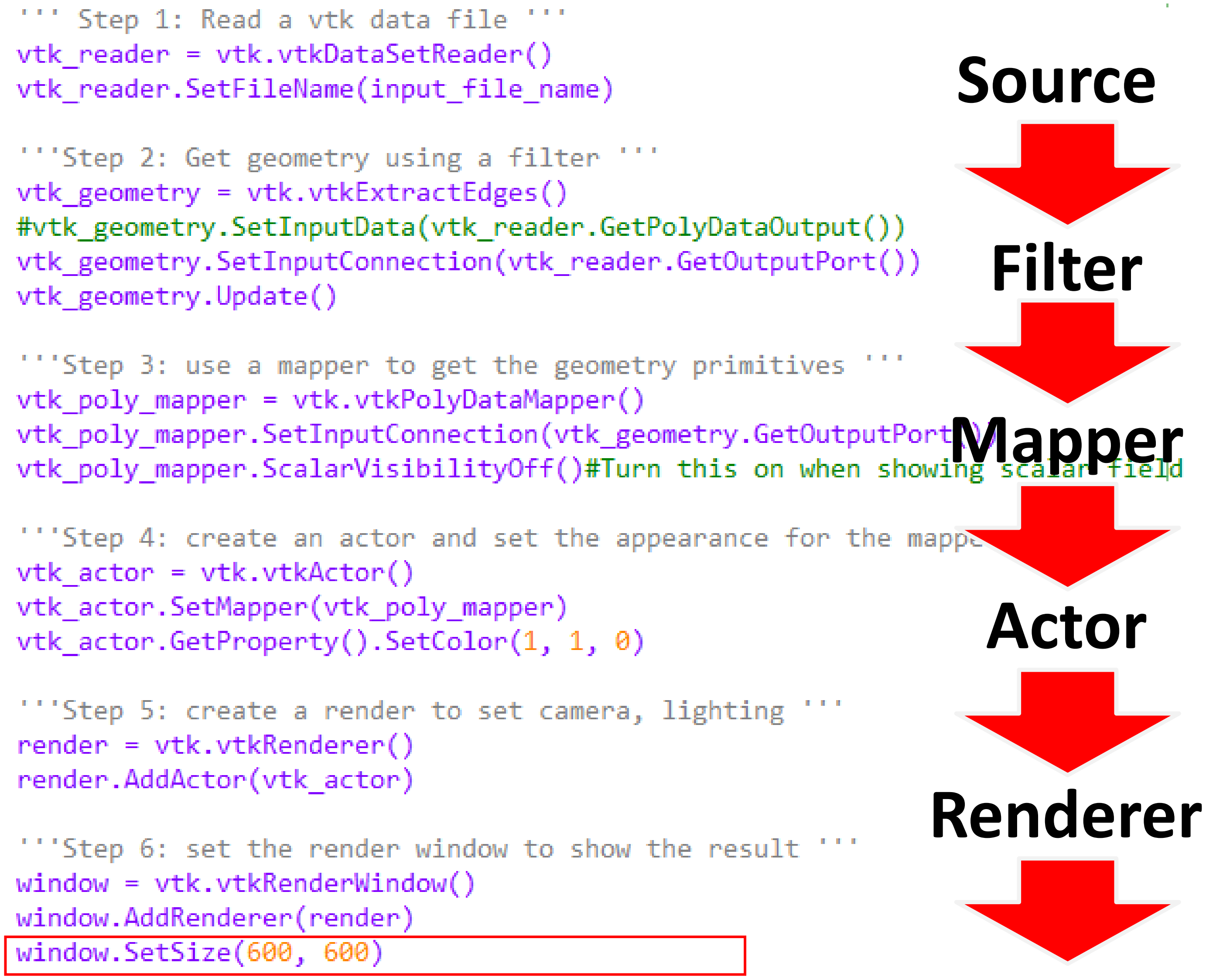
```
''' Step 1: Read a vtk data file '''
vtk_reader = vtk.vtkDataSetReader()
vtk_reader.SetFileName(input_file_name)

'''Step 2: Get geometry using a filter '''
vtk_geometry = vtk.vtkExtractEdges()
#vtk_geometry.SetInputData(vtk_reader.GetPolyDataOutput())
vtk_geometry.SetInputConnection(vtk_reader.GetOutputPort())
vtk_geometry.Update()

'''Step 3: use a mapper to get the geometry primitives '''
vtk_poly_mapper = vtk.vtkPolyDataMapper()
vtk_poly_mapper.SetInputConnection(vtk_geometry.GetOutputPort())
vtk_poly_mapper.ScalarVisibilityOff()#Turn this on when showing scalar field

'''Step 4: create an actor and set the appearance for the mappe
vtk_actor = vtk.vtkActor()
vtk_actor.SetMapper(vtk_poly_mapper)
vtk_actor.GetProperty().SetColor(1, 1, 0)

'''Step 5: create a render to set camera, lighting '''
render = vtk.vtkRenderer()
render.AddActor(vtk_actor)

'''Step 6: set the render window to show the result '''
window = vtk.vtkRenderWindow()
window.AddRenderer(render)
```

Render our scene in that window



The **render window** displays the result on the screen and sets the resolution

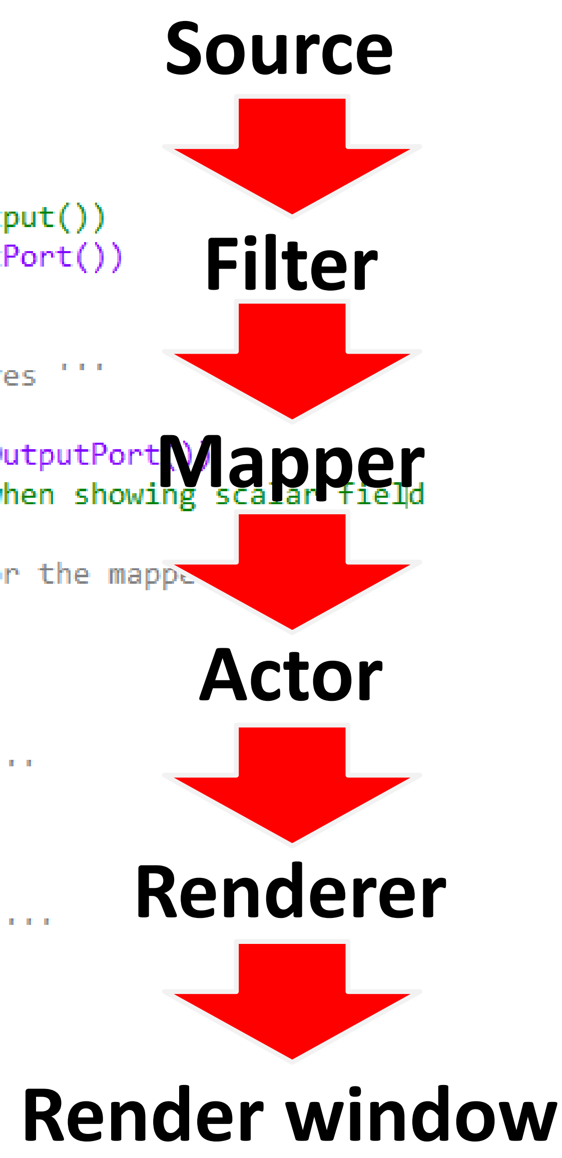
VTK pipeline



```
''' Step 1: Read a vtk data file '''
vtk_reader = vtk.vtkDataSetReader()
vtk_reader.SetFileName(input_file_name)

'''Step 2: Get geometry using a filter '''
vtk_geometry = vtk.vtkExtractEdges()
#vtk_geometry.SetInputData(vtk_reader.GetPolyDataOutput())
vtk_geometry.SetInputConnection(vtk_reader.GetOutputPort())
vtk_geometry.Update()

'''Step 3: use a mapper to get the geometry primitives '''
vtk_poly_mapper = vtk.vtkPolyDataMapper()
vtk_poly_mapper.SetInputConnection(vtk_geometry.GetOutputPort())
vtk_poly_mapper.ScalarVisibilityOff()#Turn this on when showing scalar field

'''Step 4: create an actor and set the appearance for the mappe
vtk_actor = vtk.vtkActor()
vtk_actor.SetMapper(vtk_poly_mapper)
vtk_actor.GetProperty().SetColor(1, 1, 0)

'''Step 5: create a render to set camera, lighting '''
render = vtk.vtkRenderer()
render.AddActor(vtk_actor)

'''Step 6: set the render window to show the result '''
window = vtk.vtkRenderWindow()
window.AddRenderer(render)
window.SetSize(600, 600)
```

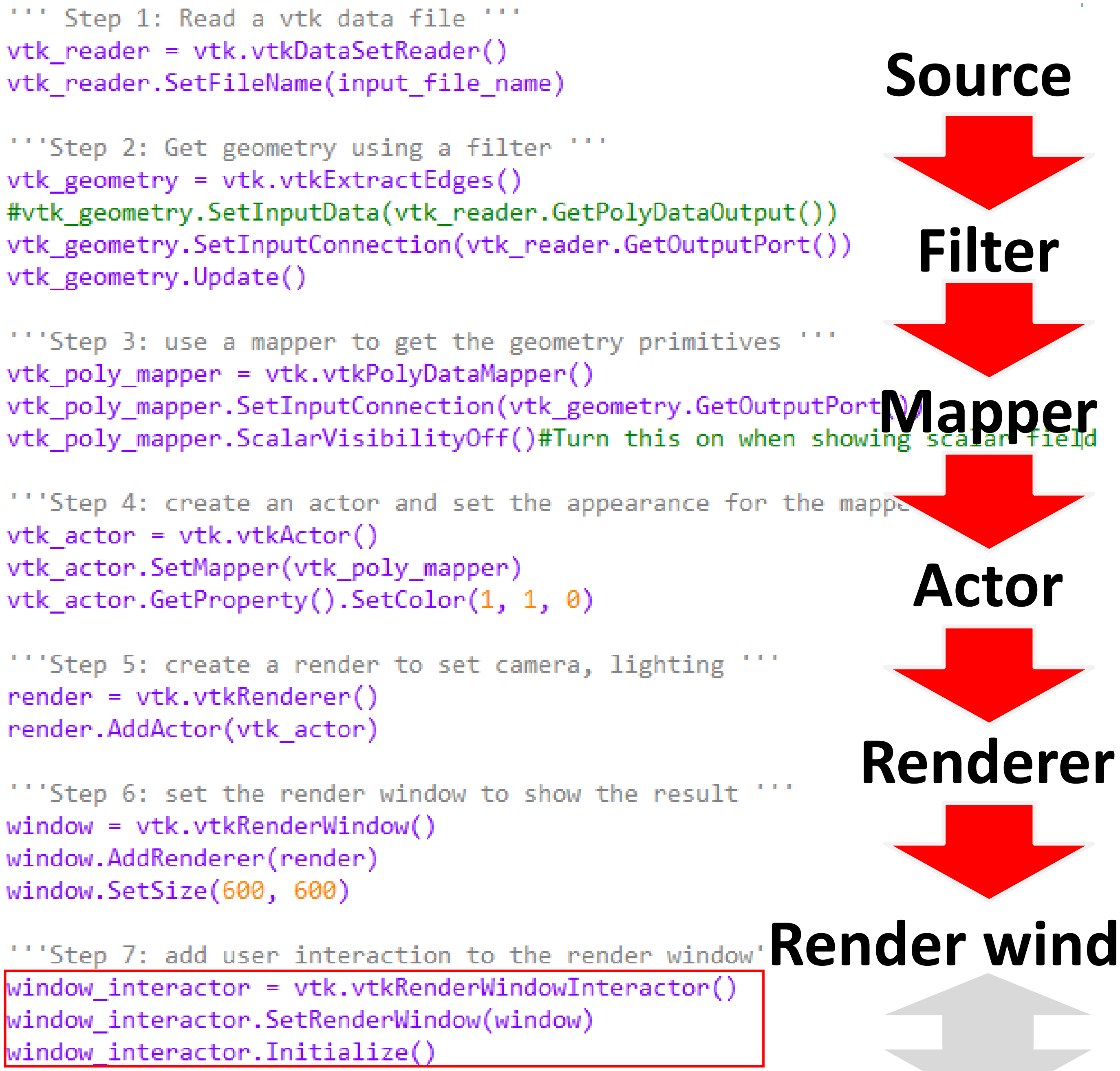
$\uparrow$

Set the resolution of the window.



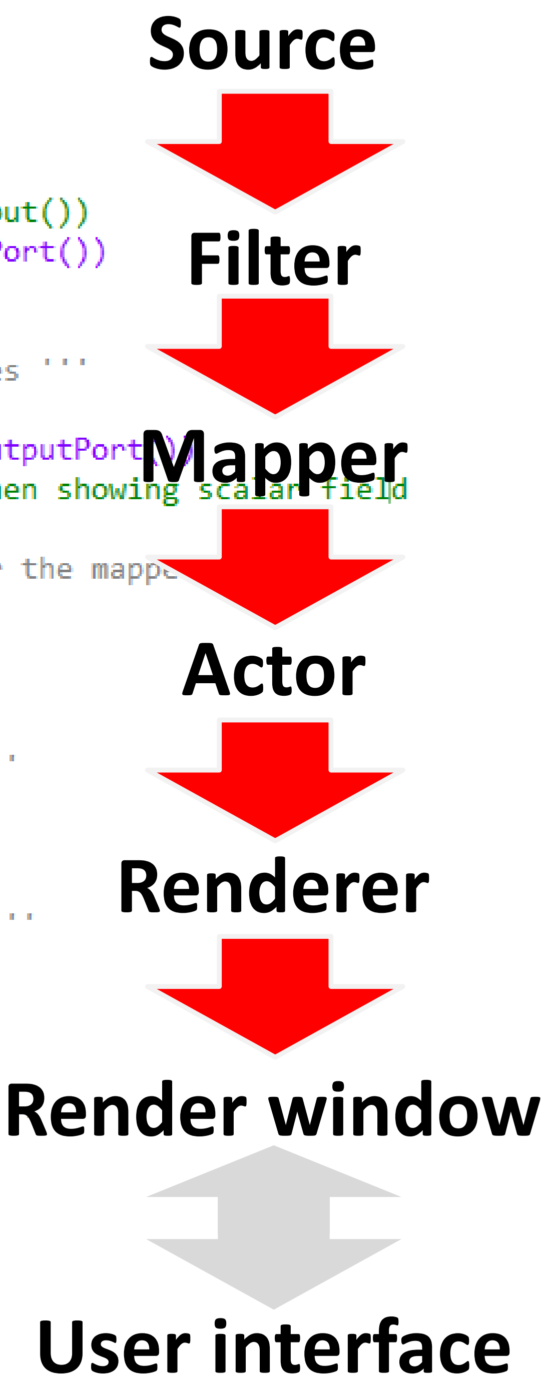
The **render window** displays the result on the screen and sets the resolution

VTK pipeline



```
''' Step 1: Read a vtk data file '''
vtk_reader = vtk.vtkDataSetReader()
vtk_reader.SetFileName(input_file_name)

'''Step 2: Get geometry using a filter '''
vtk_geometry = vtk.vtkExtractEdges()
#vtk_geometry.SetInputData(vtk_reader.GetPolyDataOutput())
vtk_geometry.SetInputConnection(vtk_reader.GetOutputPort())
vtk_geometry.Update()

'''Step 3: use a mapper to get the geometry primitives '''
vtk_poly_mapper = vtk.vtkPolyDataMapper()
vtk_poly_mapper.SetInputConnection(vtk_geometry.GetOutputPort())
vtk_poly_mapper.ScalarVisibilityOff()#Turn this on when showing scalar field

'''Step 4: create an actor and set the appearance for the mappe
vtk_actor = vtk.vtkActor()
vtk_actor.SetMapper(vtk_poly_mapper)
vtk_actor.GetProperty().SetColor(1, 1, 0)

'''Step 5: create a render to set camera, lighting '''
render = vtk.vtkRenderer()
render.AddActor(vtk_actor)

'''Step 6: set the render window to show the result '''
window = vtk.vtkRenderWindow()
window.AddRenderer(render)
window.SetSize(600, 600)

'''Step 7: add user interaction to the render window'
window_interactor = vtk.vtkRenderWindowInteractor()
window_interactor.SetRenderWindow(window)
window_interactor.Initialize()
```

Add some user interaction (via mouse) to the render window



VTK pipeline

```python
''' Step 1: Read a vtk data file '''
vtk_reader = vtk.vtkDataSetReader()
vtk_reader.SetFileName(input_file_name)

'''Step 2: Get geometry using a filter '''
vtk_geometry = vtk.vtkExtractEdges()
#vtk_geometry.SetInputData(vtk_reader.GetPolyDataOutput())
vtk_geometry.SetInputConnection(vtk_reader.GetOutputPort())
vtk_geometry.Update()

'''Step 3: use a mapper to get the geometry primitives '''
vtk_poly_mapper = vtk.vtkPolyDataMapper()
vtk_poly_mapper.SetInputConnection(vtk_geometry.GetOutputPort())
vtk_poly_mapper.ScalarVisibilityOff()#Turn this on when showing scalar field

'''Step 4: create an actor and set the appearance for the mapper
vtk_actor = vtk.vtkActor()
vtk_actor.SetMapper(vtk_poly_mapper)
vtk_actor.GetProperty().SetColor(1, 1, 0)

'''Step 5: create a render to set camera, lighting '''
render = vtk.vtkRenderer()
render.AddActor(vtk_actor)

'''Step 6: set the render window to show the result '''
window = vtk.vtkRenderWindow()
window.AddRenderer(render)
window.SetSize(600, 600)

'''Step 7: add user interaction to the render window'''
window_interactor = vtk.vtkRenderWindowInteractor()
window_interactor.SetRenderWindow(window)
window_interactor.Initialize()

'''Launch the window '''
window.Render()
```

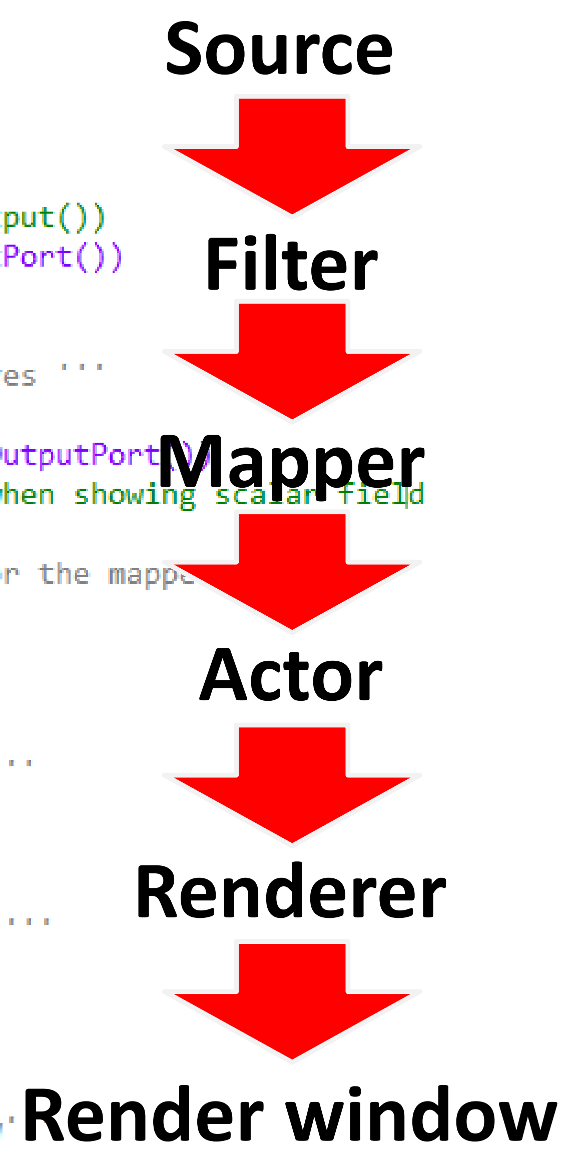
$$\begin{tikzcd}
\text{Source} \arrow[d] \\
\text{Filter} \arrow[d] \\
\text{Mapper} \arrow[d] \\
\text{Actor} \arrow[d] \\
\text{Renderer} \arrow[d] \\
\text{Render window}
\end{tikzcd}$$

Nothing will happen until Render() is called.

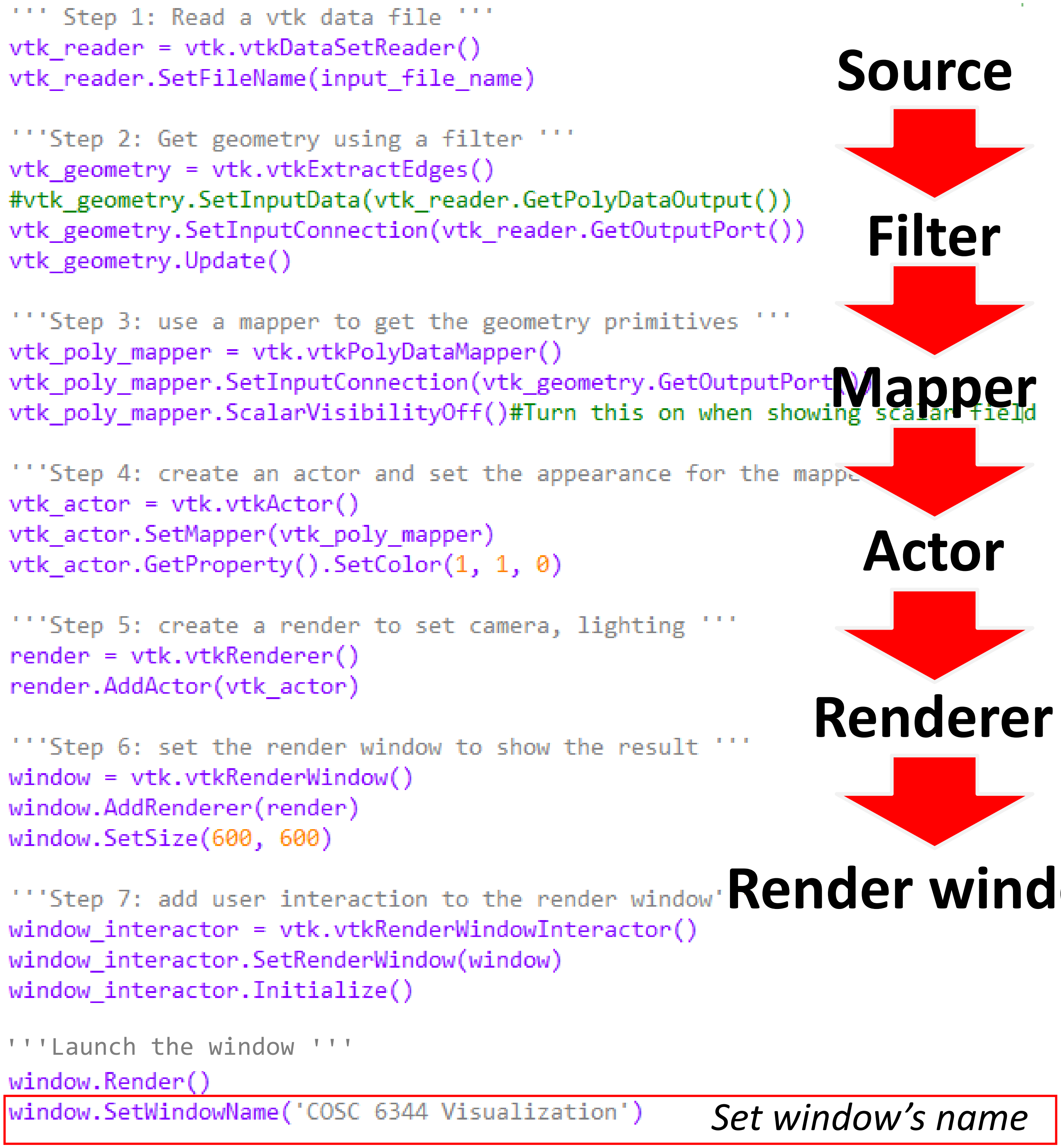
VTK pipeline

```
''' Step 1: Read a vtk data file '''
vtk_reader = vtk.vtkDataSetReader()
vtk_reader.SetFileName(input_file_name)

'''Step 2: Get geometry using a filter '''
vtk_geometry = vtk.vtkExtractEdges()
#vtk_geometry.SetInputData(vtk_reader.GetPolyDataOutput())
vtk_geometry.SetInputConnection(vtk_reader.GetOutputPort())
vtk_geometry.Update()

'''Step 3: use a mapper to get the geometry primitives '''
vtk_poly_mapper = vtk.vtkPolyDataMapper()
vtk_poly_mapper.SetInputConnection(vtk_geometry.GetOutputPort())
vtk_poly_mapper.ScalarVisibilityOff()#Turn this on when showing scalar field

'''Step 4: create an actor and set the appearance for the mappe
vtk_actor = vtk.vtkActor()
vtk_actor.SetMapper(vtk_poly_mapper)
vtk_actor.GetProperty().SetColor(1, 1, 0)

'''Step 5: create a render to set camera, lighting '''
render = vtk.vtkRenderer()
render.AddActor(vtk_actor)

'''Step 6: set the render window to show the result '''
window = vtk.vtkRenderWindow()
window.AddRenderer(render)
window.SetSize(600, 600)

'''Step 7: add user interaction to the render window'
window_interactor = vtk.vtkRenderWindowInteractor()
window_interactor.SetRenderWindow(window)
window_interactor.Initialize()

'''Launch the window '''
window.Render()
```

Once Render() is called.



VTK pipeline

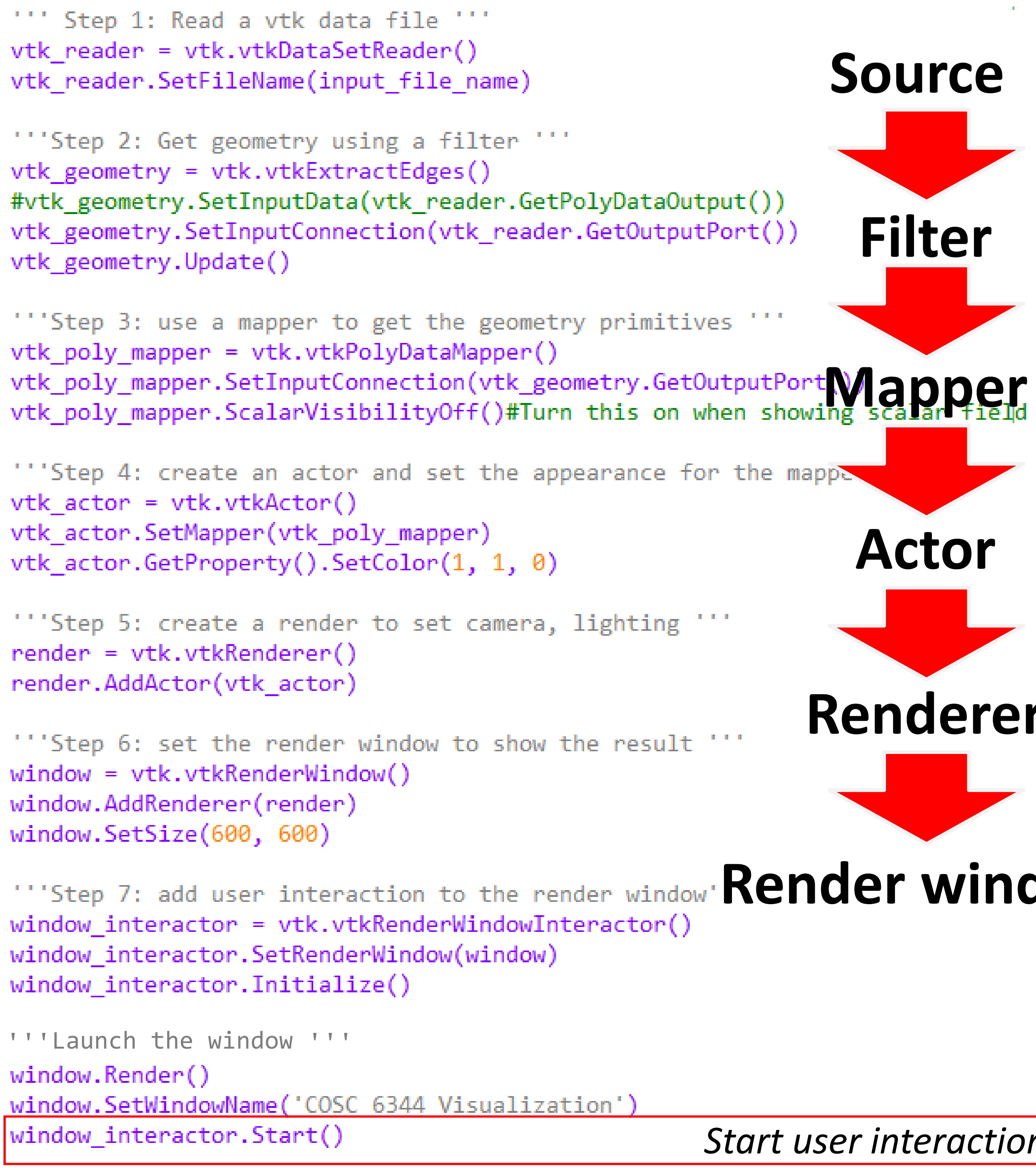


```
''' Step 1: Read a vtk data file '''
vtk_reader = vtk.vtkDataSetReader()
vtk_reader.SetFileName(input_file_name)

'''Step 2: Get geometry using a filter '''
vtk_geometry = vtk.vtkExtractEdges()
#vtk_geometry.SetInputData(vtk_reader.GetPolyDataOutput())
vtk_geometry.SetInputConnection(vtk_reader.GetOutputPort())
vtk_geometry.Update()

'''Step 3: use a mapper to get the geometry primitives '''
vtk_poly_mapper = vtk.vtkPolyDataMapper()
vtk_poly_mapper.SetInputConnection(vtk_geometry.GetOutputPort())
vtk_poly_mapper.ScalarVisibilityOff()#Turn this on when showing scalar field

'''Step 4: create an actor and set the appearance for the mappe
vtk_actor = vtk.vtkActor()
vtk_actor.SetMapper(vtk_poly_mapper)
vtk_actor.GetProperty().SetColor(1, 1, 0)

'''Step 5: create a render to set camera, lighting '''
render = vtk.vtkRenderer()
render.AddActor(vtk_actor)

'''Step 6: set the render window to show the result '''
window = vtk.vtkRenderWindow()
window.AddRenderer(render)
window.SetSize(600, 600)

'''Step 7: add user interaction to the render window'
window_interactor = vtk.vtkRenderWindowInteractor()
window_interactor.SetRenderWindow(window)
window_interactor.Initialize()

'''Launch the window '''
window.Render()
window.SetWindowName('COSC 6344 Visualization')
```

Set window's name

$$\begin{tikzcd}
\textbf{Source} \arrow[d] \\
\textbf{Filter} \arrow[d] \\
\textbf{Mapper} \arrow[d] \\
\textbf{Actor} \arrow[d] \\
\textbf{Renderer} \arrow[d] \\
\textbf{Render window}
\end{tikzcd}$$

VTK pipeline



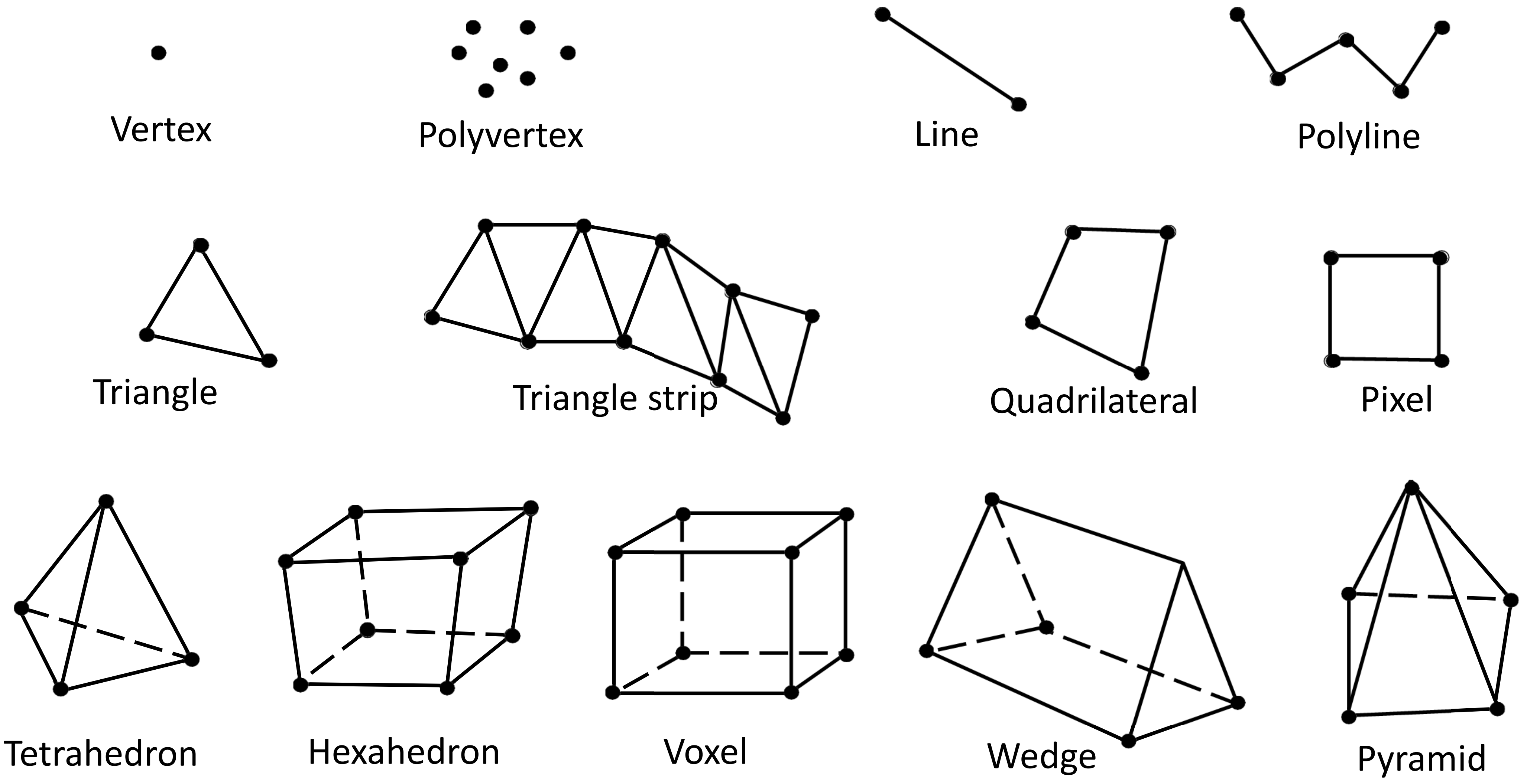
```
''' Step 1: Read a vtk data file '''
vtk_reader = vtk.vtkDataSetReader()
vtk_reader.SetFileName(input_file_name)

'''Step 2: Get geometry using a filter '''
vtk_geometry = vtk.vtkExtractEdges()
#vtk_geometry.SetInputData(vtk_reader.GetPolyDataOutput())
vtk_geometry.SetInputConnection(vtk_reader.GetOutputPort())
vtk_geometry.Update()

'''Step 3: use a mapper to get the geometry primitives '''
vtk_poly_mapper = vtk.vtkPolyDataMapper()
vtk_poly_mapper.SetInputConnection(vtk_geometry.GetOutputPort())
vtk_poly_mapper.ScalarVisibilityOff()#Turn this on when showing scalar field

'''Step 4: create an actor and set the appearance for the mappe
vtk_actor = vtk.vtkActor()
vtk_actor.SetMapper(vtk_poly_mapper)
vtk_actor.GetProperty().SetColor(1, 1, 0)

'''Step 5: create a render to set camera, lighting '''
render = vtk.vtkRenderer()
render.AddActor(vtk_actor)

'''Step 6: set the render window to show the result '''
window = vtk.vtkRenderWindow()
window.AddRenderer(render)
window.SetSize(600, 600)

'''Step 7: add user interaction to the render window'
window_interactor = vtk.vtkRenderWindowInteractor()
window_interactor.SetRenderWindow(window)
window_interactor.Initialize()

'''Launch the window '''
window.Render()
window.SetWindowName('COSC 6344 Visualization')
window_interactor.Start()
```

Start user interaction

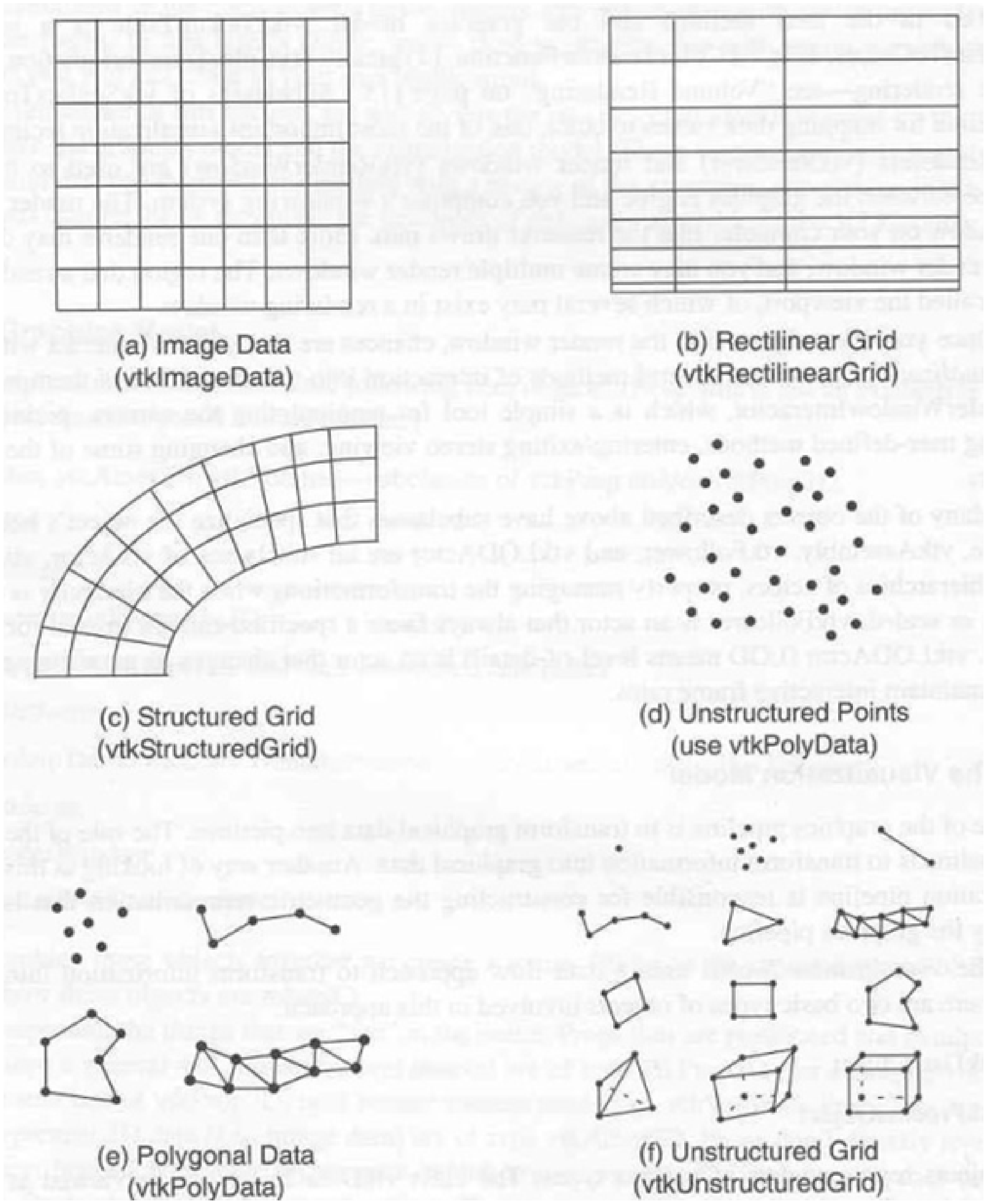
$$\begin{tikzcd}
\text{Source} \arrow[d] \\
\text{Filter} \arrow[d] \\
\text{Mapper} \arrow[d] \\
\text{Actor} \arrow[d] \\
\text{Renderer} \arrow[d] \\
\text{Render window}
\end{tikzcd}$$

Show demo

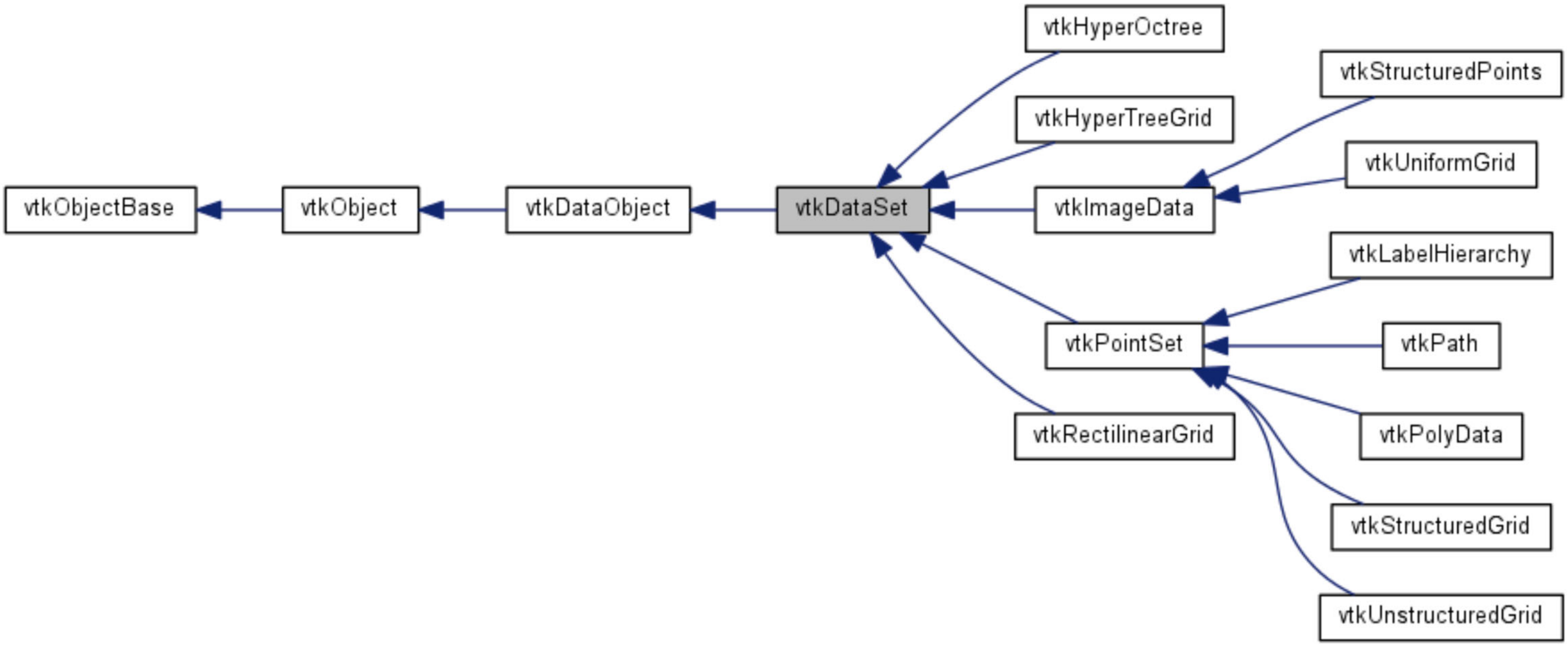
VTK Cell Types



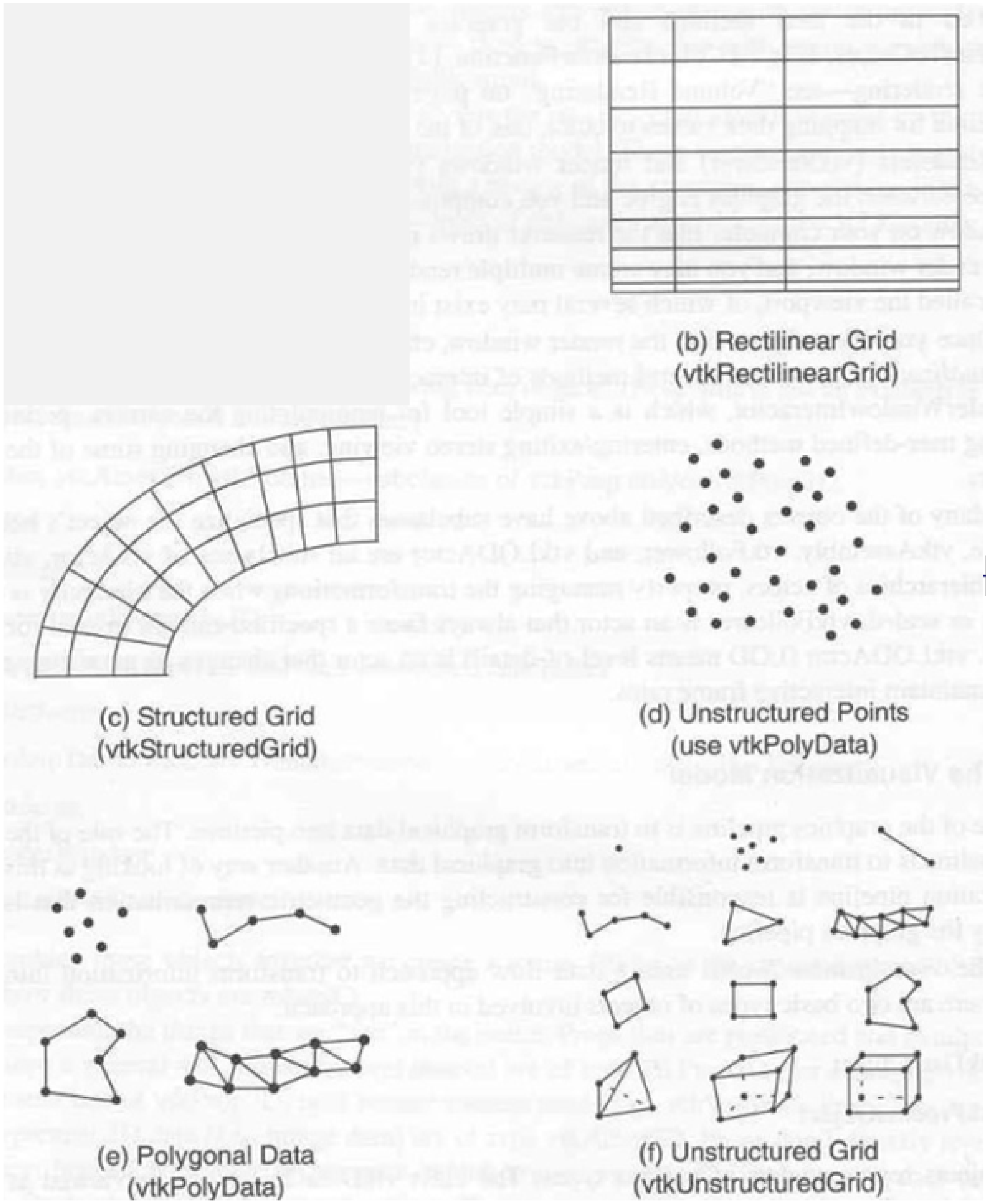
VTK Dataset Types

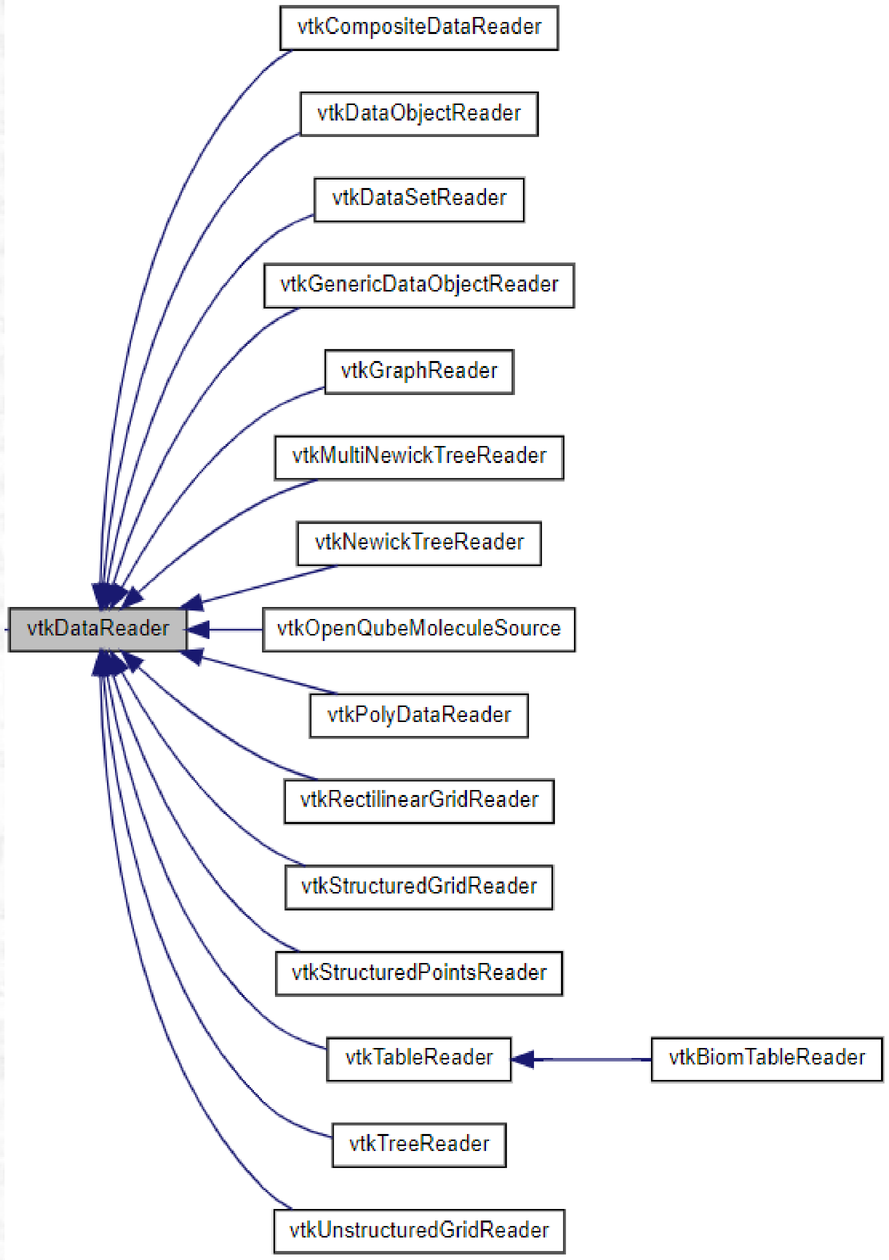


VTK Dataset Hierarchy

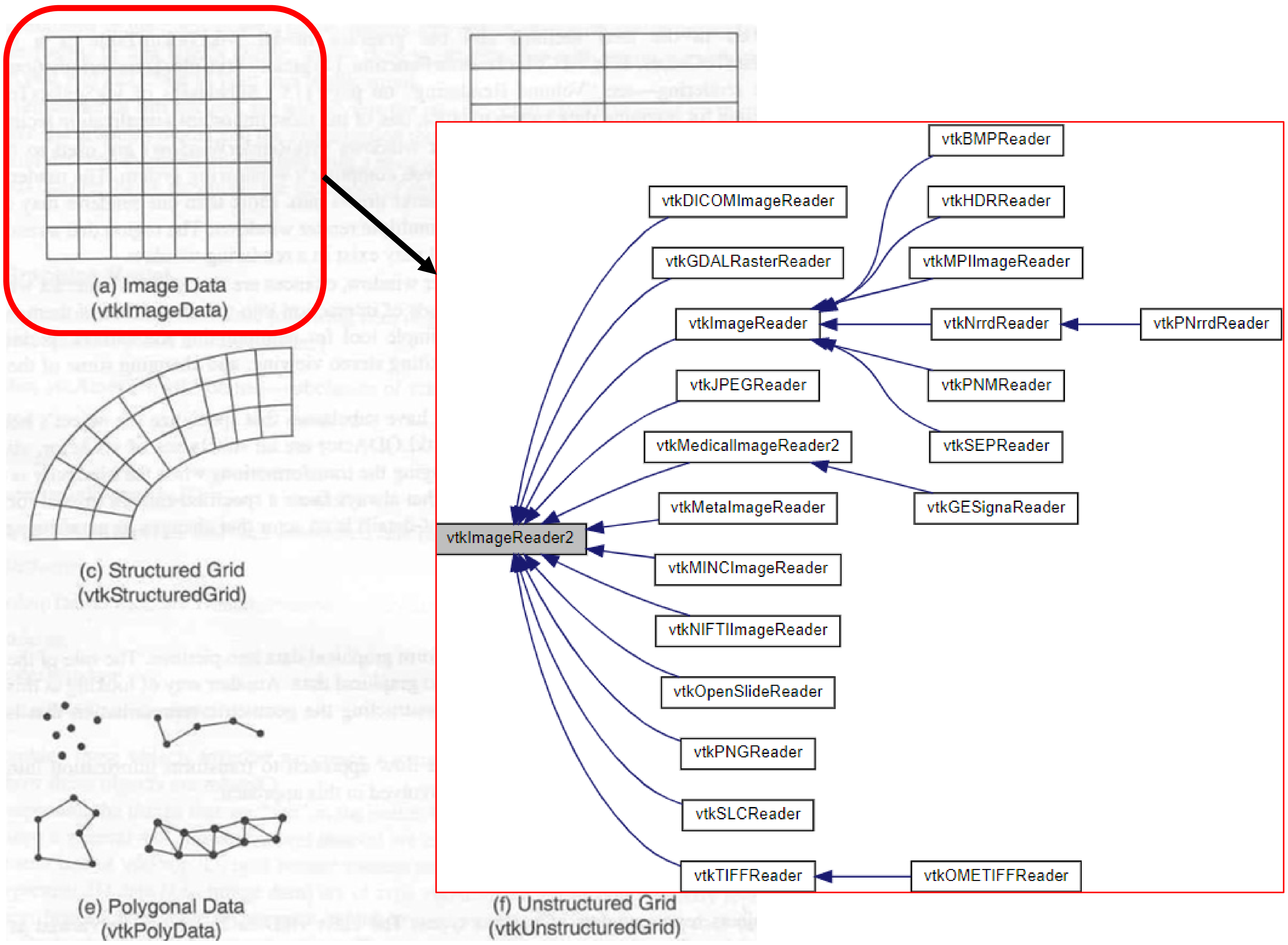


VTK Dataset Types

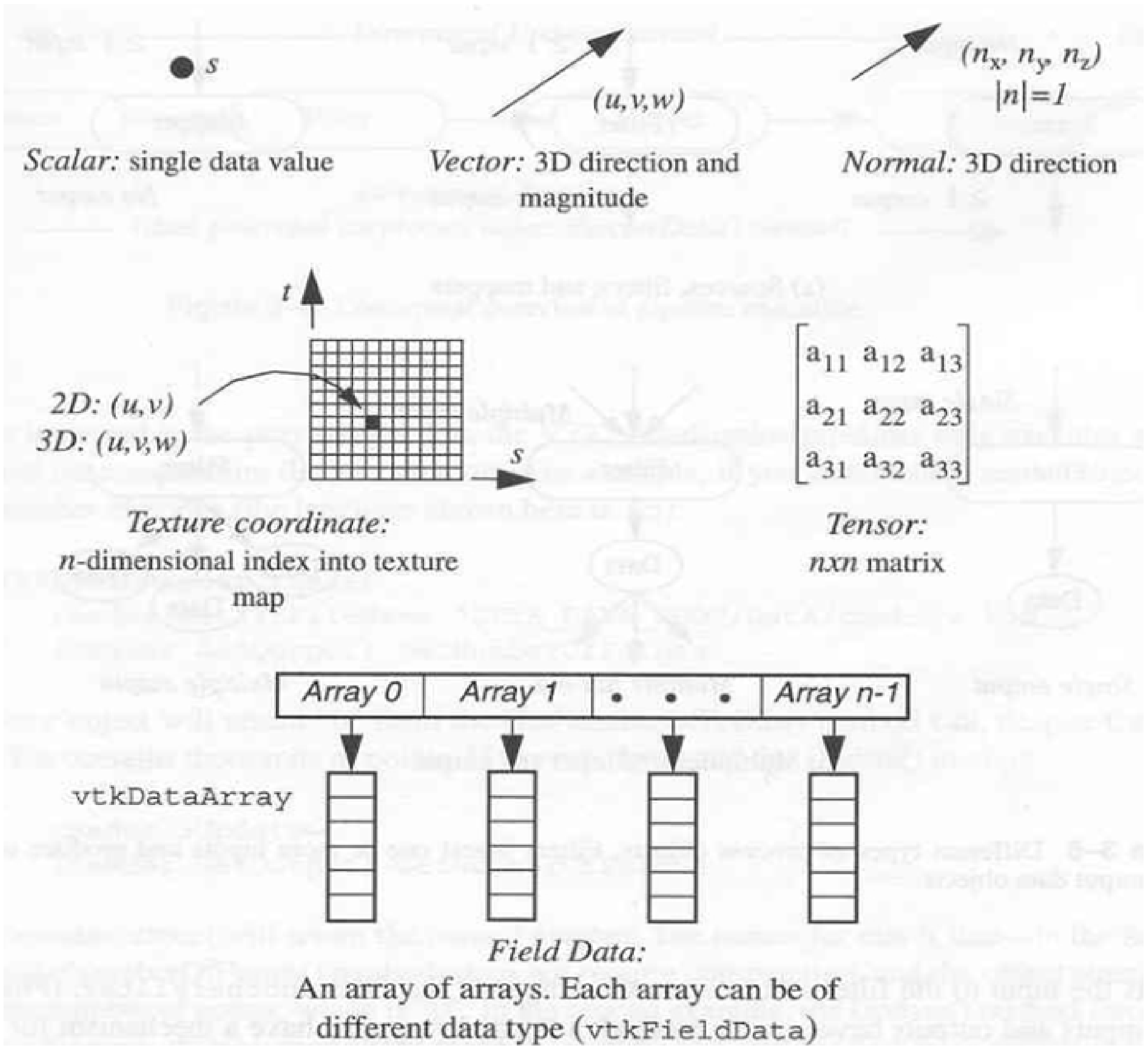




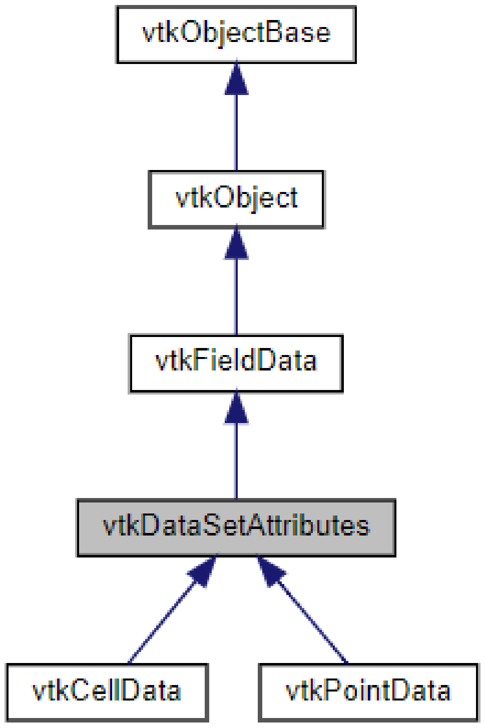
VTK Dataset Types

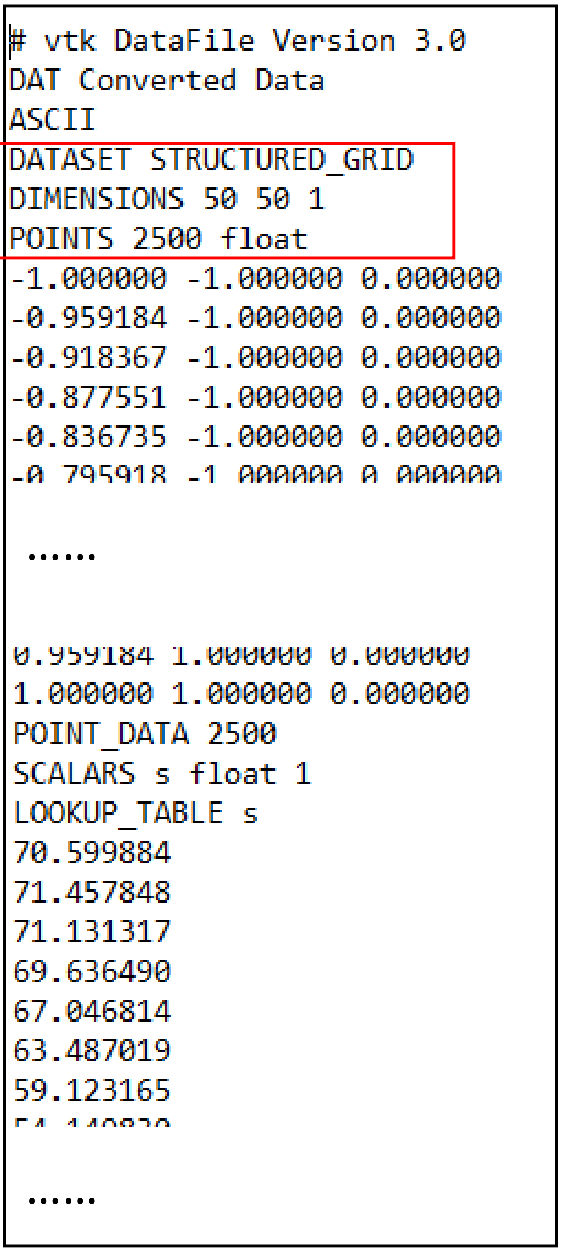


Attribute Types

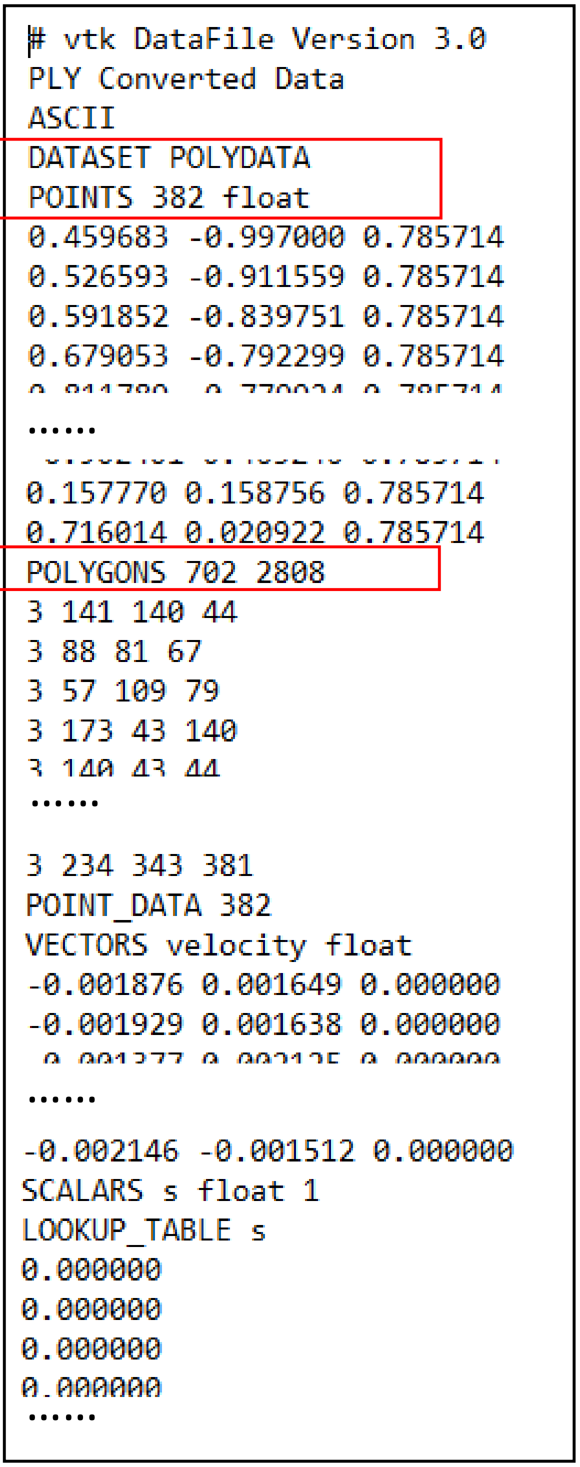


s
Scalar: single data value
(u,v,w)
Vector: 3D direction and magnitude
(n_x, n_y, n_z)
$|n|=1$
Normal: 3D direction
t
s
2D: (u,v)
3D: (u,v,w)
Texture coordinate: n-dimensional index into texture map
$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$
Tensor: nxn matrix
Array 0
Array 1
Array n-1
vtkDataArray
Field Data: An array of arrays. Each array can be of different data type (vtkFieldData)

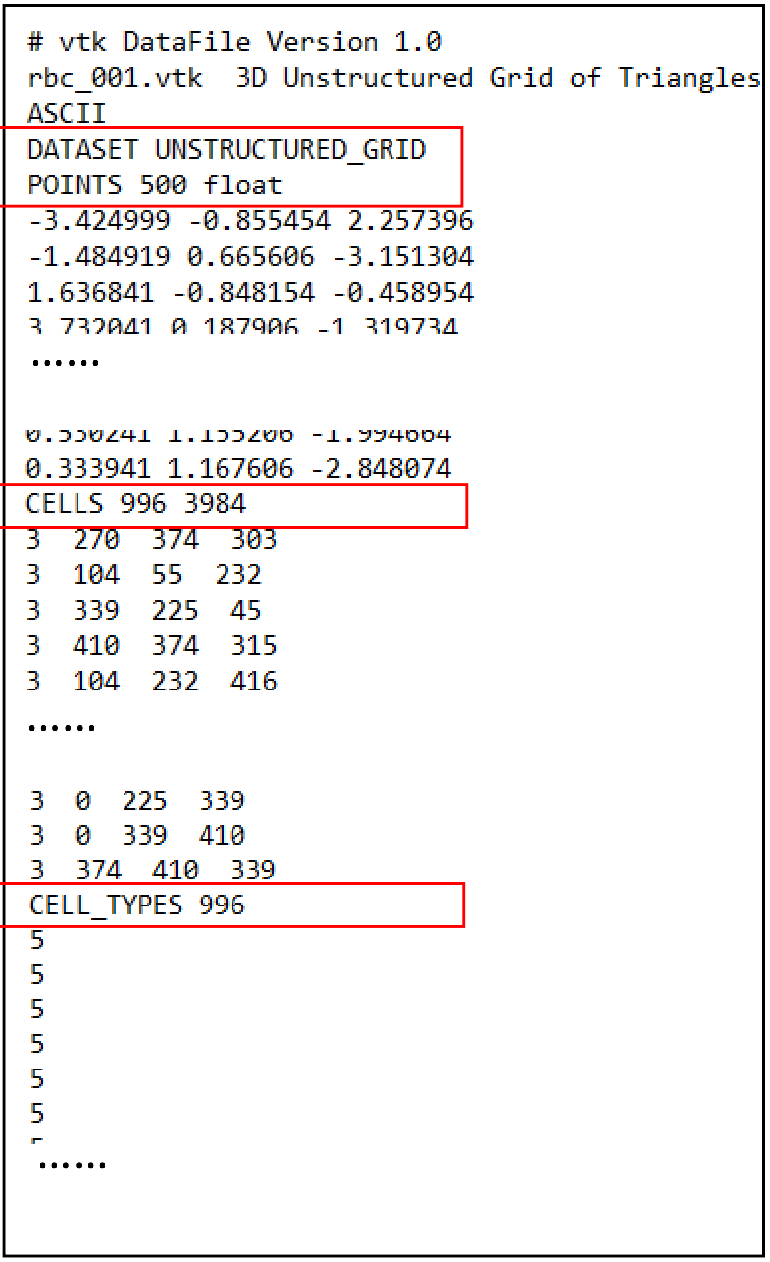




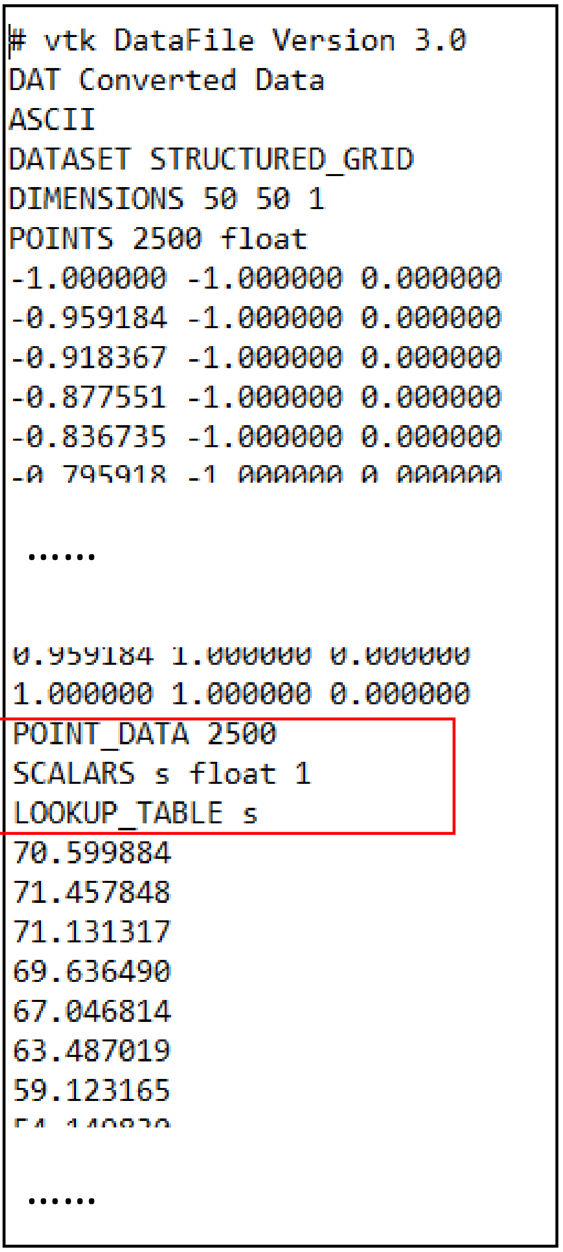
```
# vtk DataFile Version 3.0
DAT Converted Data
ASCII
DATASET STRUCTURED_GRID
DIMENSIONS 50 50 1
POINTS 2500 float
-1.000000 -1.000000 0.000000
-0.959184 -1.000000 0.000000
-0.918367 -1.000000 0.000000
-0.877551 -1.000000 0.000000
-0.836735 -1.000000 0.000000
-0.795918 -1.000000 0.000000

 ......

0.959184 1.000000 0.000000
1.000000 1.000000 0.000000
POINT_DATA 2500
SCALARS s float 1
LOOKUP_TABLE s
70.599884
71.457848
71.131317
69.636490
67.046814
63.487019
59.123165
[illegible]

 ......
```



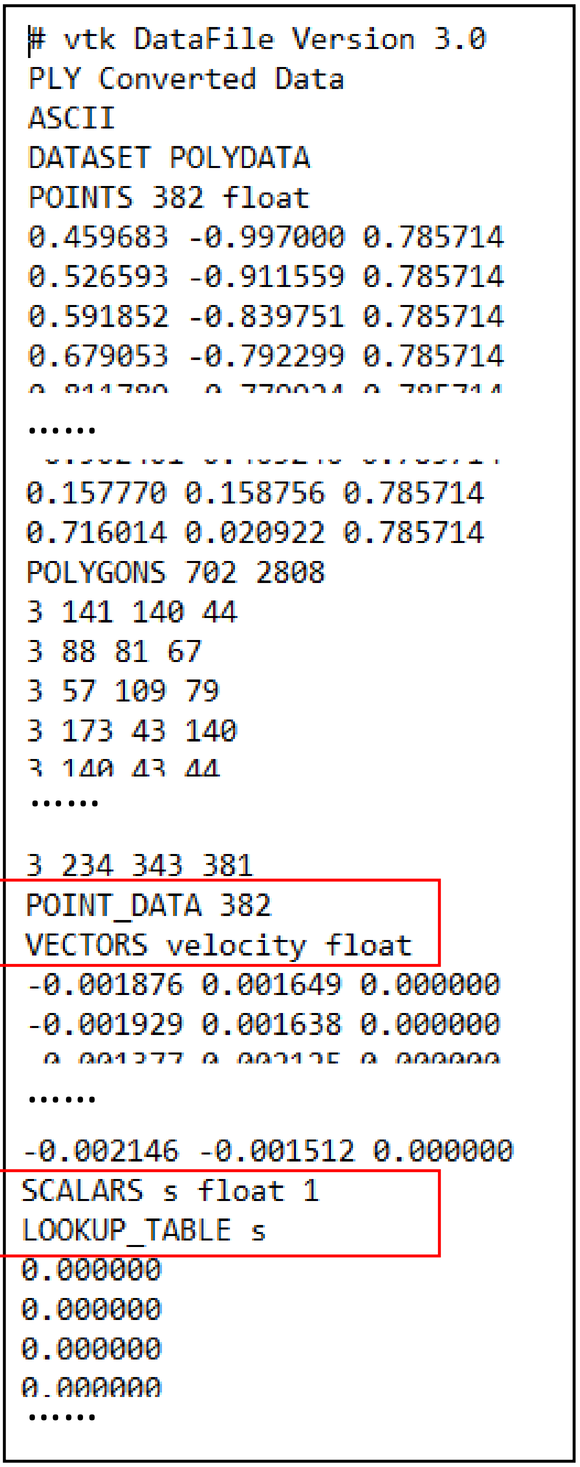
```
# vtk DataFile Version 3.0
PLY Converted Data
ASCII
DATASET POLYDATA
POINTS 382 float
0.459683 -0.997000 0.785714
0.526593 -0.911559 0.785714
0.591852 -0.839751 0.785714
0.679053 -0.792299 0.785714
0.811780 -0.770024 0.785714
......
[illegible]
0.157770 0.158756 0.785714
0.716014 0.020922 0.785714
POLYGONS 702 2808
3 141 140 44
3 88 81 67
3 57 109 79
3 173 43 140
3 140 43 44
......

3 234 343 381
POINT_DATA 382
VECTORS velocity float
-0.001876 0.001649 0.000000
-0.001929 0.001638 0.000000
-0.001377 0.002125 0.000000
......

-0.002146 -0.001512 0.000000
SCALARS s float 1
LOOKUP_TABLE s
0.000000
0.000000
0.000000
0.000000
......
```



```
# vtk DataFile Version 1.0
rbc_001.vtk  3D Unstructured Grid of Triangles
ASCII
DATASET UNSTRUCTURED_GRID
POINTS 500 float
-3.424999 -0.855454 2.257396
-1.484919 0.665606 -3.151304
1.636841 -0.848154 -0.458954
3.732041 0.187906 -1.319734
......

0.550241 1.155206 -1.994664
0.333941 1.167606 -2.848074
CELLS 996 3984
3  270  374  303
3  104  55  232
3  339  225  45
3  410  374  315
3  104  232  416
......

3  0  225  339
3  0  339  410
3  374  410  339
CELL_TYPES 996
5
5
5
5
5
5
......
```



```
# vtk DataFile Version 3.0
DAT Converted Data
ASCII
DATASET STRUCTURED_GRID
DIMENSIONS 50 50 1
POINTS 2500 float
-1.000000 -1.000000 0.000000
-0.959184 -1.000000 0.000000
-0.918367 -1.000000 0.000000
-0.877551 -1.000000 0.000000
-0.836735 -1.000000 0.000000
-0.795918 -1.000000 0.000000

 ......

0.959184 1.000000 0.000000
1.000000 1.000000 0.000000
POINT_DATA 2500
SCALARS s float 1
LOOKUP_TABLE s
70.599884
71.457848
71.131317
69.636490
67.046814
63.487019
59.123165

 ......
```



```
# vtk DataFile Version 3.0
PLY Converted Data
ASCII
DATASET POLYDATA
POINTS 382 float
0.459683 -0.997000 0.785714
0.526593 -0.911559 0.785714
0.591852 -0.839751 0.785714
0.679053 -0.792299 0.785714
......
0.157770 0.158756 0.785714
0.716014 0.020922 0.785714
POLYGONS 702 2808
3 141 140 44
3 88 81 67
3 57 109 79
3 173 43 140
3 140 43 44
......

3 234 343 381
POINT_DATA 382
VECTORS velocity float
-0.001876 0.001649 0.000000
-0.001929 0.001638 0.000000
-0.001377 0.002125 0.000000
......

-0.002146 -0.001512 0.000000
SCALARS s float 1
LOOKUP_TABLE s
0.000000
0.000000
0.000000
0.000000
......
```

Example of getting value range of a scalar field

```
vtk_geometry.GetOutput().GetPointData().GetArray(scalar_field).GetRange()
```

Grid data

Data stored at points

name of the scalar field

Example of getting all scalar values

```
allscalars = vtk_geometry.GetOutput().GetPointData().GetScalars(''s'')
```

name of the scalar field

Example of getting all vector values

```
allvectors = vtk_reader.GetOutput().GetPointData().GetVectors(''velocity'')
```

name of the vector field

Additional References

- VTK Wiki http://www.vtk.org/Wiki/VTK

- VTK Examples
  - Python: https://lorensen.github.io/VTKExamples/site/Python/
  - C++: https://vtk.org/Wiki/VTK/Examples/Cxx

- Books:
  - VTK User's Guide, Kitware Inc. ISBN 1-930934-0804 (https://www.kitware.com/products/books/VTKUsersGuide.pdf)

  - The Visualization Toolkit, An object-orented Approach to 3D Graphics, 4th edition, by W. Schroeder, K. Martin, B. Lorensen, Kitware
    Online version
    https://lorensen.github.io/VTKExamples/site/VTKBook/
    download:
    https://raw.githubusercontent.com/lorensen/VTKExamples/master/src/VTKBookLaTeX/VTKTextBook.pdf