CS53000 - Spring 2018

Introduction to Scientific Visualization

Introduction to vtk

January 11, 2018

The Visualization Toolkit

- Open source library for
  - *Visualization*
  - *Computer Graphics*
  - *Imaging*
- Written in C++
- Supports scripting in *Python* and *Java* via wrappers

Outline

- Visualization pipeline
- Internal data representation
- Examples

Outline

- Visualization pipeline
- Internal data representation
- Examples

Visualization Pipeline

- In VTK, visualizations are created by pipelines

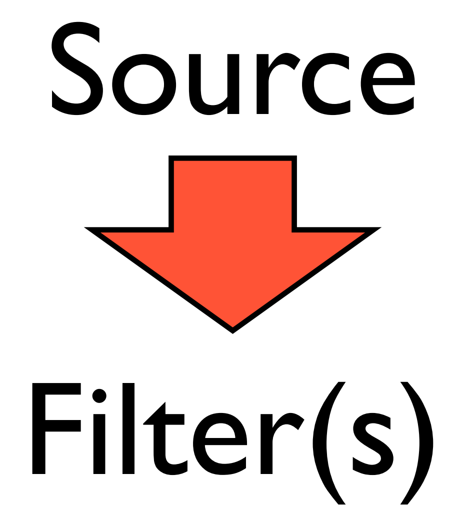
Visualization Pipeline

- In VTK, visualizations are created by pipelines:
  - The *source* imports (from file) or creates (e.g., function) the data

Source

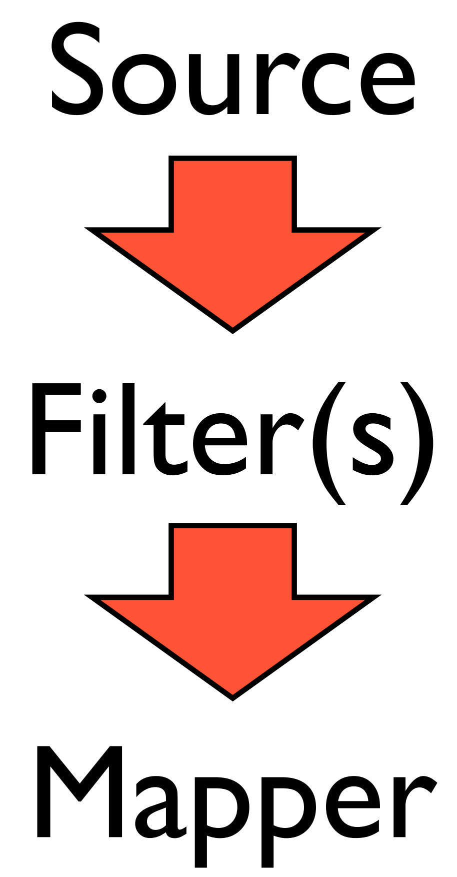
Visualization Pipeline

- In VTK, visualizations are created by pipelines:
  - One or more *filters* process the data to create geometric objects



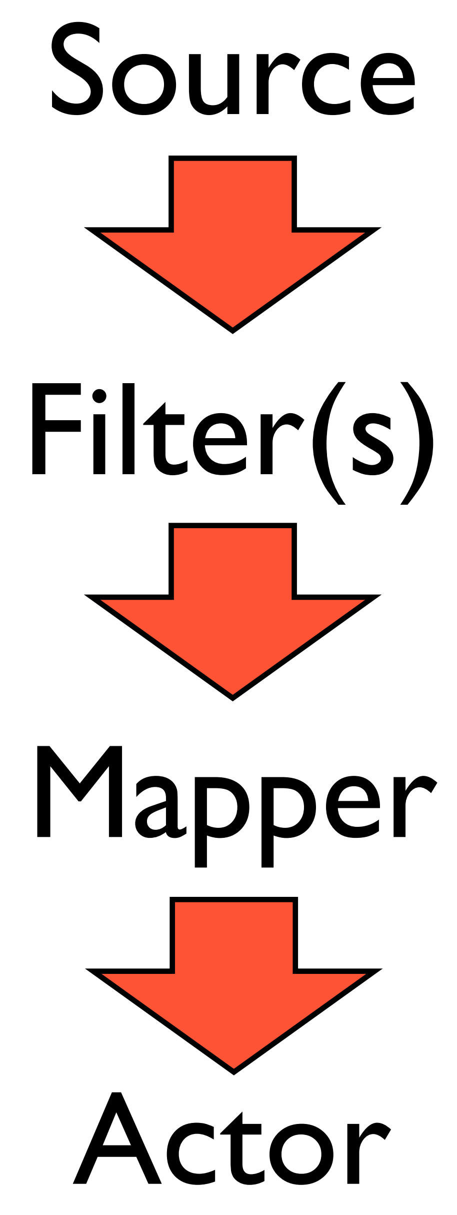
Visualization Pipeline

- In VTK, visualizations are created by pipelines:
  - The *mapper* converts geometry to graphical primitives



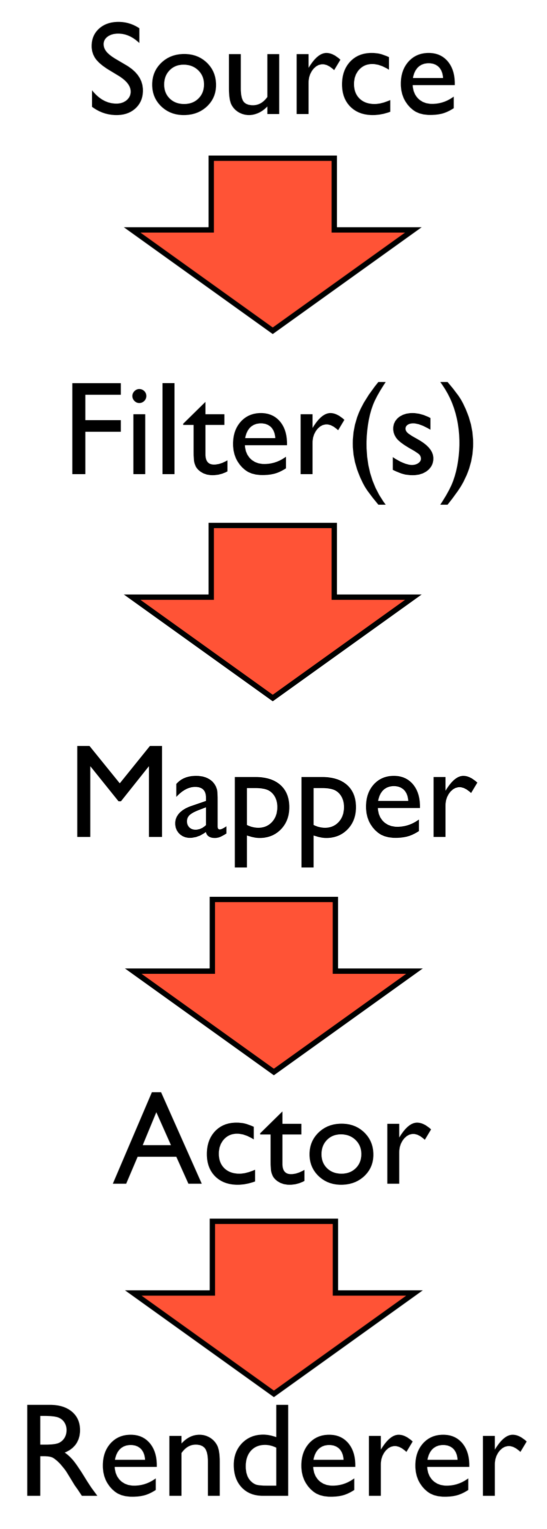
Visualization Pipeline

- In VTK, visualizations are created by pipelines:
    - The *actor* positions the primitives in the scene and controls their appearance



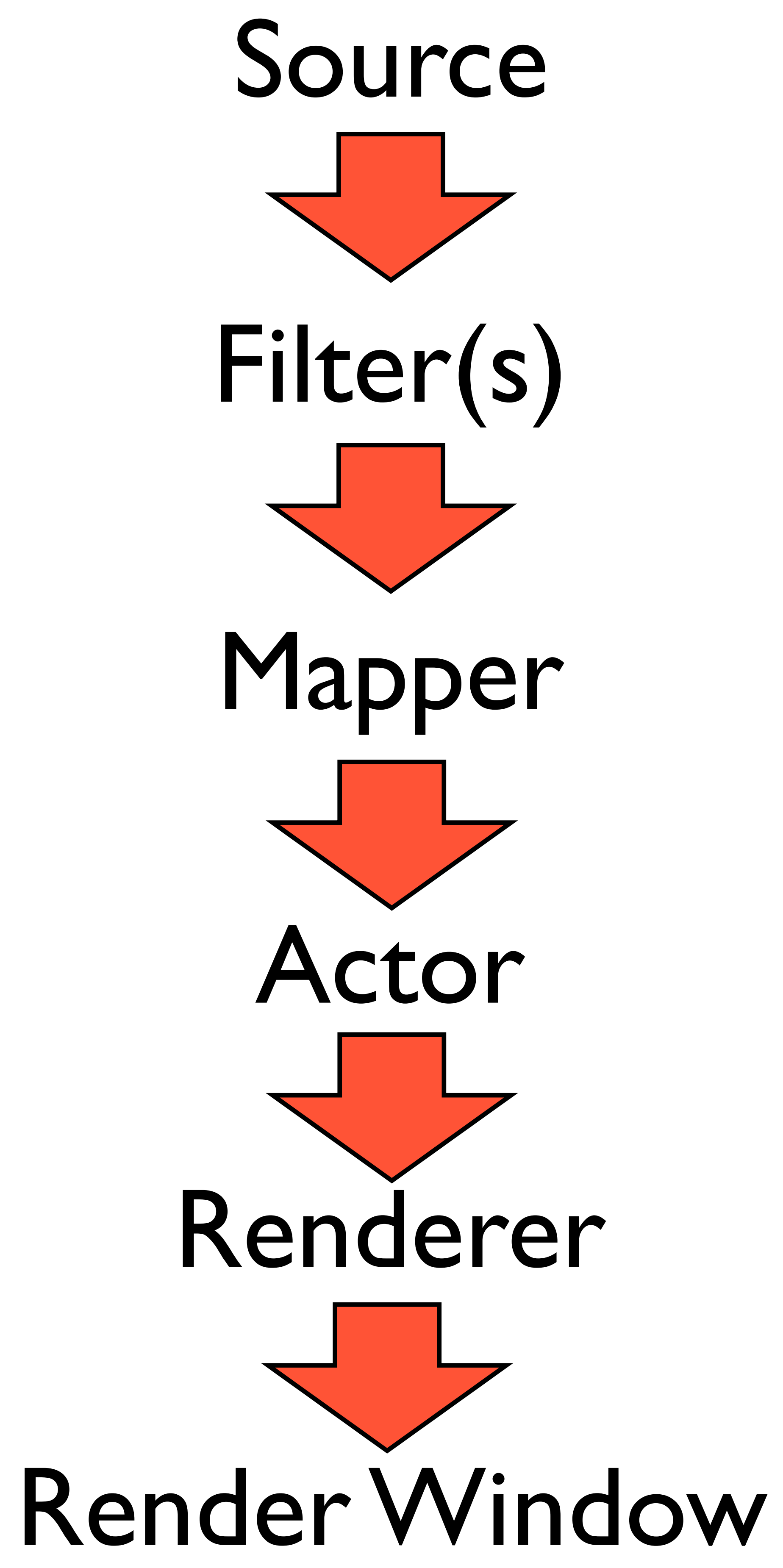
Visualization Pipeline

- In VTK, visualizations are created by pipelines:
  - The *renderer* controls the camera and the lighting



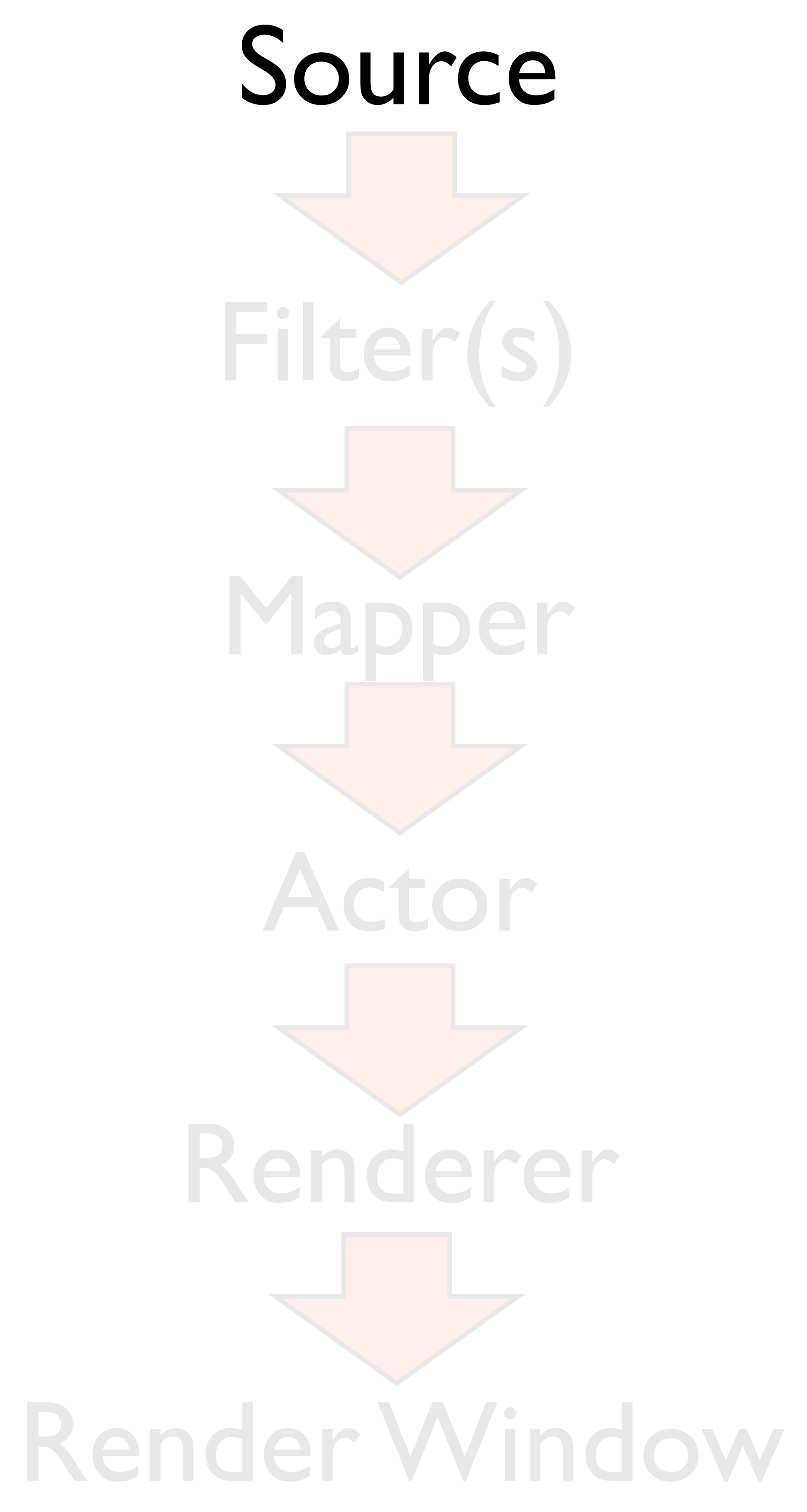
Visualization Pipeline

- In VTK, visualizations are created by pipelines:
    - The *render window* displays the result on the screen and sets the resolution



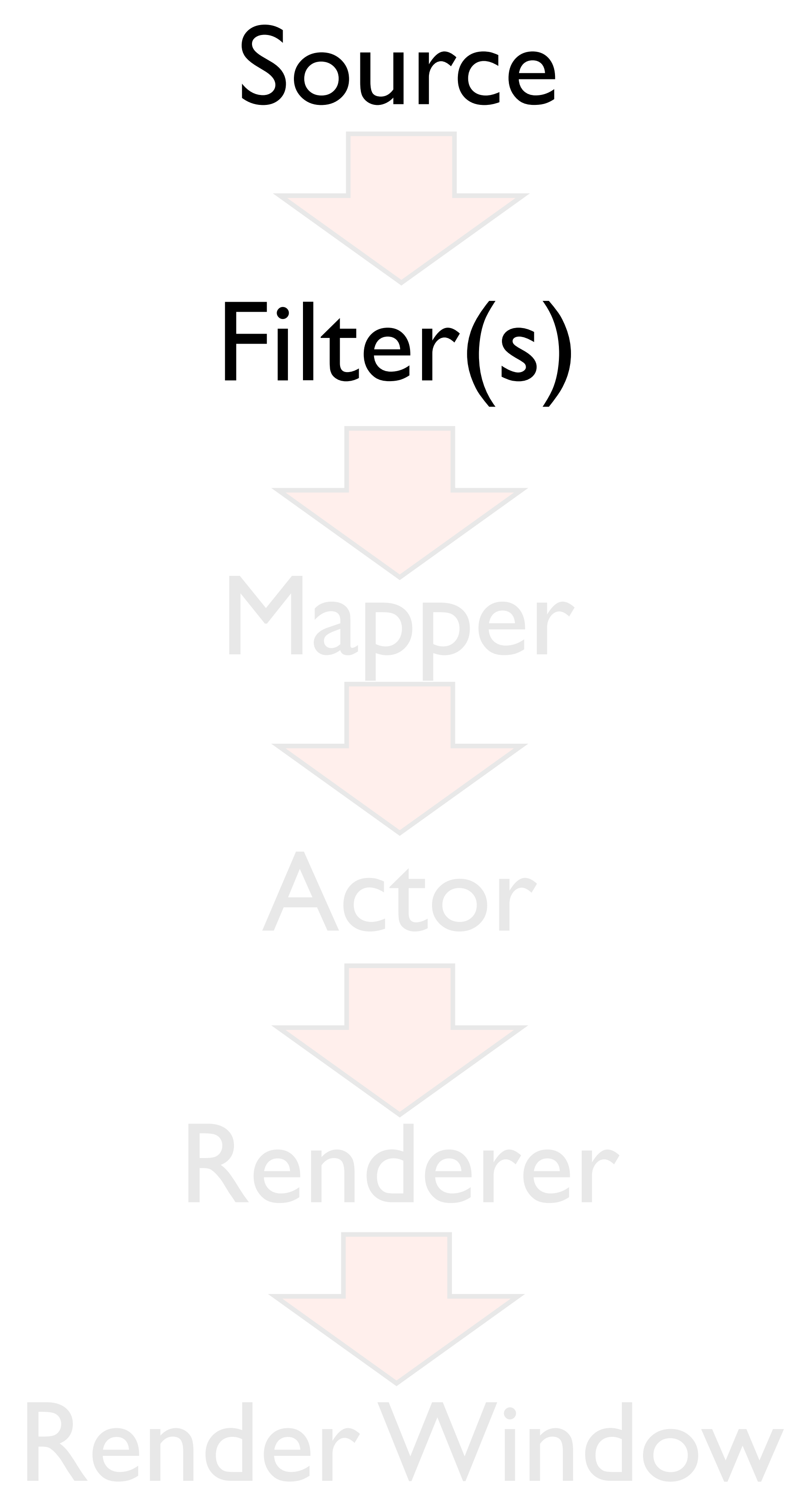
```
reader = vtk.vtkDataSetReader()
reader.SetFileName(filename)
```

Read data from file



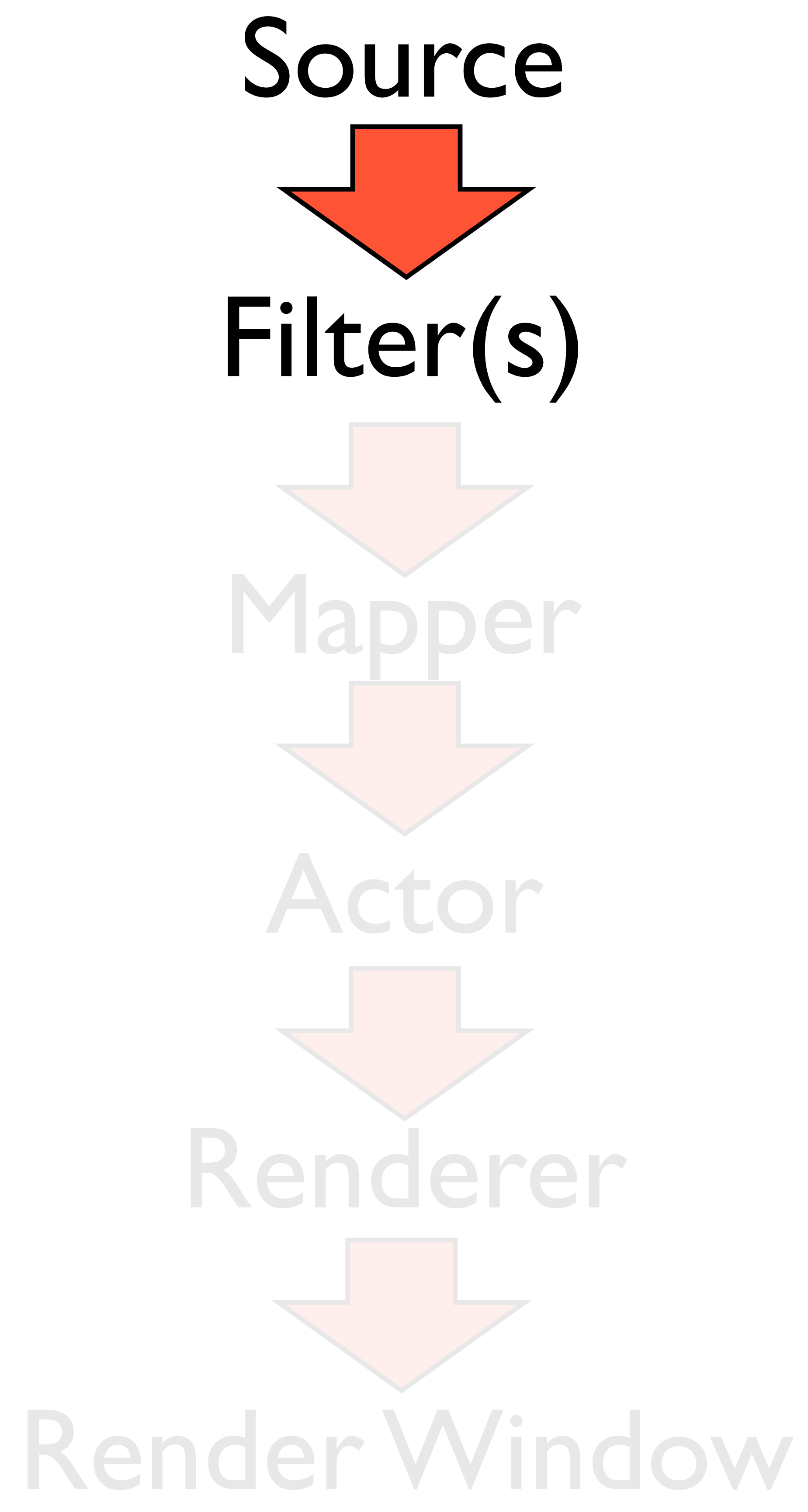
```
reader = vtk.vtkDataSetReader()
reader.SetFileName(filename)

contour = vtk.vtkContourFilter()
contour.SetValue(0, float(value))
```

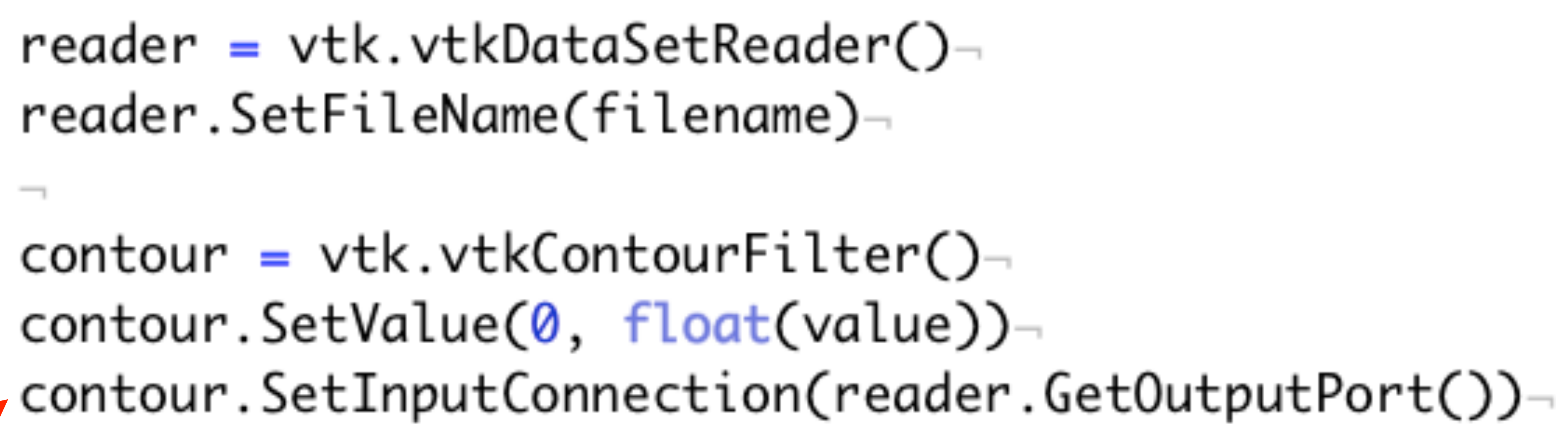
Create a visualization filter
and sets its parameter



```
reader = vtk.vtkDataSetReader()
reader.SetFileName(filename)

contour = vtk.vtkContourFilter()
contour.SetValue(0, float(value))
contour.SetInputConnection(reader.GetOutputPort())
```

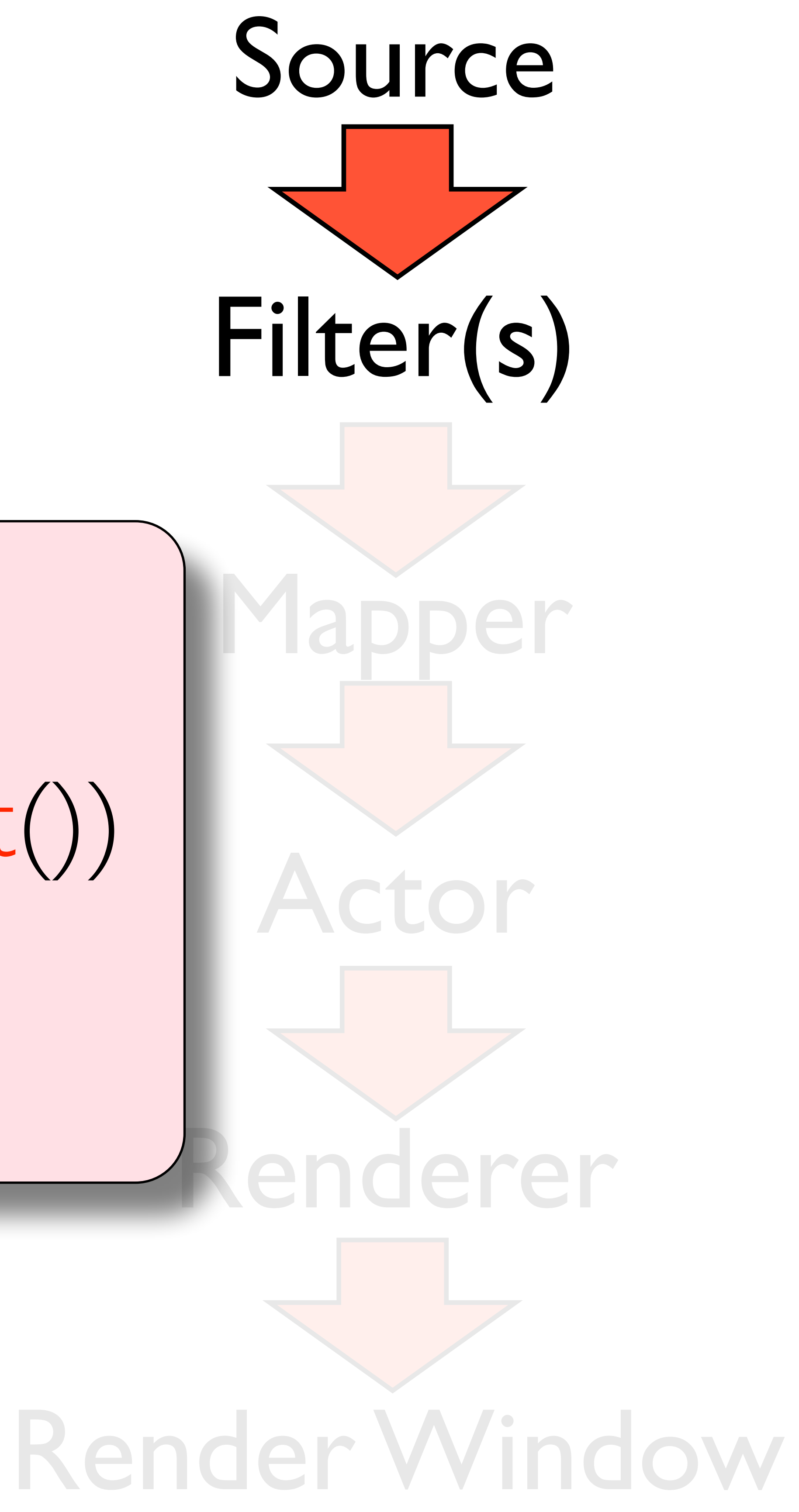
Apply visualization filter to our data





```
reader = vtk.vtkDataSetReader()
reader.SetFileName(filename)

contour = vtk.vtkContourFilter()
contour.SetValue(0, float(value))
contour.SetInputConnection(reader.GetOutputPort())
```

Apply visualization filter to our data



VTK pipeline connection syntax:

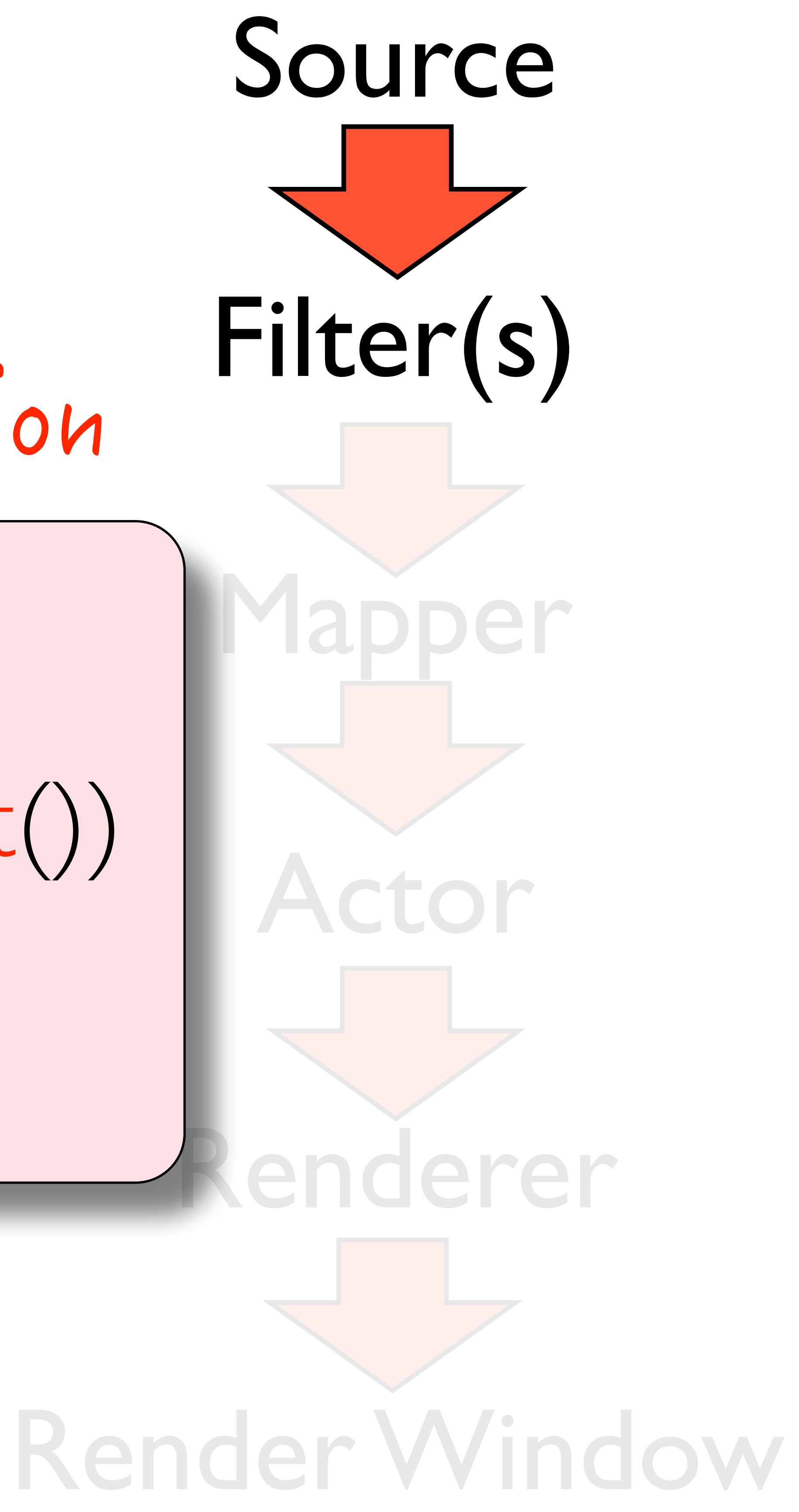
(1) *Receiver*.SetInputConnection(*Supplier*.GetOutputPort())

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

```
reader = vtk.vtkDataSetReader()
reader.SetFileName(filename)

contour = vtk.vtkContourFilter()
contour.SetValue(0, float(value))
contour.SetInputConnection(reader.GetOutputPort())
```

Apply visualization filter to our data

pipeline connection



VTK pipeline connection syntax:

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

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

```
reader = vtk.vtkDataSetReader()
reader.SetFileName(filename)

contour = vtk.vtkContourFilter()
contour.SetValue(0, float(value))
contour.SetInputConnection(reader.GetOutputPort())
```

Apply visualization filter to our data

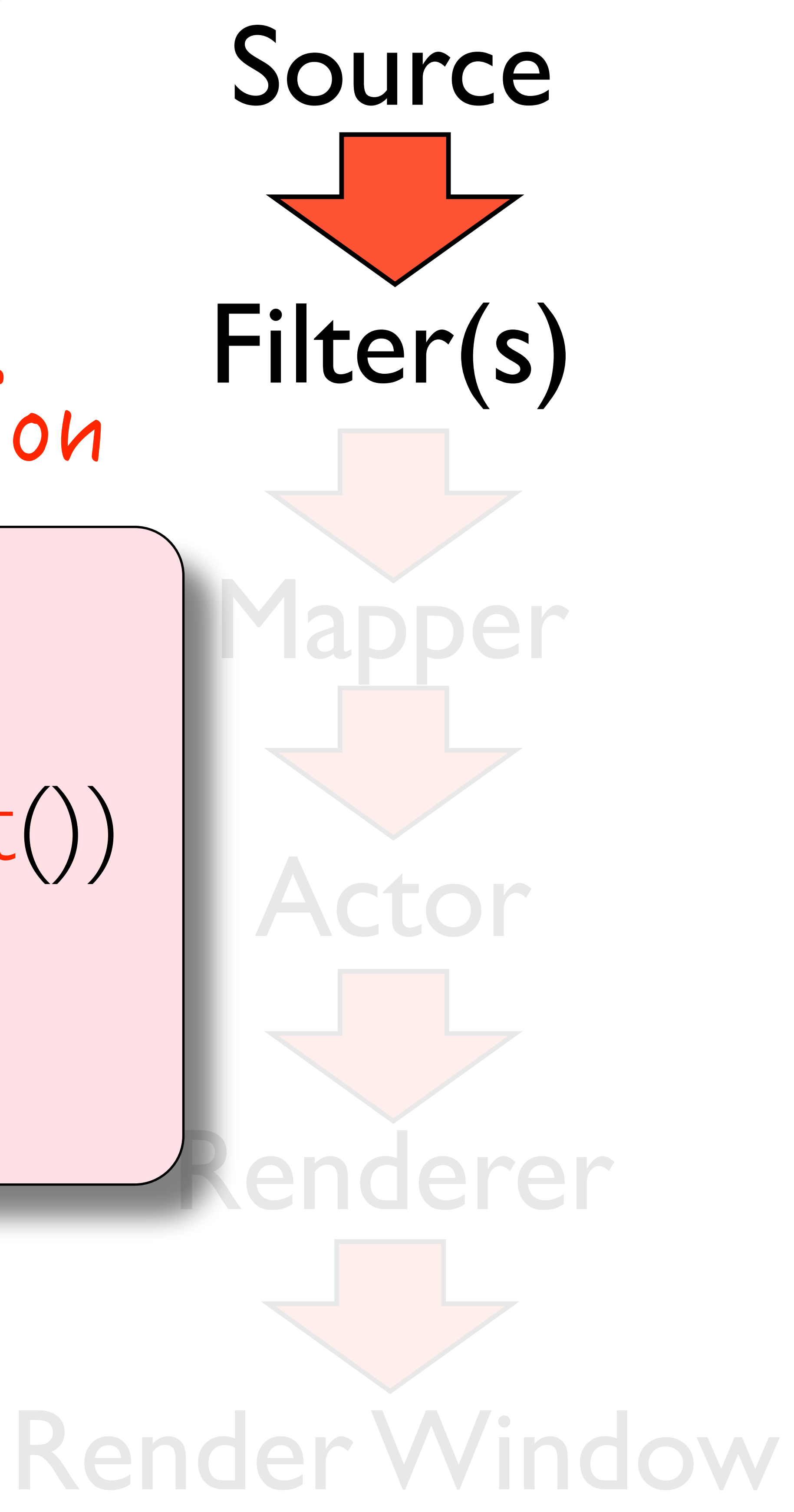
pipeline connection

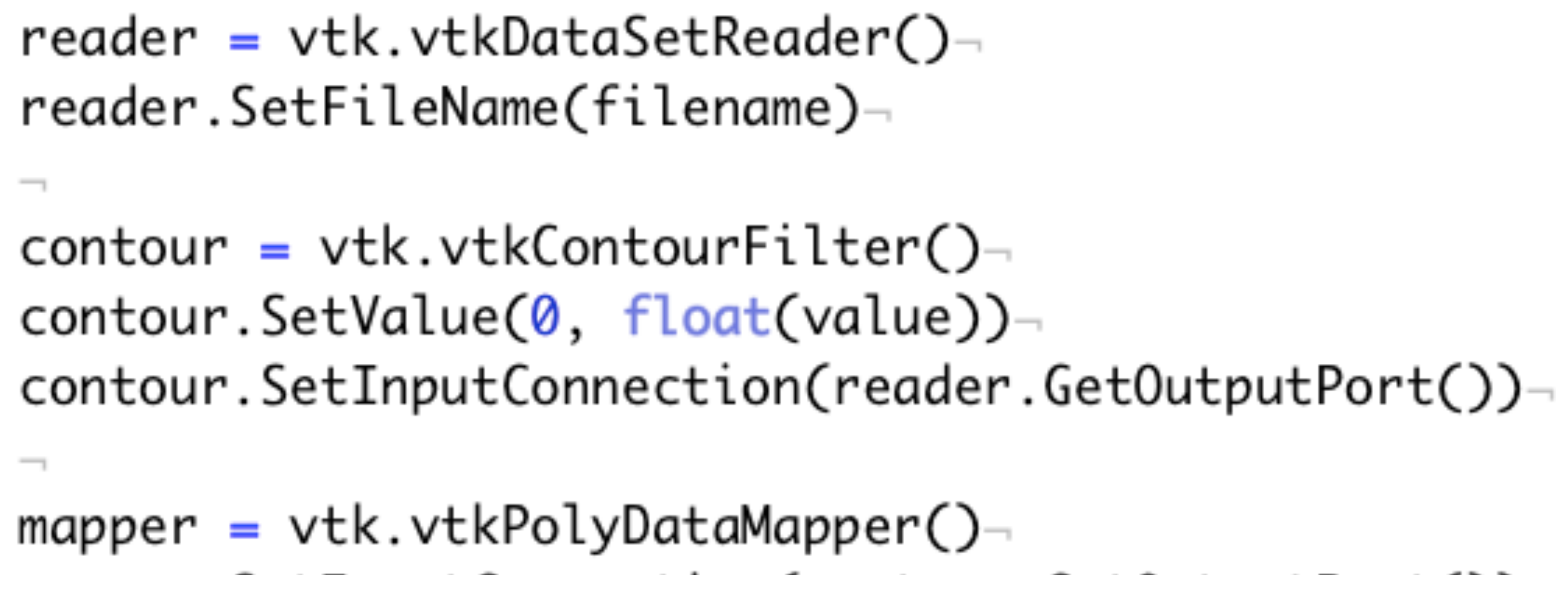
VTK pipeline connection syntax:

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

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

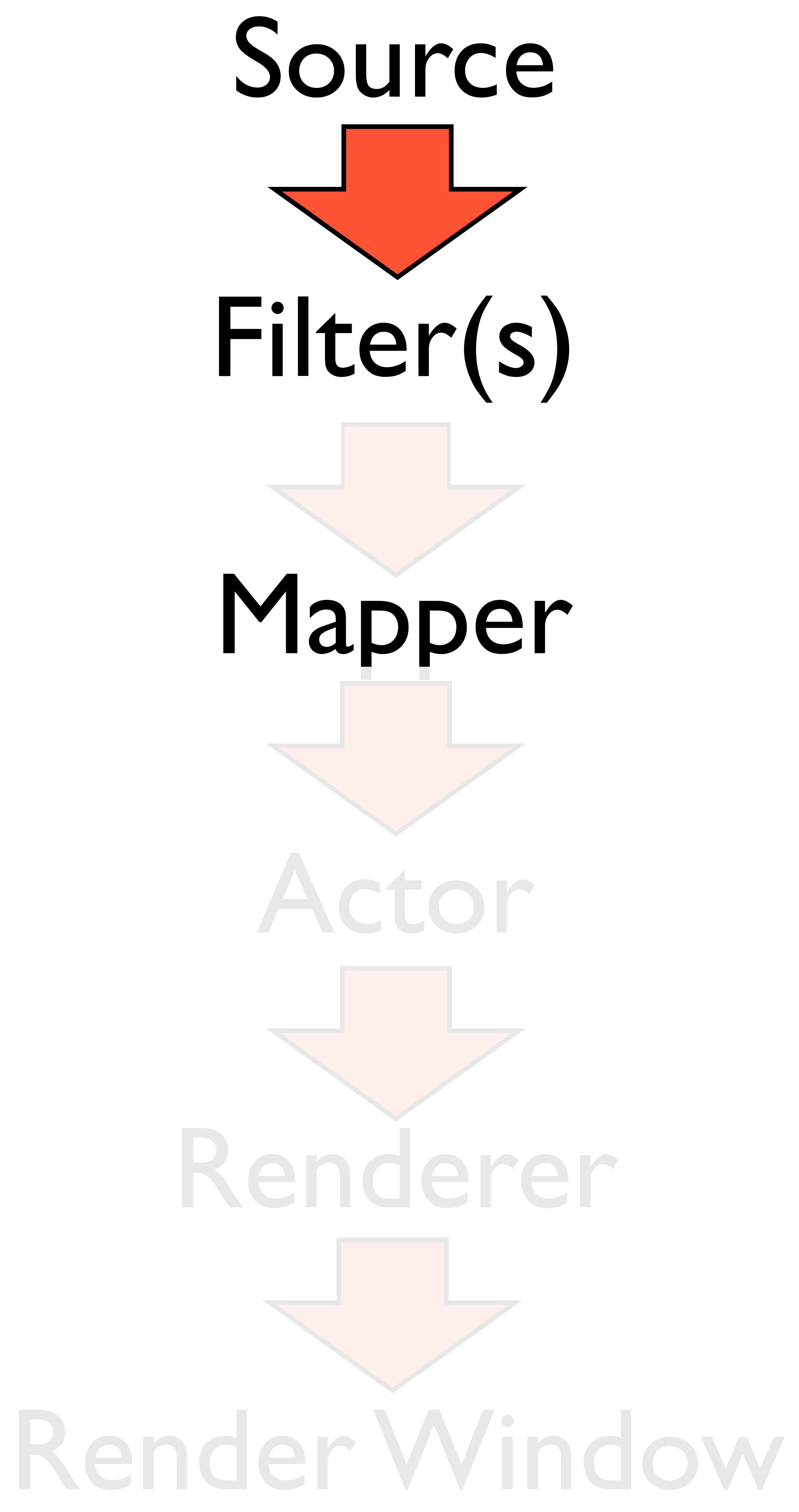
data pointers





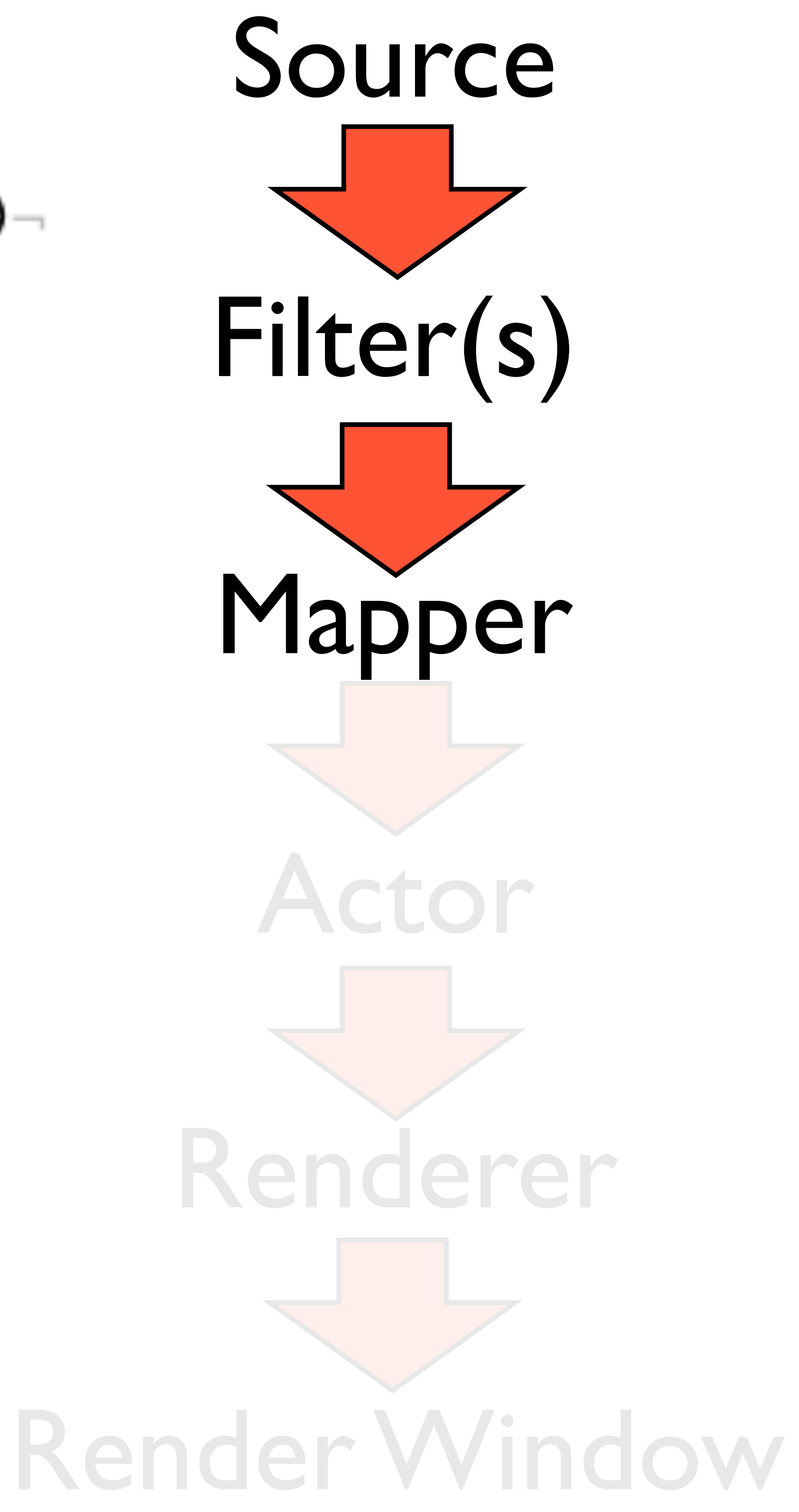
```
reader = vtk.vtkDataSetReader()
reader.SetFileName(filename)

contour = vtk.vtkContourFilter()
contour.SetValue(0, float(value))
contour.SetInputConnection(reader.GetOutputPort())

mapper = vtk.vtkPolyDataMapper()
```

Create a graphical mapper



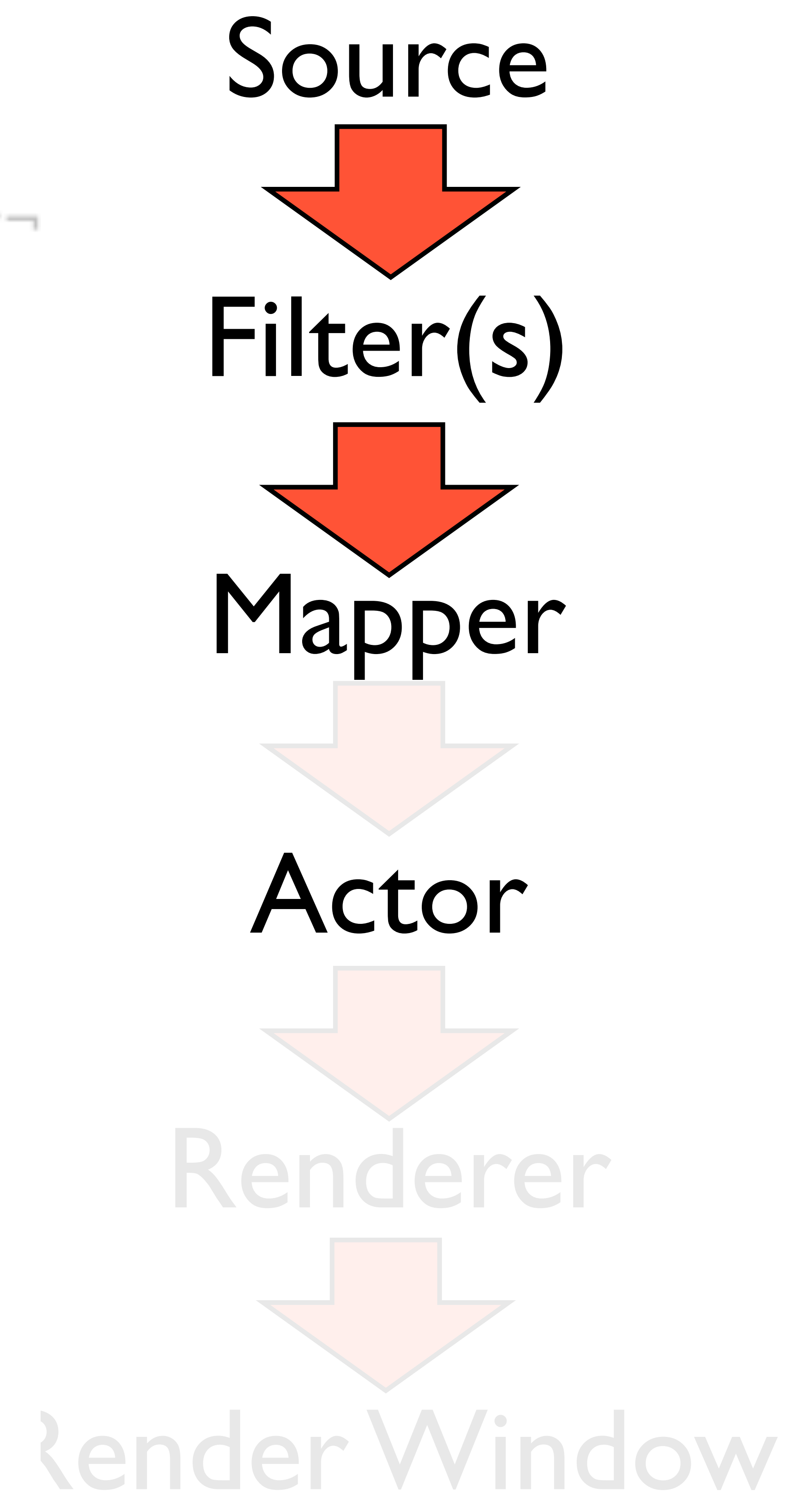
```python
reader = vtk.vtkDataSetReader()
reader.SetFileName(filename)

contour = vtk.vtkContourFilter()
contour.SetValue(0, float(value))
contour.SetInputConnection(reader.GetOutputPort())

mapper = vtk.vtkPolyDataMapper()
mapper.SetInputConnection(contour.GetOutputPort())
```

Apply it to geometry created by our filter



```
reader = vtk.vtkDataSetReader()
reader.SetFileName(filename)

contour = vtk.vtkContourFilter()
contour.SetValue(0, float(value))
contour.SetInputConnection(reader.GetOutputPort())

mapper = vtk.vtkPolyDataMapper()
mapper.SetInputConnection(contour.GetOutputPort())
mapper.ScalarVisibilityOff()

actor = vtk.vtkActor()
```

Create an actor (handle)



```
reader = vtk.vtkDataSetReader()
reader.SetFileName(filename)

contour = vtk.vtkContourFilter()
contour.SetValue(0, float(value))
contour.SetInputConnection(reader.GetOutputPort())

mapper = vtk.vtkPolyDataMapper()
mapper.SetInputConnection(contour.GetOutputPort())
mapper.ScalarVisibilityOff()

actor = vtk.vtkActor()
actor.SetMapper(mapper)
actor.GetProperty().SetColor(1, 1, 1)
```

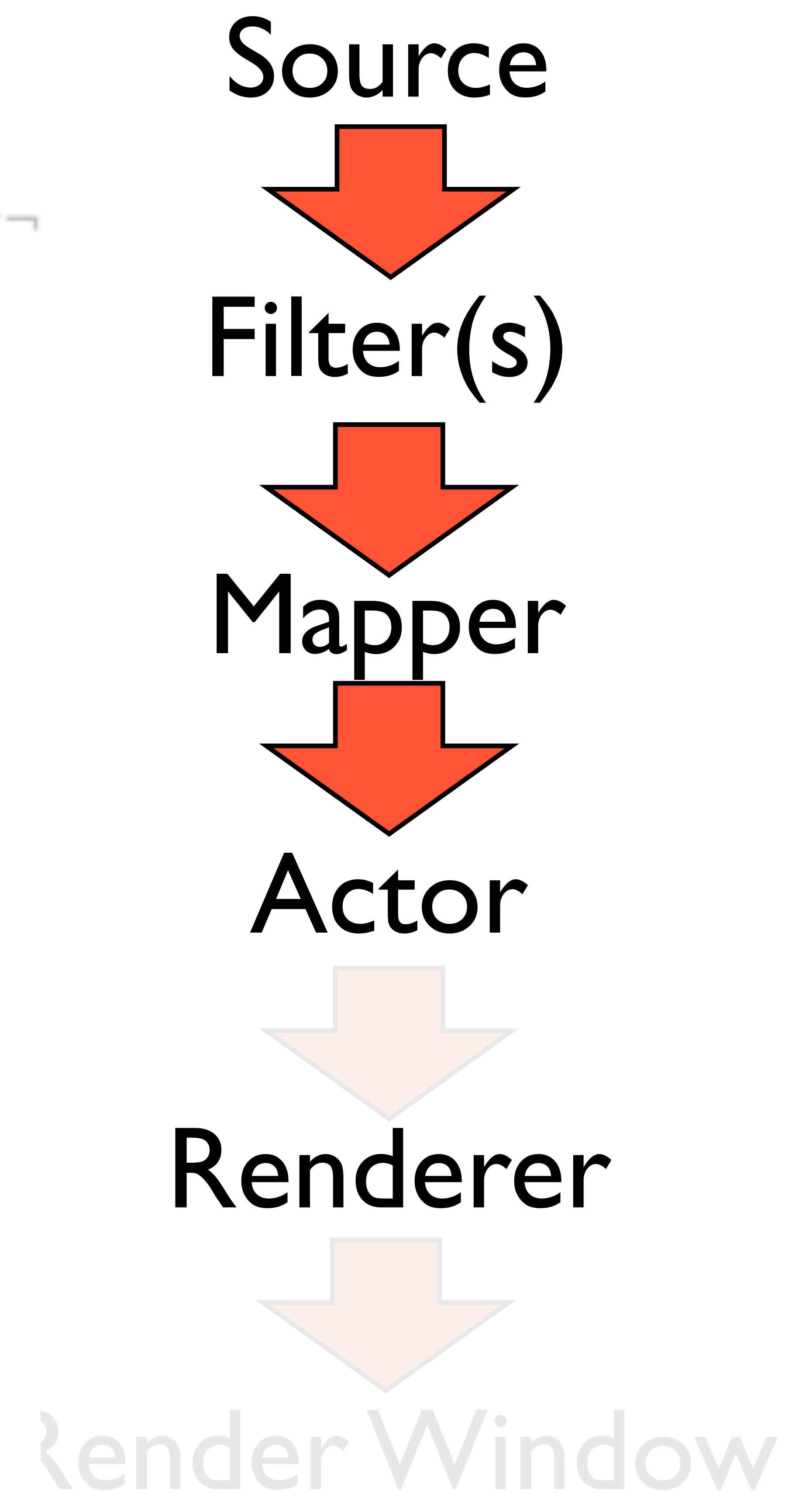
Attach it to our graphical primitives

Color them in white

$$\begin{tikzcd}
\text{Source} \arrow[d] \\
\text{Filter(s)} \arrow[d] \\
\text{Mapper} \arrow[d] \\
\text{Actor} \arrow[d] \\
\text{Renderer} \arrow[d] \\
\text{Render Window}
\end{tikzcd}$$

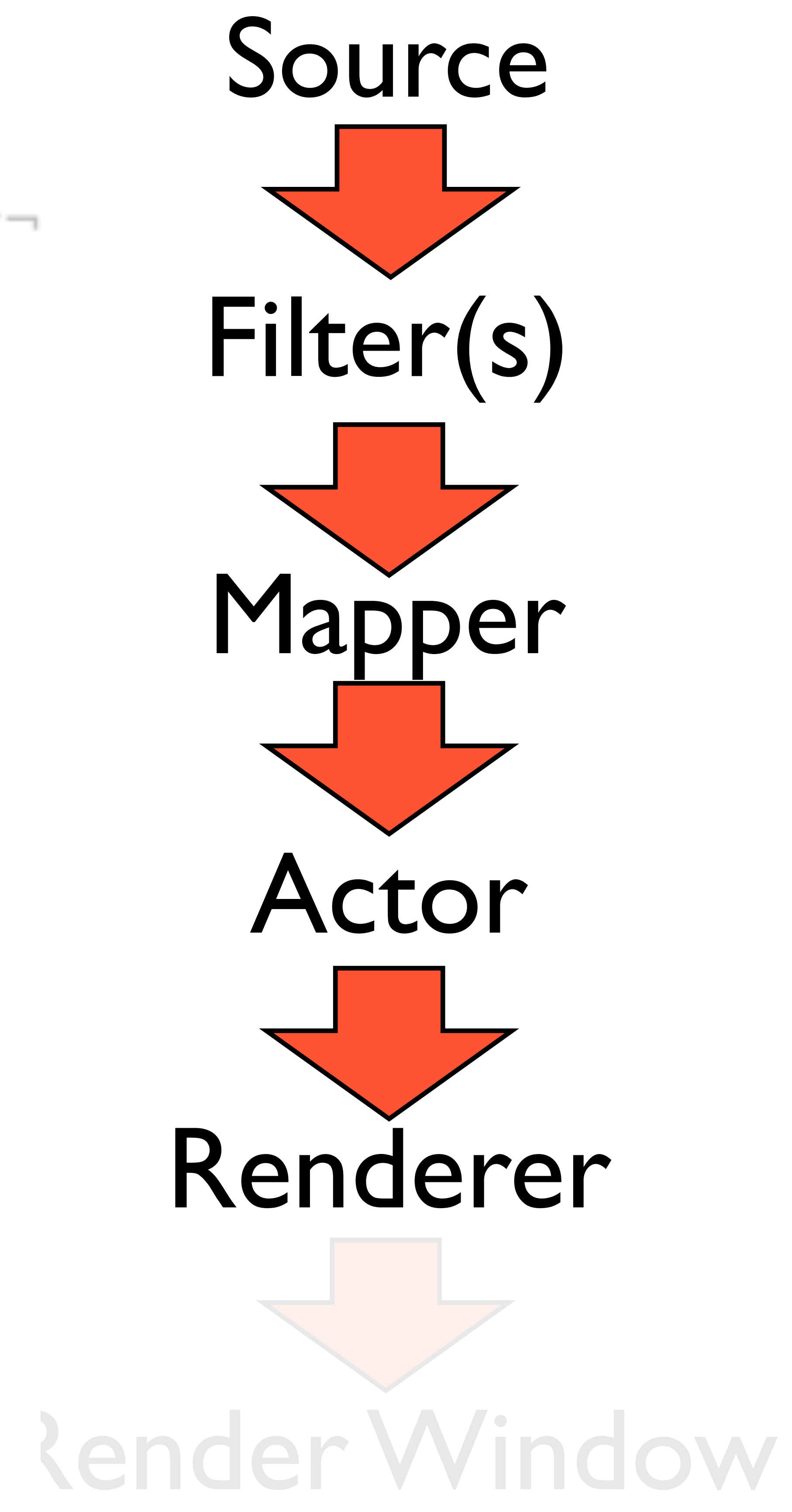
```
reader = vtk.vtkDataSetReader()
reader.SetFileName(filename)

contour = vtk.vtkContourFilter()
contour.SetValue(0, float(value))
contour.SetInputConnection(reader.GetOutputPort())

mapper = vtk.vtkPolyDataMapper()
mapper.SetInputConnection(contour.GetOutputPort())
mapper.ScalarVisibilityOff()

actor = vtk.vtkActor()
actor.SetMapper(mapper)
actor.GetProperty().SetColor(1, 1, 1)

renderer = vtk.vtkRenderer()
```

Create a scene renderer (camera, lights, ...)



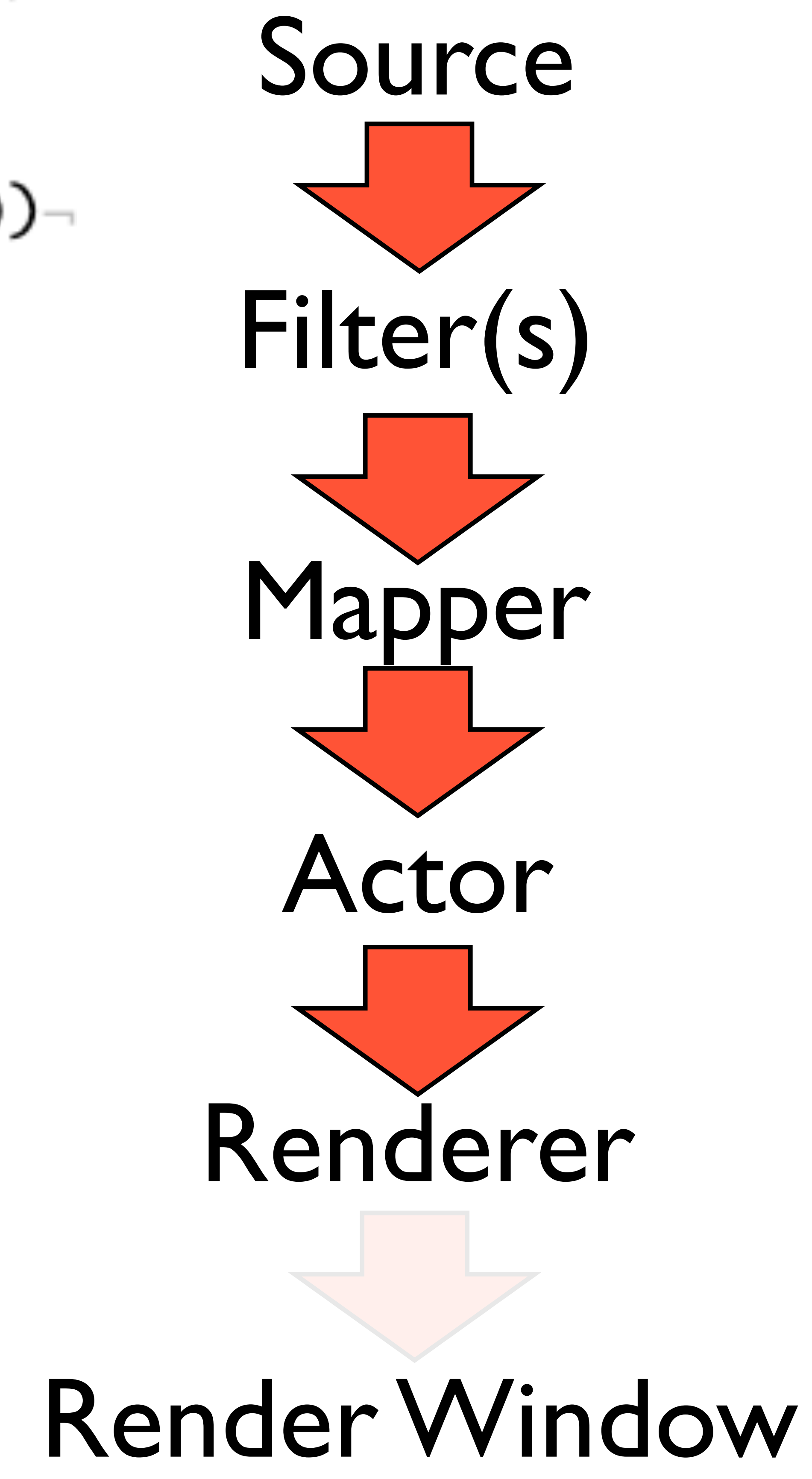
```python
reader = vtk.vtkDataSetReader()
reader.SetFileName(filename)

contour = vtk.vtkContourFilter()
contour.SetValue(0, float(value))
contour.SetInputConnection(reader.GetOutputPort())

mapper = vtk.vtkPolyDataMapper()
mapper.SetInputConnection(contour.GetOutputPort())
mapper.ScalarVisibilityOff()

actor = vtk.vtkActor()
actor.SetMapper(mapper)
actor.GetProperty().SetColor(1, 1, 1)

renderer = vtk.vtkRenderer()
renderer.AddActor(actor)
```

Insert our graphical objects in the scene



```
reader = vtk.vtkDataSetReader()
reader.SetFileName(filename)

contour = vtk.vtkContourFilter()
contour.SetValue(0, float(value))
contour.SetInputConnection(reader.GetOutputPort())

mapper = vtk.vtkPolyDataMapper()
mapper.SetInputConnection(contour.GetOutputPort())
mapper.ScalarVisibilityOff()

actor = vtk.vtkActor()
actor.SetMapper(mapper)
actor.GetProperty().SetColor(1, 1, 1)

renderer = vtk.vtkRenderer()
renderer.AddActor(actor)

window = vtk.vtkRenderWindow()
```

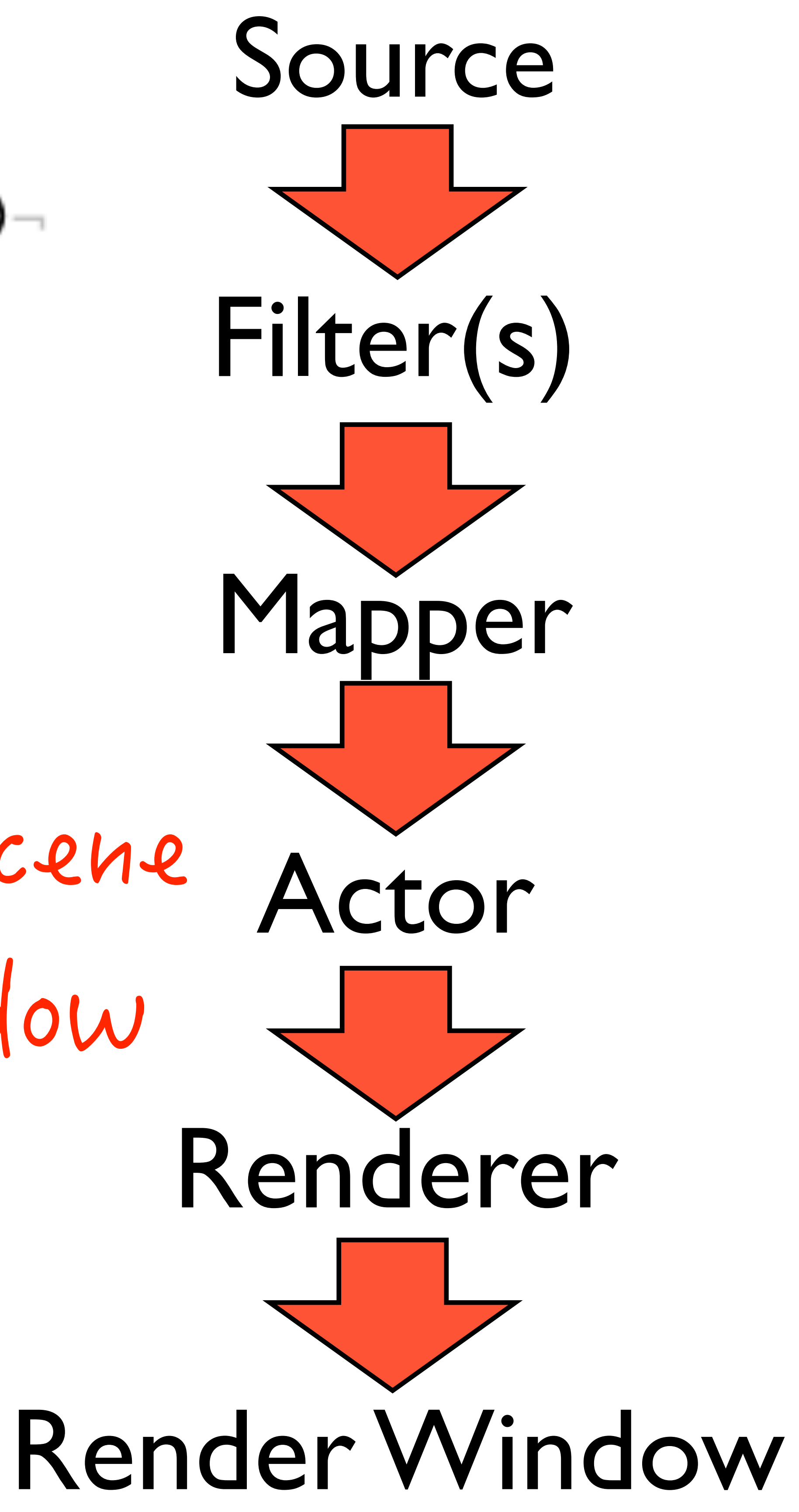
Create a window on the screen



```
reader = vtk.vtkDataSetReader()
reader.SetFileName(filename)

contour = vtk.vtkContourFilter()
contour.SetValue(0, float(value))
contour.SetInputConnection(reader.GetOutputPort())

mapper = vtk.vtkPolyDataMapper()
mapper.SetInputConnection(contour.GetOutputPort())
mapper.ScalarVisibilityOff()

actor = vtk.vtkActor()
actor.SetMapper(mapper)
actor.GetProperty().SetColor(1, 1, 1)

renderer = vtk.vtkRenderer()
renderer.AddActor(actor)

window = vtk.vtkRenderWindow()
window.AddRenderer(renderer)
window.SetSize(600, 600)
```

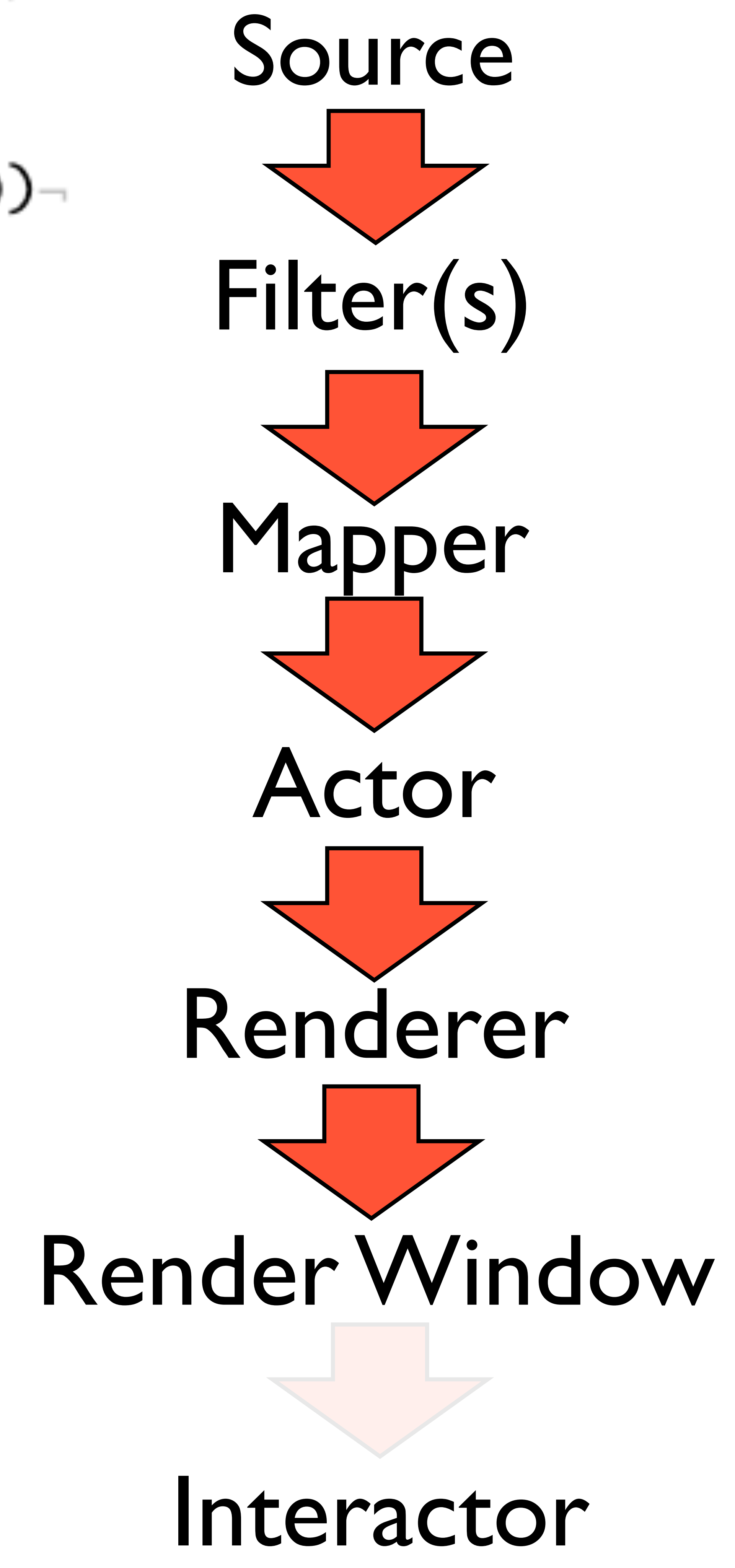
Render our scene in that window

Set window/picture resolution



```
reader = vtk.vtkDataSetReader()
reader.SetFileName(filename)

contour = vtk.vtkContourFilter()
contour.SetValue(0, float(value))
contour.SetInputConnection(reader.GetOutputPort())

mapper = vtk.vtkPolyDataMapper()
mapper.SetInputConnection(contour.GetOutputPort())
mapper.ScalarVisibilityOff()

actor = vtk.vtkActor()
actor.SetMapper(mapper)
actor.GetProperty().SetColor(1, 1, 1)

renderer = vtk.vtkRenderer()
renderer.AddActor(actor)

window = vtk.vtkRenderWindow()
window.AddRenderer(renderer)
window.SetSize(600, 600)

interactor = vtk.vtkRenderWindowInteractor()
```

Create support for mouse interaction



```
reader = vtk.vtkDataSetReader()
reader.SetFileName(filename)

contour = vtk.vtkContourFilter()
contour.SetValue(0, float(value))
contour.SetInputConnection(reader.GetOutputPort())

mapper = vtk.vtkPolyDataMapper()
mapper.SetInputConnection(contour.GetOutputPort())
mapper.ScalarVisibilityOff()

actor = vtk.vtkActor()
actor.SetMapper(mapper)
actor.GetProperty().SetColor(1, 1, 1)

renderer = vtk.vtkRenderer()
renderer.AddActor(actor)

window = vtk.vtkRenderWindow()
window.AddRenderer(renderer)
window.SetSize(600, 600)

interactor = vtk.vtkRenderWindowInteractor()
interactor.SetRenderWindow(window)
interactor.Initialize()
```

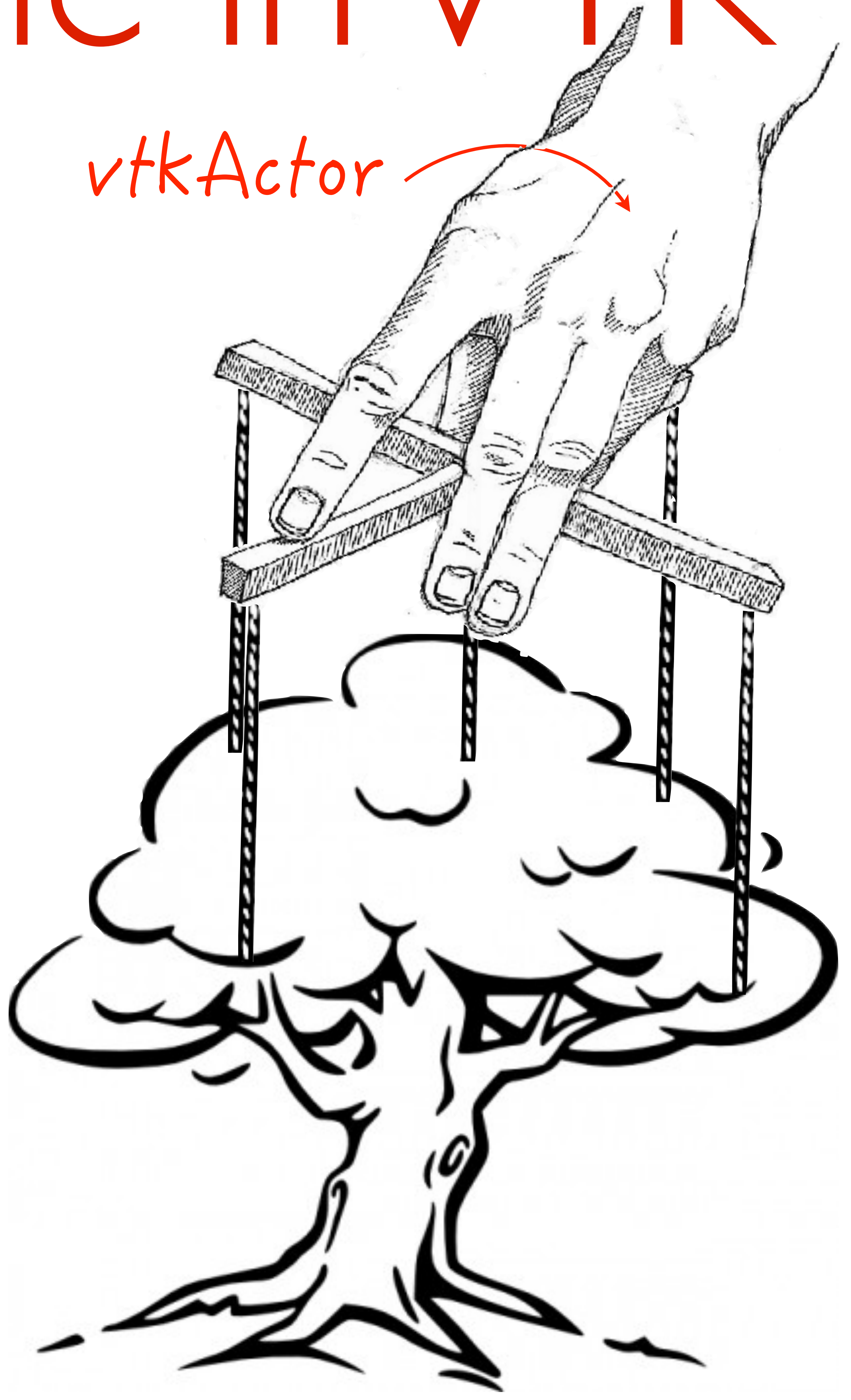
Add it to our rendering window

$$\begin{tikzcd}
\text{Source} \arrow[d] \\
\text{Filter(s)} \arrow[d] \\
\text{Mapper} \arrow[d] \\
\text{Actor} \arrow[d] \\
\text{Renderer} \arrow[d] \\
\text{Render Window} \arrow[d] \\
\text{Interactor}
\end{tikzcd}$$

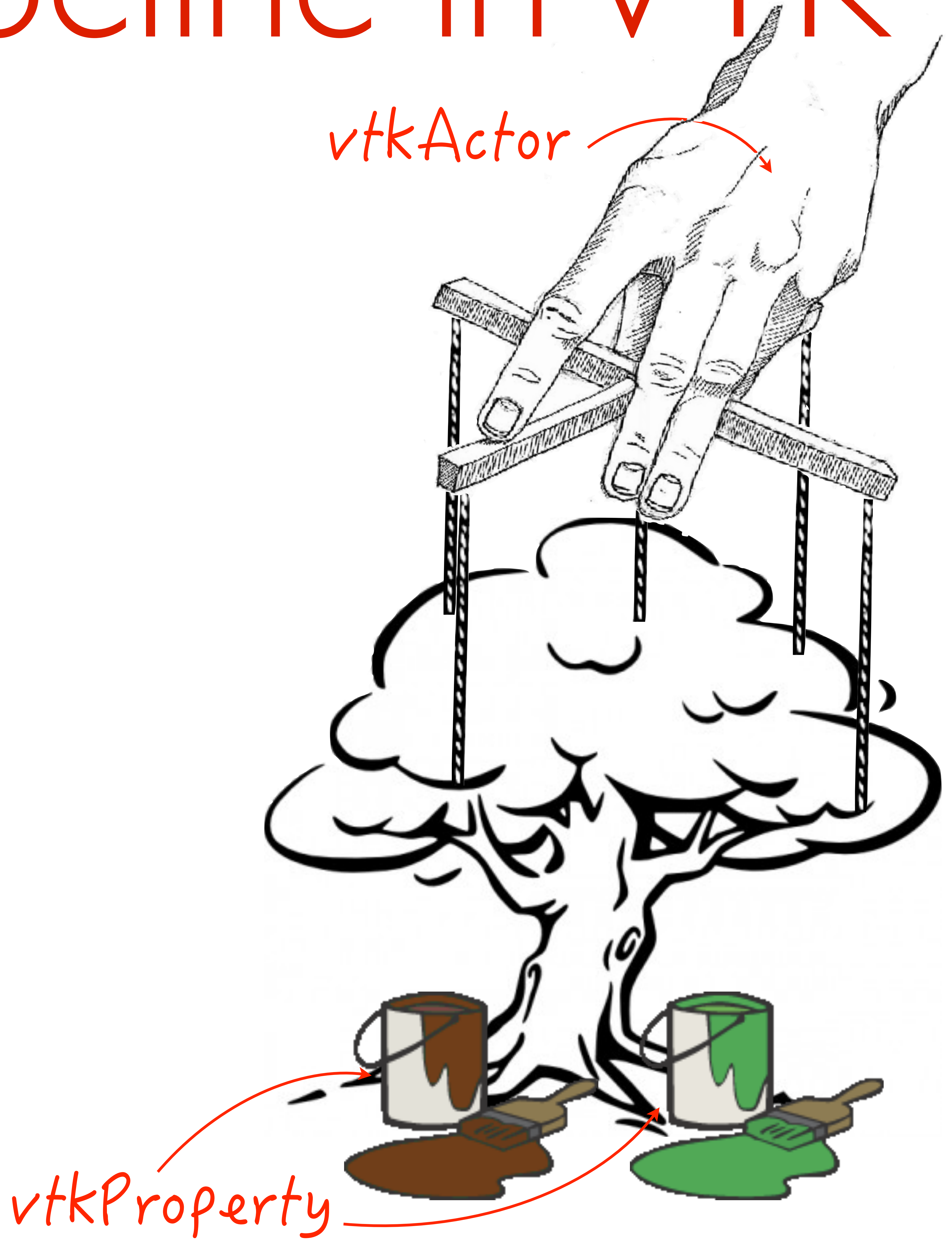
Rendering Pipeline in VTK

Rendering Pipeline in VTK

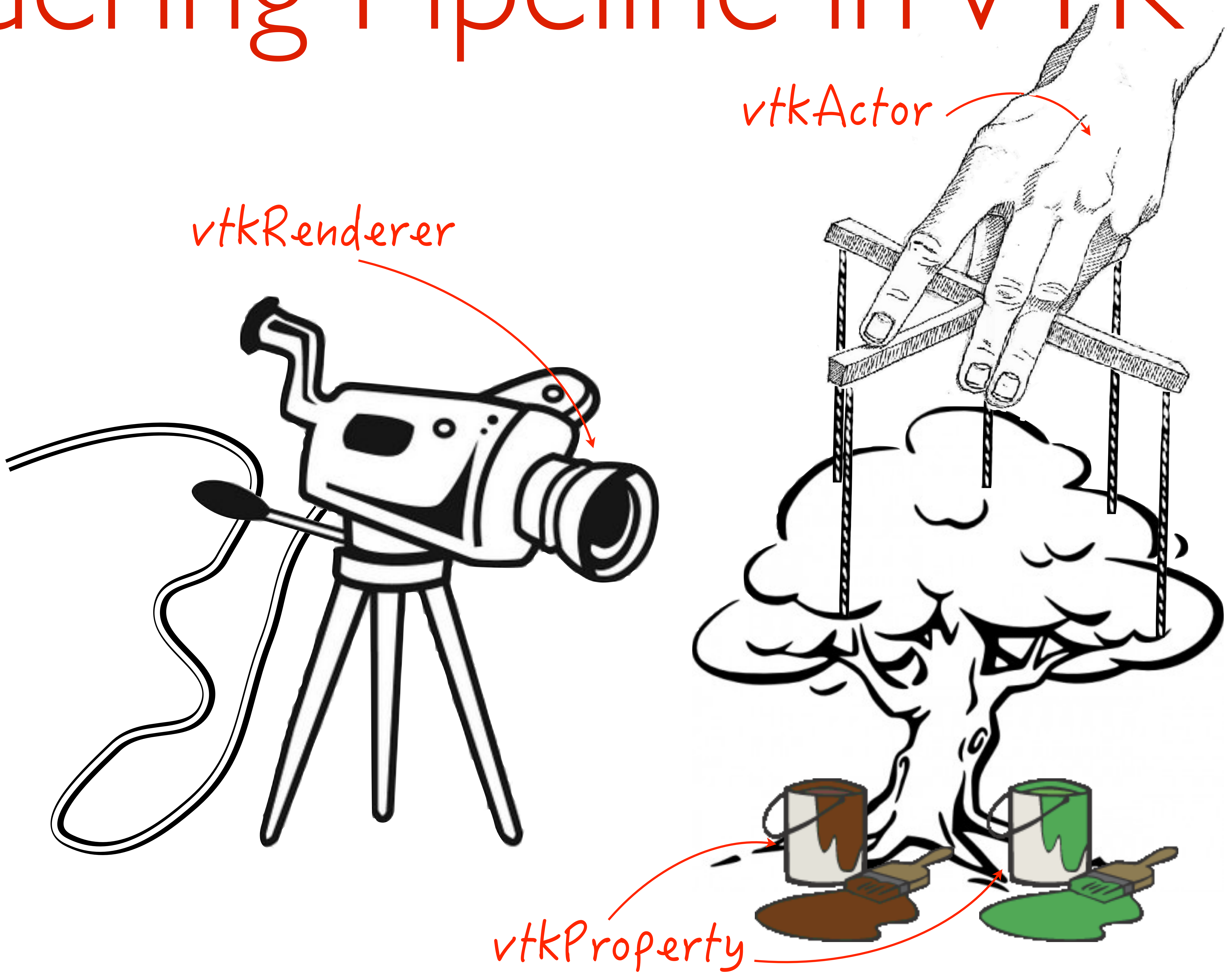
Rendering Pipeline in VTK



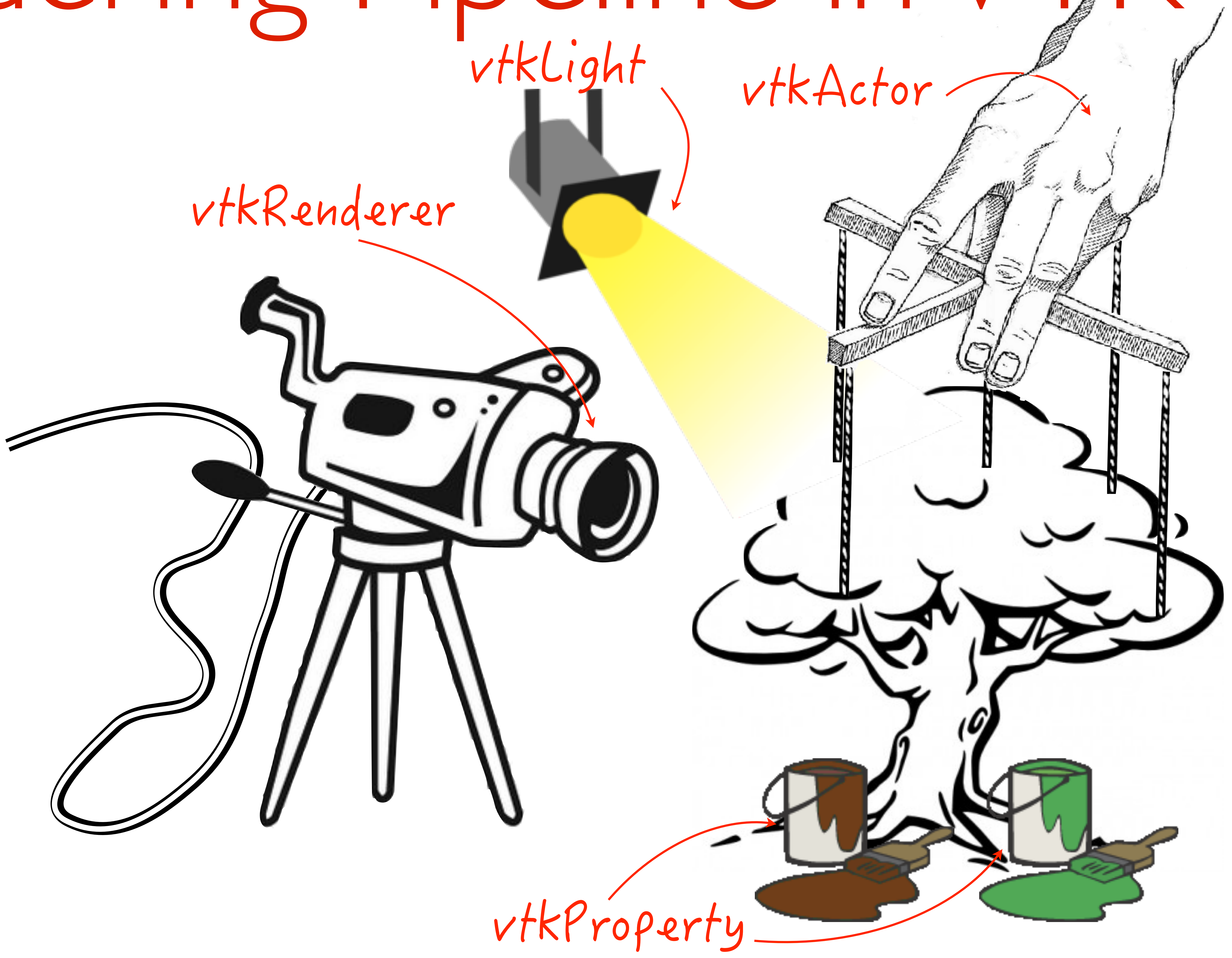
Rendering Pipeline in VTK



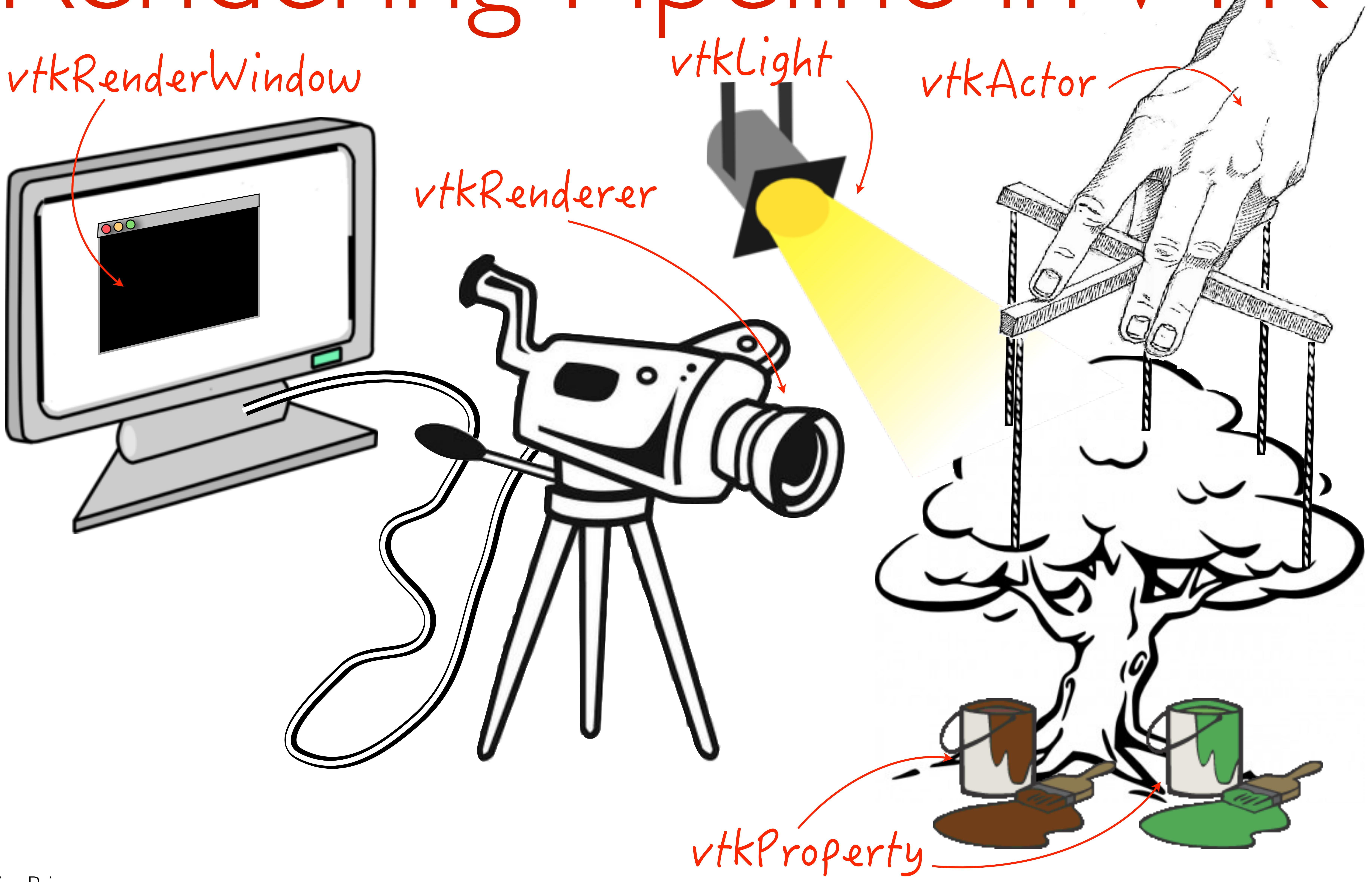
Rendering Pipeline in VTK



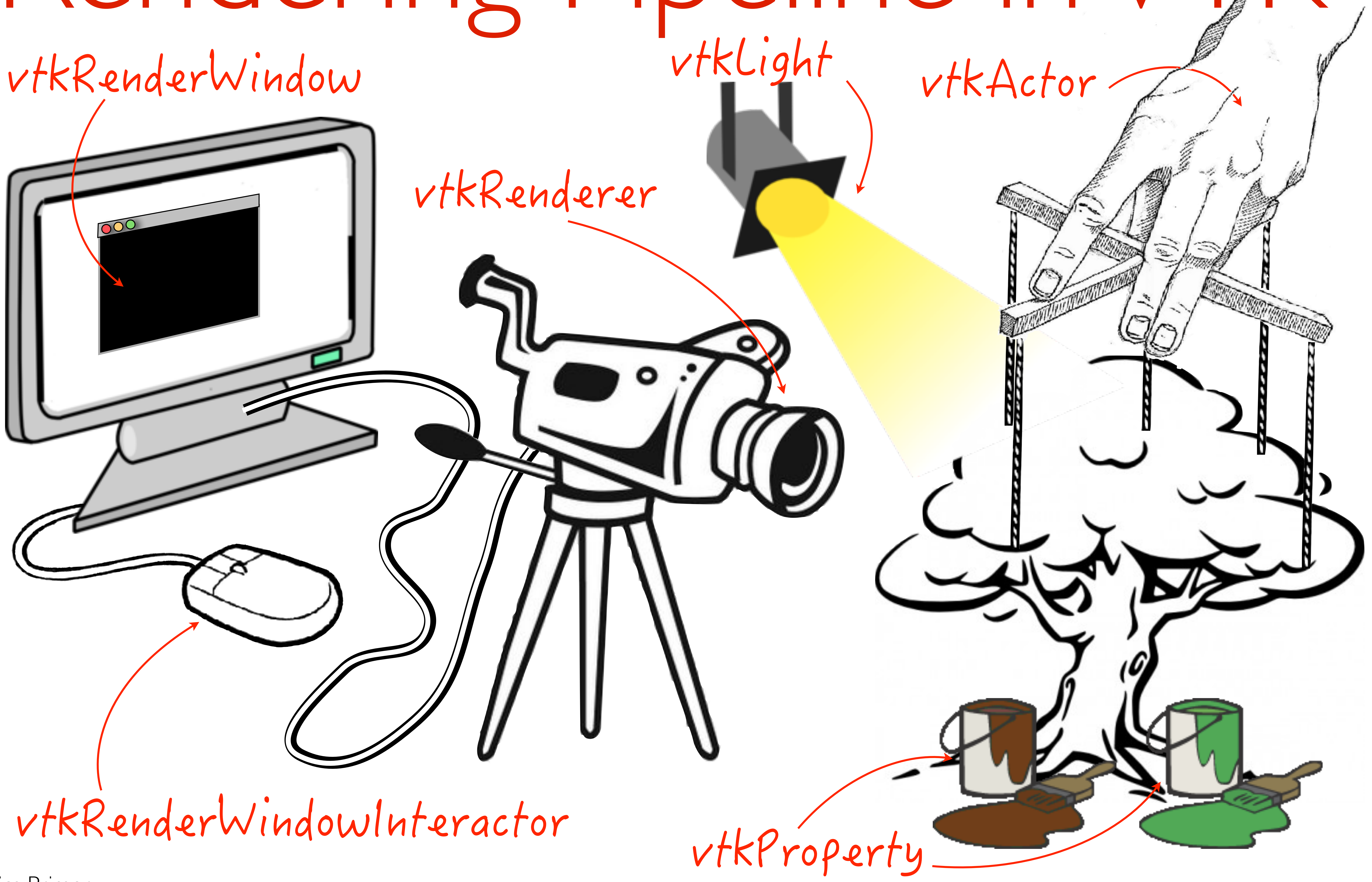
Rendering Pipeline in VTK



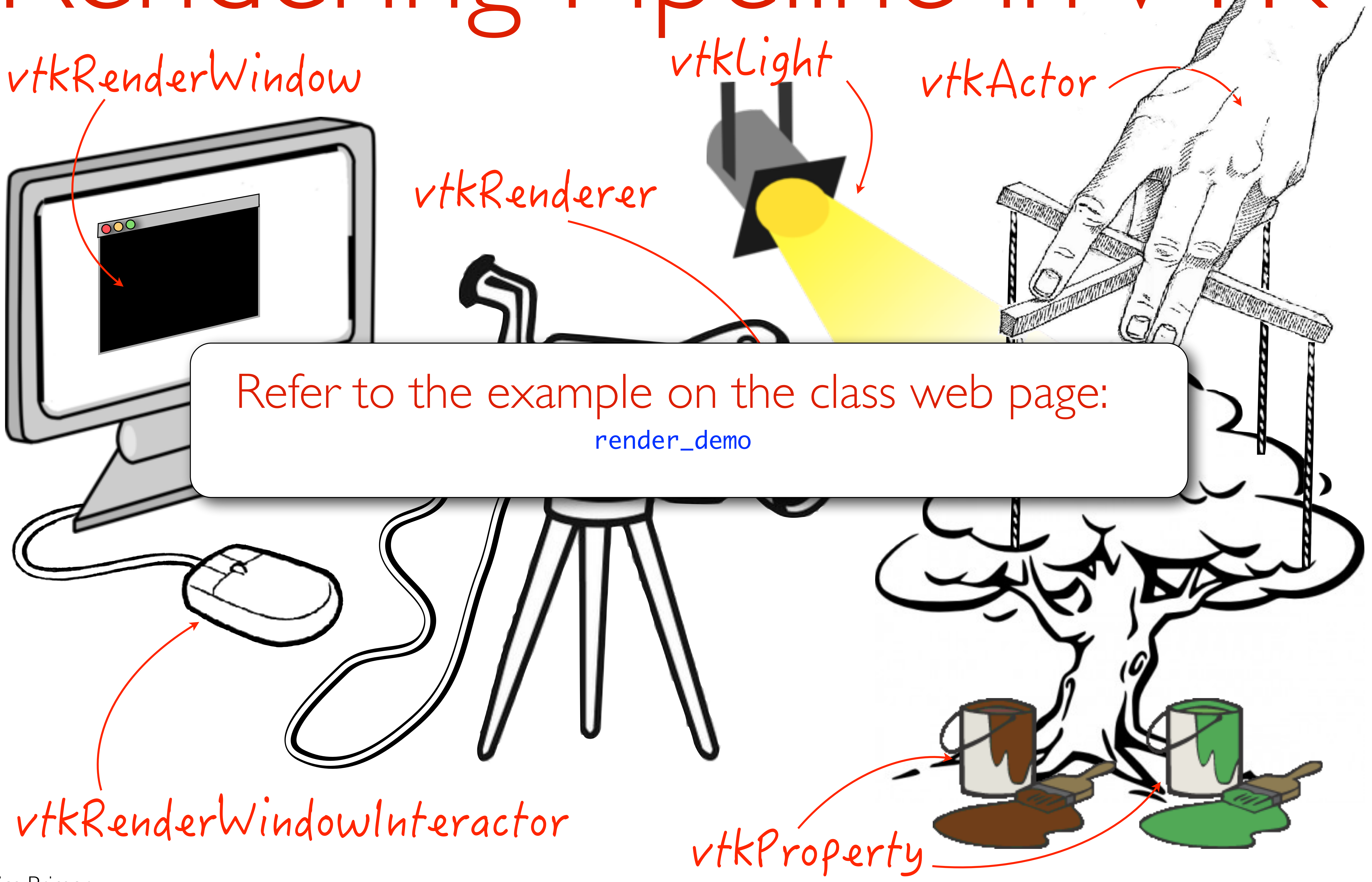
Rendering Pipeline in VTK



Rendering Pipeline in VTK



Rendering Pipeline in VTK



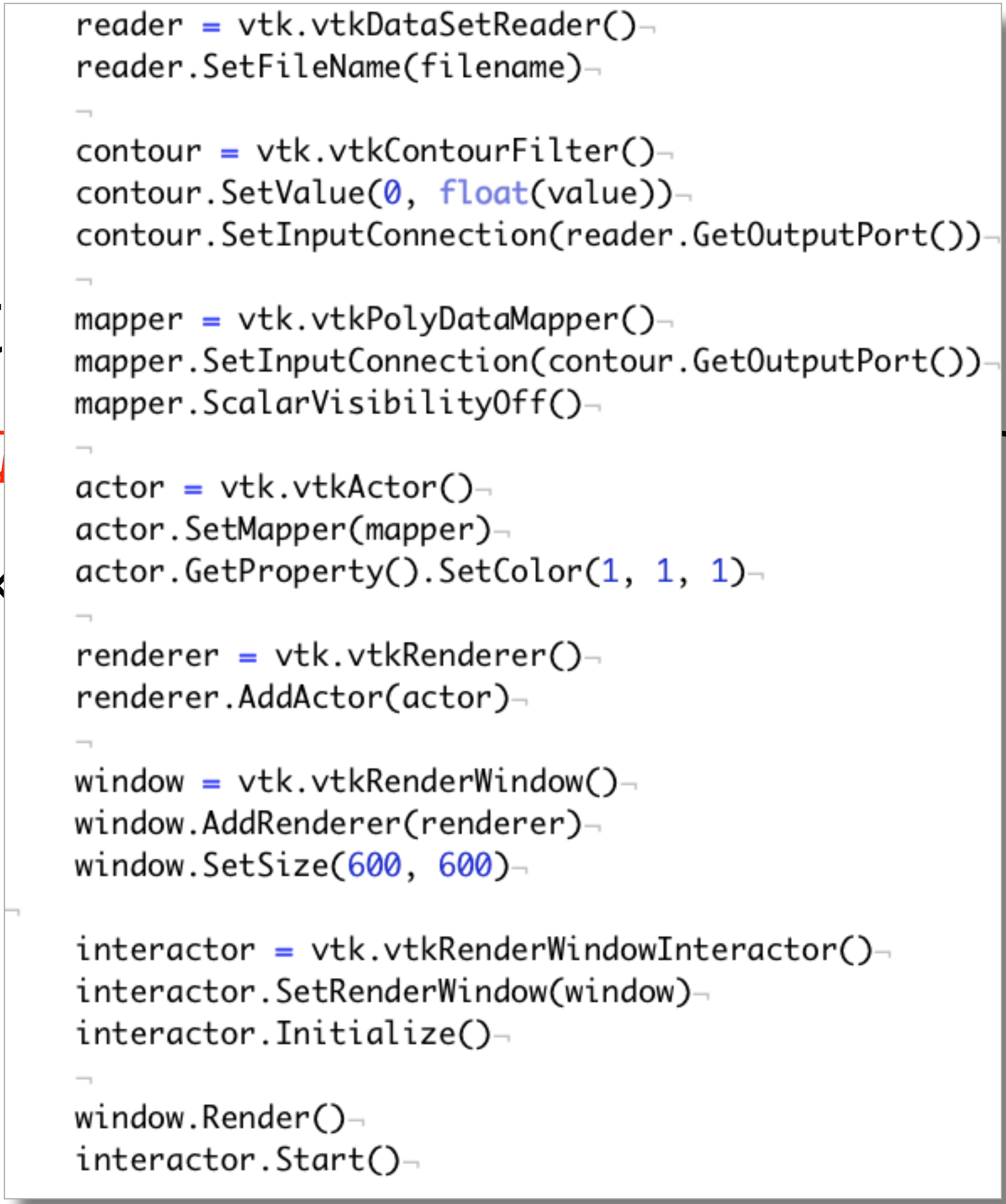
Refer to the example on the class web page:
`render_demo`

Visualization Pipeline

- Implicit control of execution (*lazy evaluation*): algorithms are only (re)executed when needed

Vi ne

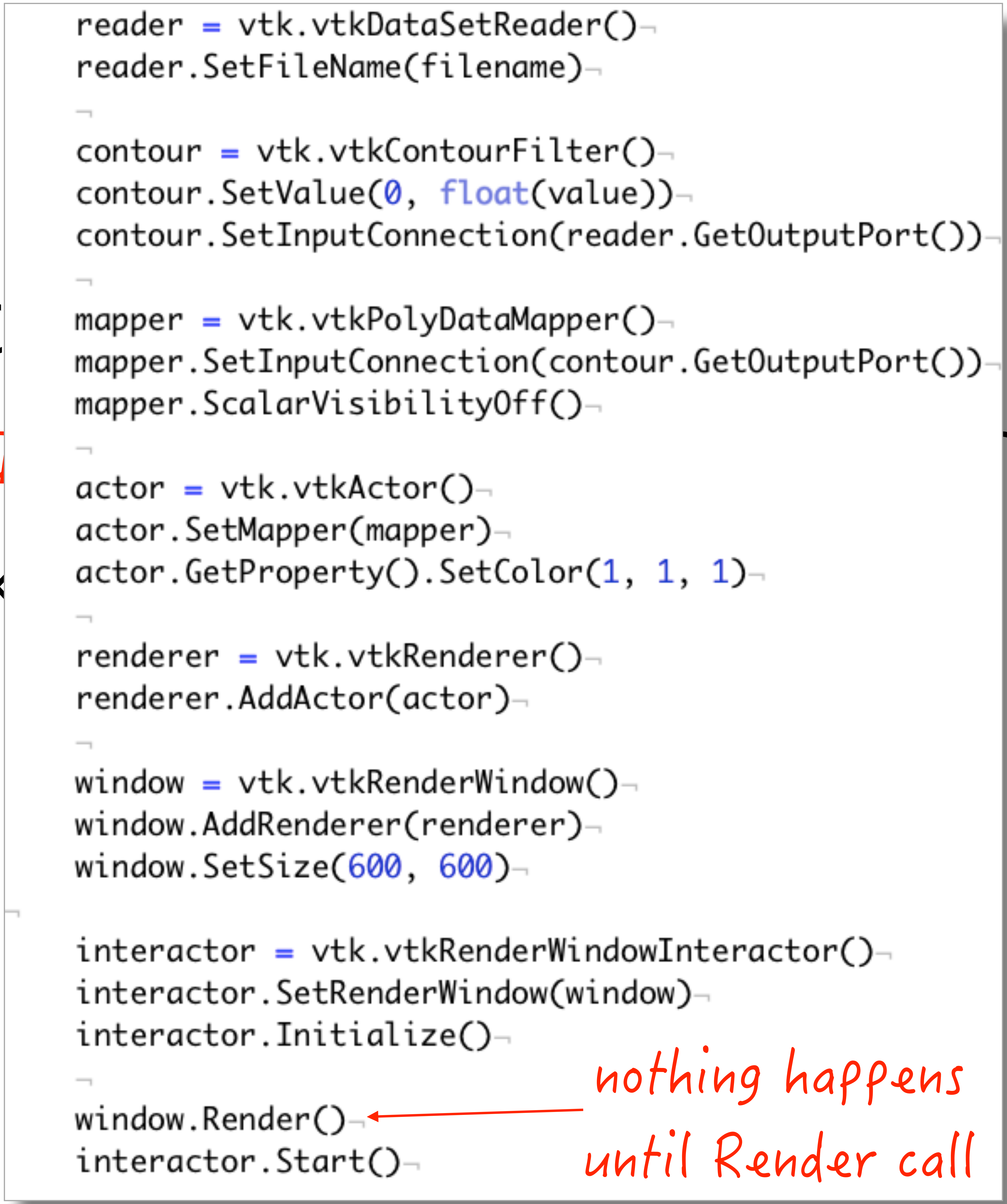
- Implic (*lazy* *evalua* nly (re)ex



```
reader = vtk.vtkDataSetReader()
reader.SetFileName(filename)

contour = vtk.vtkContourFilter()
contour.SetValue(0, float(value))
contour.SetInputConnection(reader.GetOutputPort())

mapper = vtk.vtkPolyDataMapper()
mapper.SetInputConnection(contour.GetOutputPort())
mapper.ScalarVisibilityOff()

actor = vtk.vtkActor()
actor.SetMapper(mapper)
actor.GetProperty().SetColor(1, 1, 1)

renderer = vtk.vtkRenderer()
renderer.AddActor(actor)

window = vtk.vtkRenderWindow()
window.AddRenderer(renderer)
window.SetSize(600, 600)

interactor = vtk.vtkRenderWindowInteractor()
interactor.SetRenderWindow(window)
interactor.Initialize()

window.Render()
interactor.Start()
```

Vi ne

- Implic (*lazy*
  evalua nly
  (re)ex

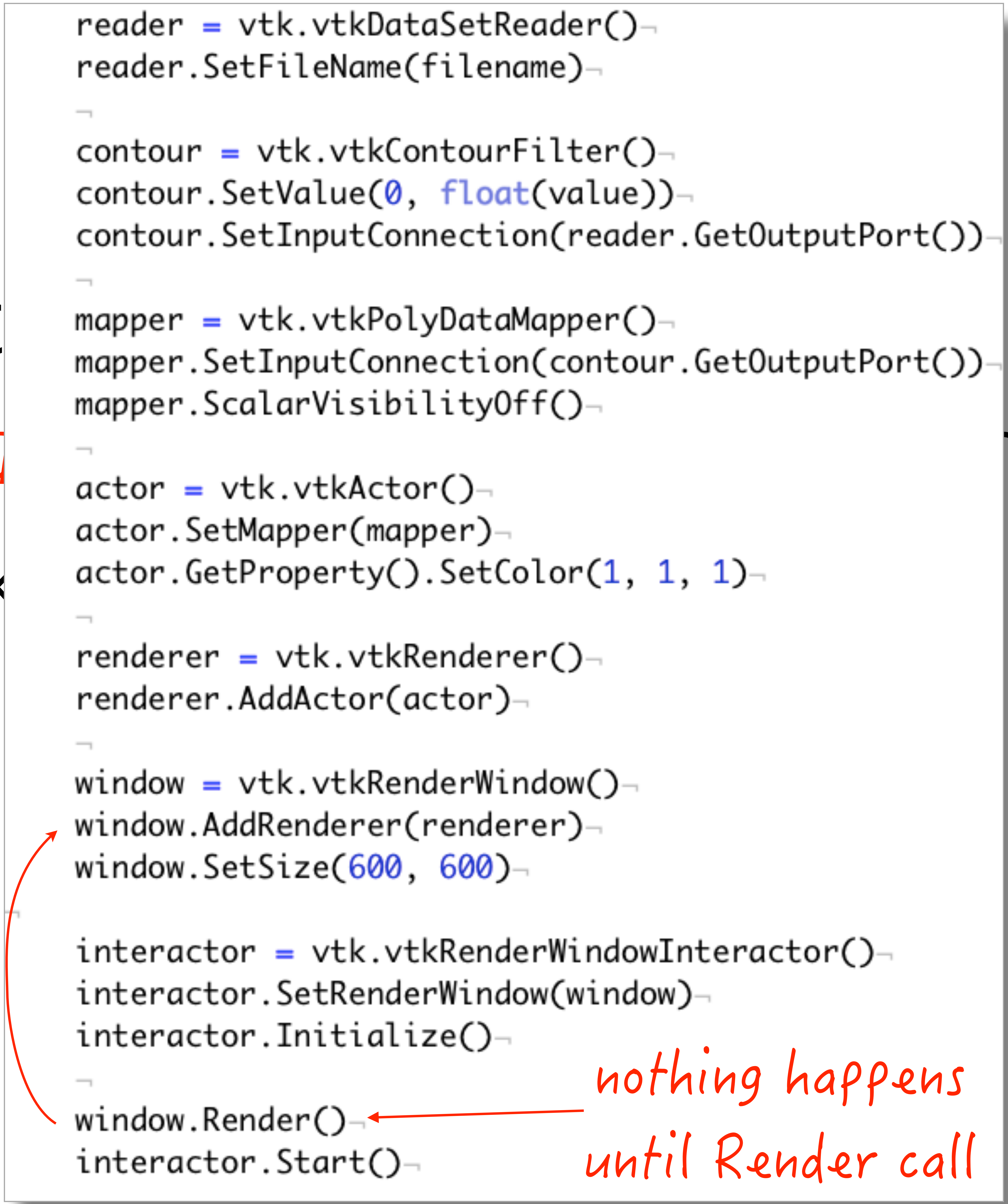


```
reader = vtk.vtkDataSetReader()
reader.SetFileName(filename)

contour = vtk.vtkContourFilter()
contour.SetValue(0, float(value))
contour.SetInputConnection(reader.GetOutputPort())

mapper = vtk.vtkPolyDataMapper()
mapper.SetInputConnection(contour.GetOutputPort())
mapper.ScalarVisibilityOff()

actor = vtk.vtkActor()
actor.SetMapper(mapper)
actor.GetProperty().SetColor(1, 1, 1)

renderer = vtk.vtkRenderer()
renderer.AddActor(actor)

window = vtk.vtkRenderWindow()
window.AddRenderer(renderer)
window.SetSize(600, 600)

interactor = vtk.vtkRenderWindowInteractor()
interactor.SetRenderWindow(window)
interactor.Initialize()

window.Render()
interactor.Start()
```

nothing happens until Render call

Vi ne

- Implic (*lazy evalua* nly (re)ex

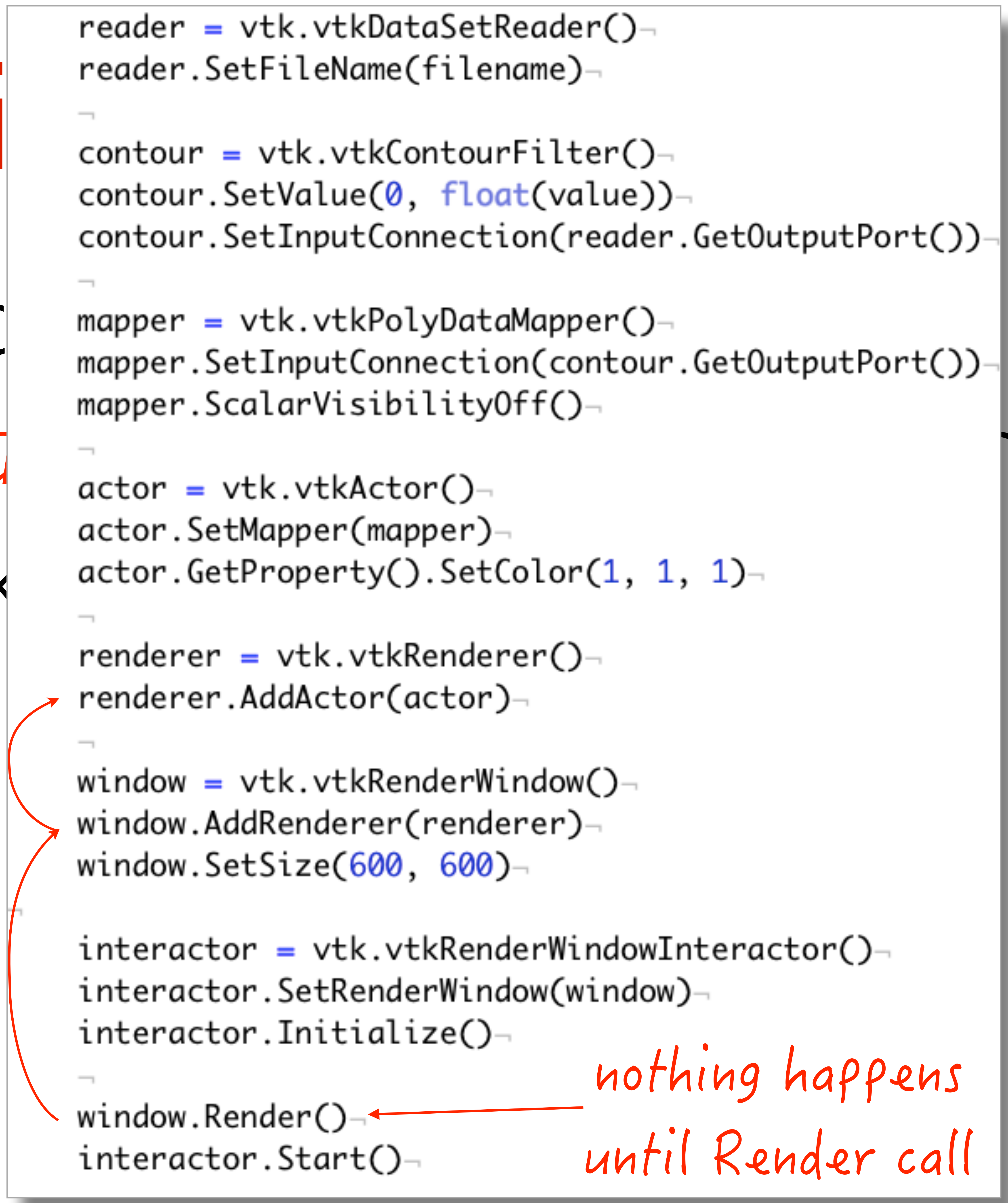


```
reader = vtk.vtkDataSetReader()
reader.SetFileName(filename)

contour = vtk.vtkContourFilter()
contour.SetValue(0, float(value))
contour.SetInputConnection(reader.GetOutputPort())

mapper = vtk.vtkPolyDataMapper()
mapper.SetInputConnection(contour.GetOutputPort())
mapper.ScalarVisibilityOff()

actor = vtk.vtkActor()
actor.SetMapper(mapper)
actor.GetProperty().SetColor(1, 1, 1)

renderer = vtk.vtkRenderer()
renderer.AddActor(actor)

window = vtk.vtkRenderWindow()
window.AddRenderer(renderer)
window.SetSize(600, 600)

interactor = vtk.vtkRenderWindowInteractor()
interactor.SetRenderWindow(window)
interactor.Initialize()

window.Render()
interactor.Start()
```

nothing happens until Render call

Vi ne

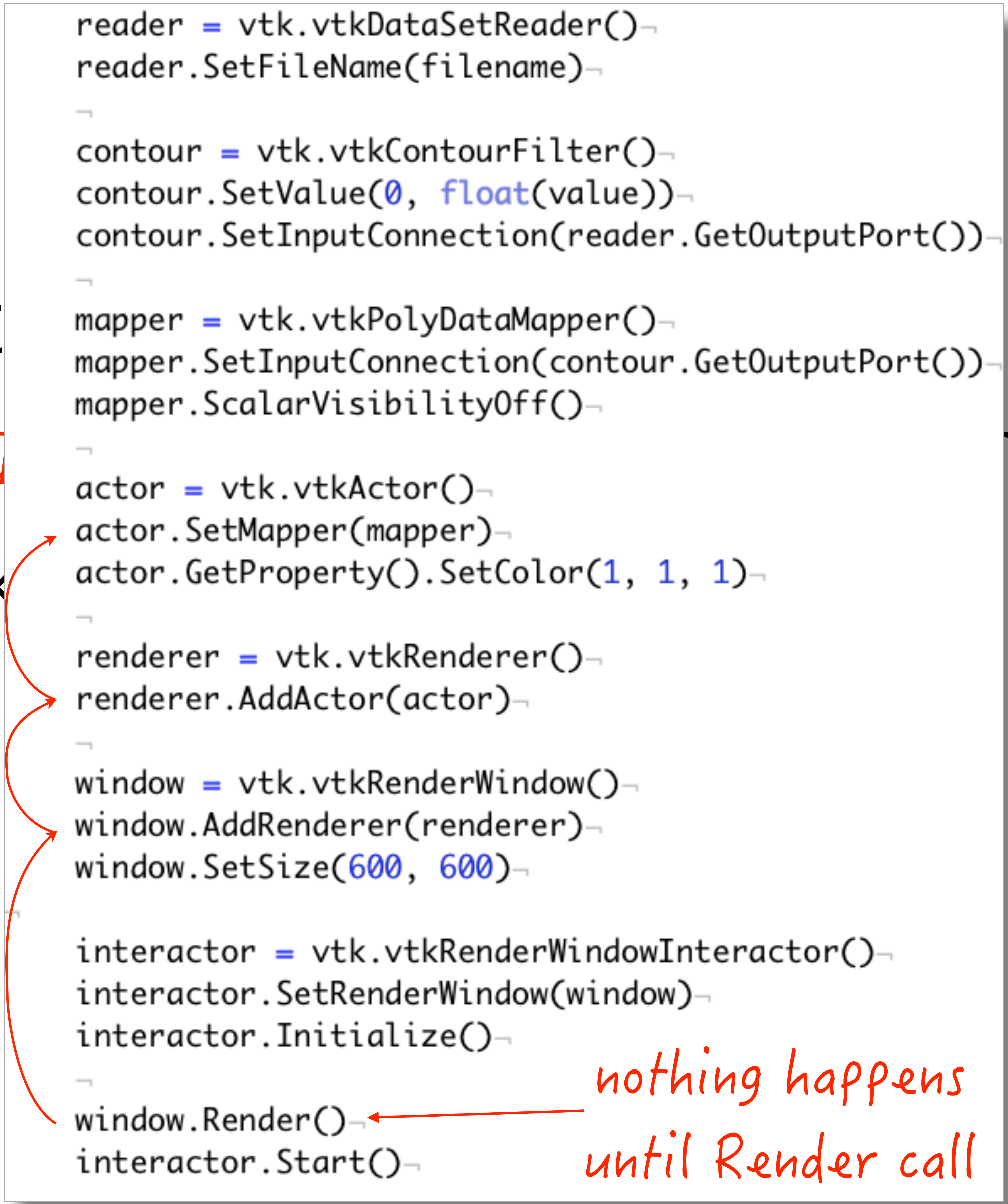
- Implic (*lazy* *evalua* nly (re)ex



```
reader = vtk.vtkDataSetReader()
reader.SetFileName(filename)

contour = vtk.vtkContourFilter()
contour.SetValue(0, float(value))
contour.SetInputConnection(reader.GetOutputPort())

mapper = vtk.vtkPolyDataMapper()
mapper.SetInputConnection(contour.GetOutputPort())
mapper.ScalarVisibilityOff()

actor = vtk.vtkActor()
actor.SetMapper(mapper)
actor.GetProperty().SetColor(1, 1, 1)

renderer = vtk.vtkRenderer()
renderer.AddActor(actor)

window = vtk.vtkRenderWindow()
window.AddRenderer(renderer)
window.SetSize(600, 600)

interactor = vtk.vtkRenderWindowInteractor()
interactor.SetRenderWindow(window)
interactor.Initialize()

window.Render()
interactor.Start()
```

Vi ne

- Implic (*lazy*
  evalua nly
  (re)ex

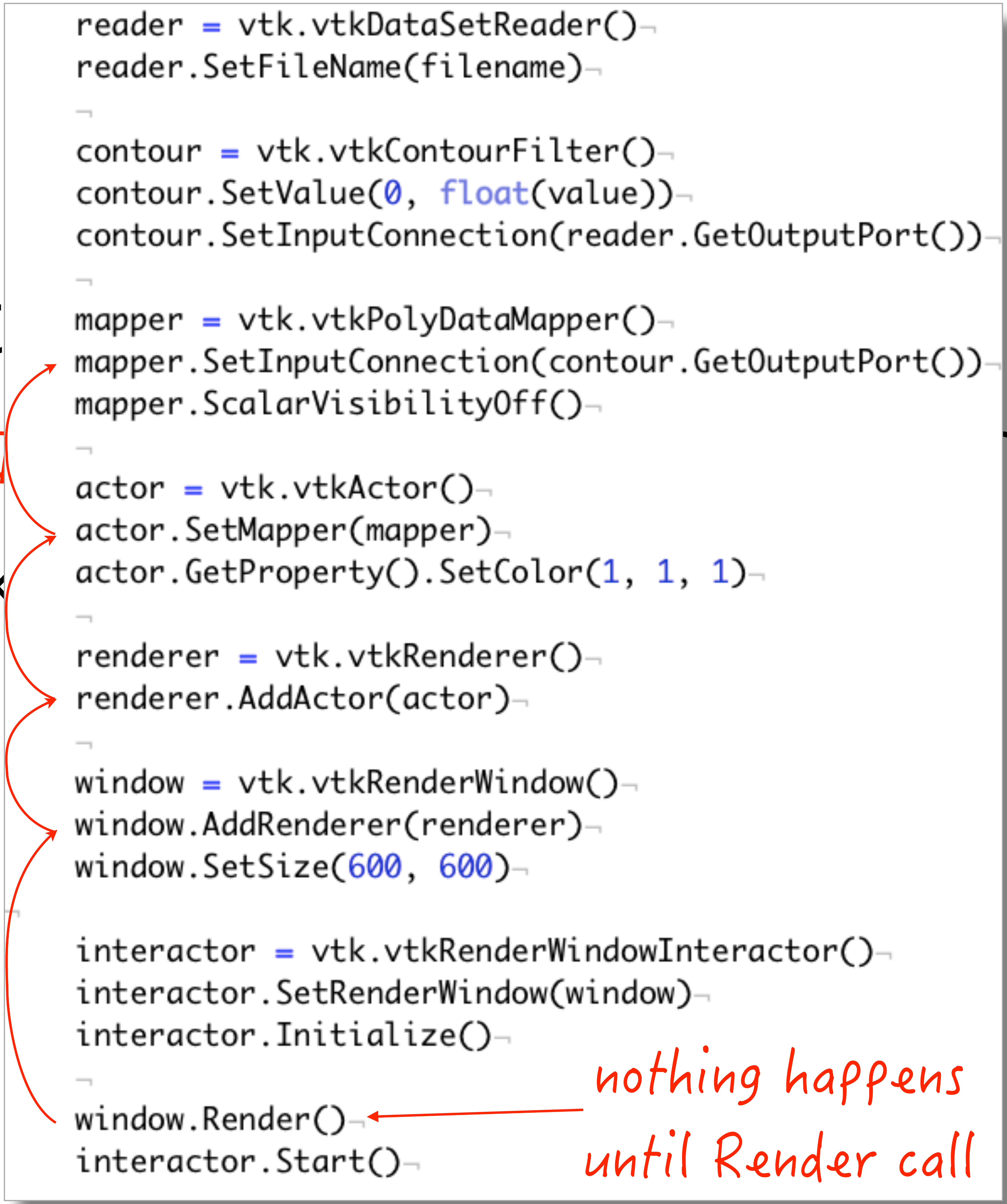


```
reader = vtk.vtkDataSetReader()
reader.SetFileName(filename)

contour = vtk.vtkContourFilter()
contour.SetValue(0, float(value))
contour.SetInputConnection(reader.GetOutputPort())

mapper = vtk.vtkPolyDataMapper()
mapper.SetInputConnection(contour.GetOutputPort())
mapper.ScalarVisibilityOff()

actor = vtk.vtkActor()
actor.SetMapper(mapper)
actor.GetProperty().SetColor(1, 1, 1)

renderer = vtk.vtkRenderer()
renderer.AddActor(actor)

window = vtk.vtkRenderWindow()
window.AddRenderer(renderer)
window.SetSize(600, 600)

interactor = vtk.vtkRenderWindowInteractor()
interactor.SetRenderWindow(window)
interactor.Initialize()

window.Render()
interactor.Start()
```

nothing happens until Render call

Vi ne

- Implic (*lazy*
  evalua nly
  (re)ex

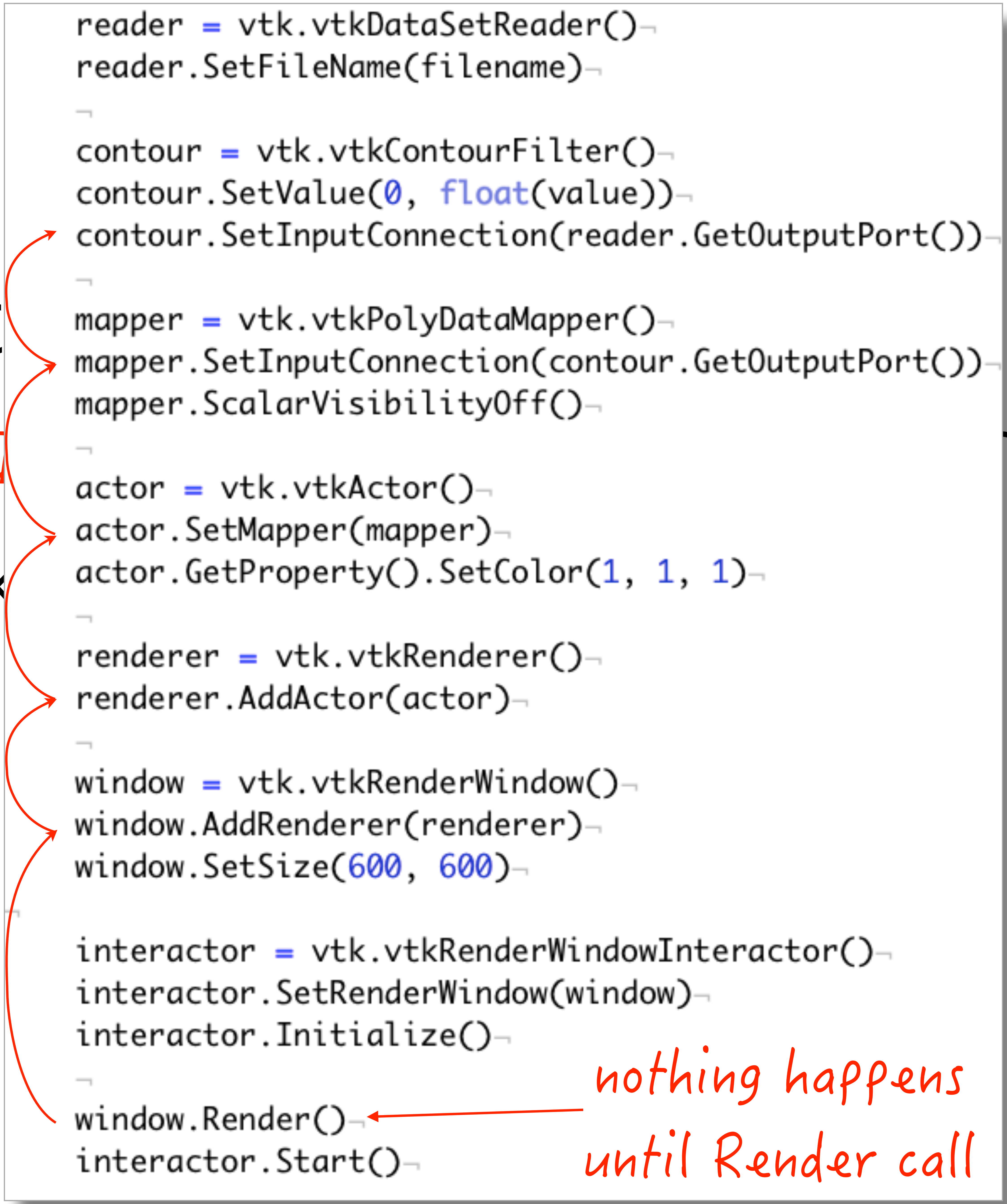


```
reader = vtk.vtkDataSetReader()
reader.SetFileName(filename)

contour = vtk.vtkContourFilter()
contour.SetValue(0, float(value))
contour.SetInputConnection(reader.GetOutputPort())

mapper = vtk.vtkPolyDataMapper()
mapper.SetInputConnection(contour.GetOutputPort())
mapper.ScalarVisibilityOff()

actor = vtk.vtkActor()
actor.SetMapper(mapper)
actor.GetProperty().SetColor(1, 1, 1)

renderer = vtk.vtkRenderer()
renderer.AddActor(actor)

window = vtk.vtkRenderWindow()
window.AddRenderer(renderer)
window.SetSize(600, 600)

interactor = vtk.vtkRenderWindowInteractor()
interactor.SetRenderWindow(window)
interactor.Initialize()

window.Render()
interactor.Start()
```

nothing happens until Render call

Vi ne

- Implic (*lazy evalua* nly (re)ex

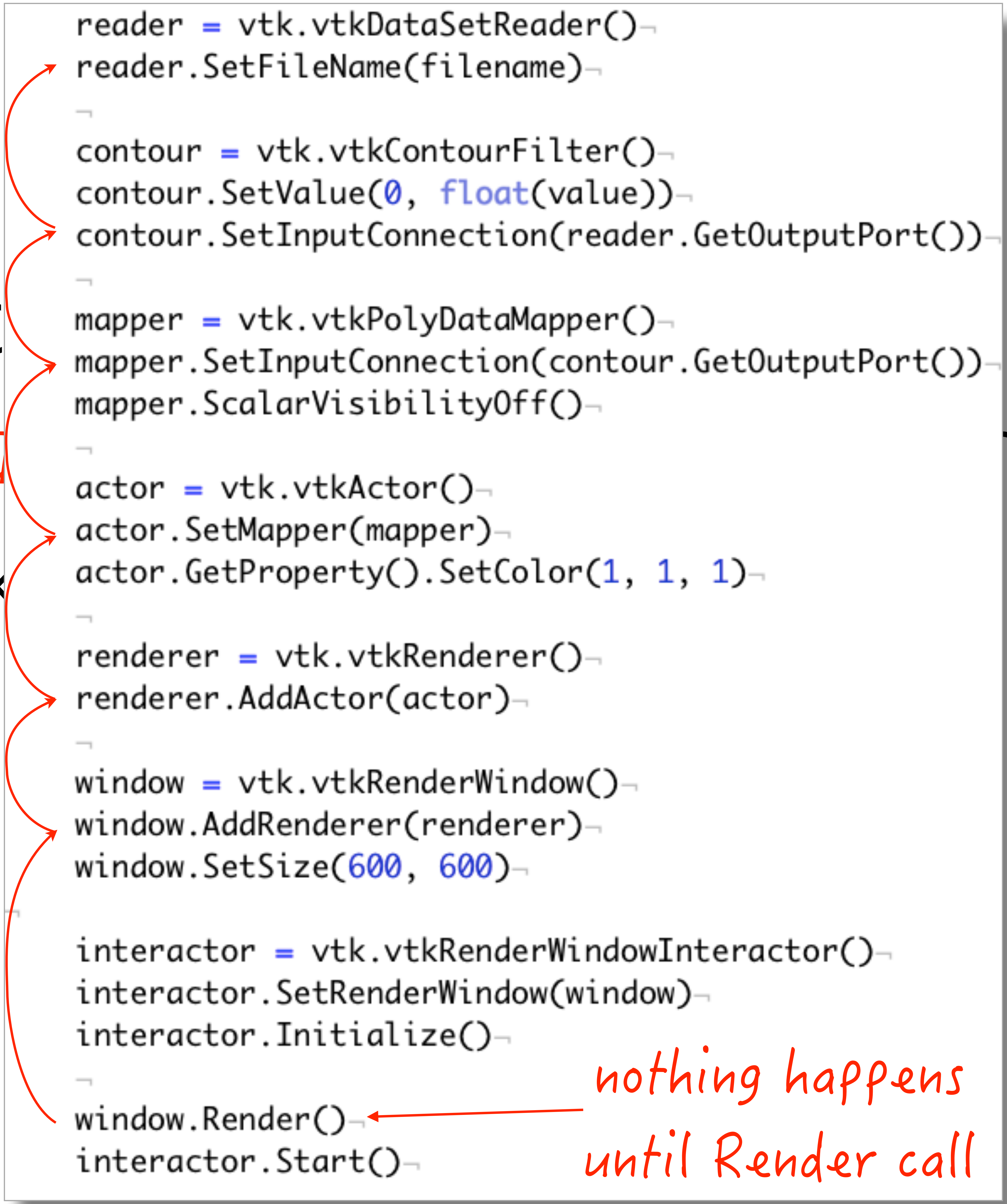


```
reader = vtk.vtkDataSetReader()
reader.SetFileName(filename)

contour = vtk.vtkContourFilter()
contour.SetValue(0, float(value))
contour.SetInputConnection(reader.GetOutputPort())

mapper = vtk.vtkPolyDataMapper()
mapper.SetInputConnection(contour.GetOutputPort())
mapper.ScalarVisibilityOff()

actor = vtk.vtkActor()
actor.SetMapper(mapper)
actor.GetProperty().SetColor(1, 1, 1)

renderer = vtk.vtkRenderer()
renderer.AddActor(actor)

window = vtk.vtkRenderWindow()
window.AddRenderer(renderer)
window.SetSize(600, 600)

interactor = vtk.vtkRenderWindowInteractor()
interactor.SetRenderWindow(window)
interactor.Initialize()

window.Render()
interactor.Start()
```

nothing happens until Render call

Vi ne

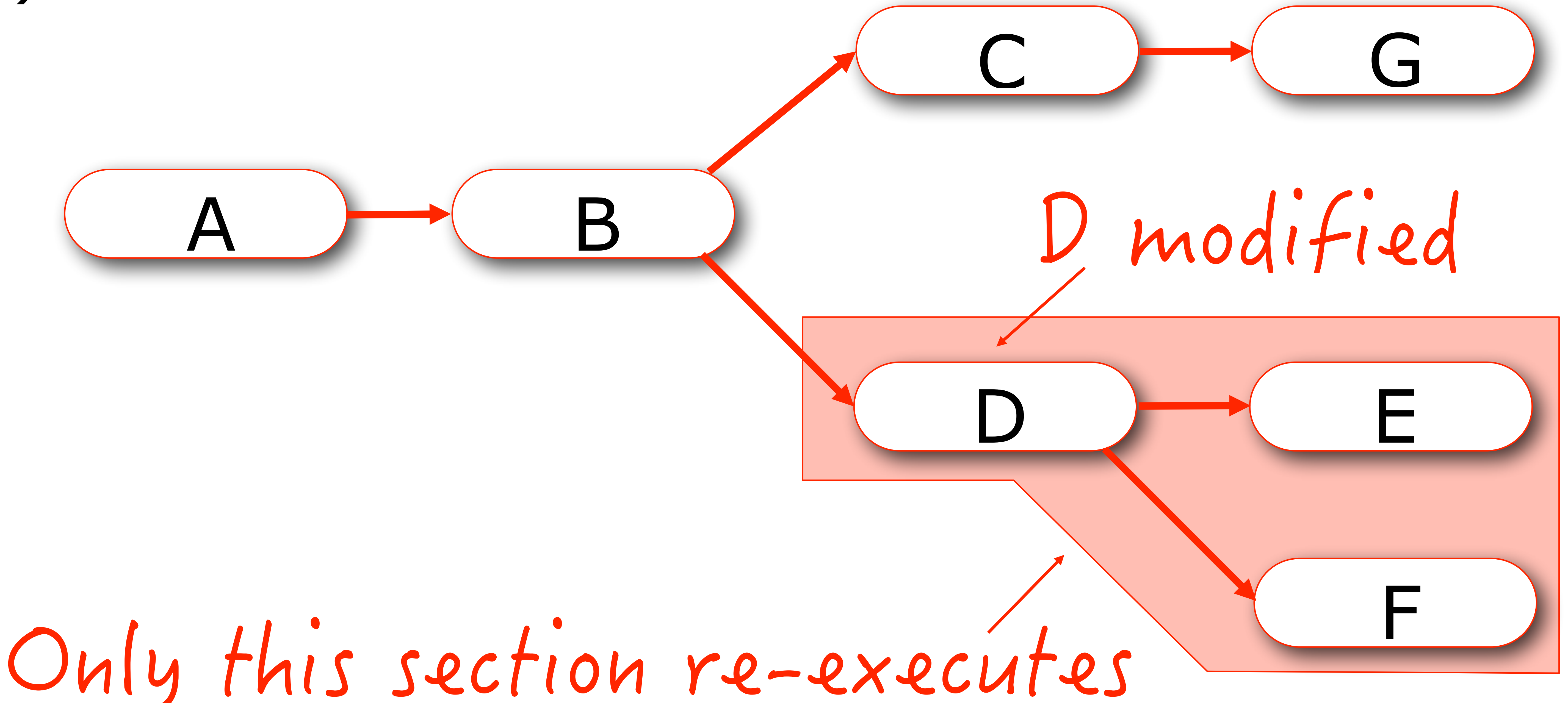
- Implic (*lazy evalua* nly (re)ex



```
reader = vtk.vtkDataSetReader()
reader.SetFileName(filename)

contour = vtk.vtkContourFilter()
contour.SetValue(0, float(value))
contour.SetInputConnection(reader.GetOutputPort())

mapper = vtk.vtkPolyDataMapper()
mapper.SetInputConnection(contour.GetOutputPort())
mapper.ScalarVisibilityOff()

actor = vtk.vtkActor()
actor.SetMapper(mapper)
actor.GetProperty().SetColor(1, 1, 1)

renderer = vtk.vtkRenderer()
renderer.AddActor(actor)

window = vtk.vtkRenderWindow()
window.AddRenderer(renderer)
window.SetSize(600, 600)

interactor = vtk.vtkRenderWindowInteractor()
interactor.SetRenderWindow(window)
interactor.Initialize()

window.Render()
interactor.Start()
```

Visualization Pipeline

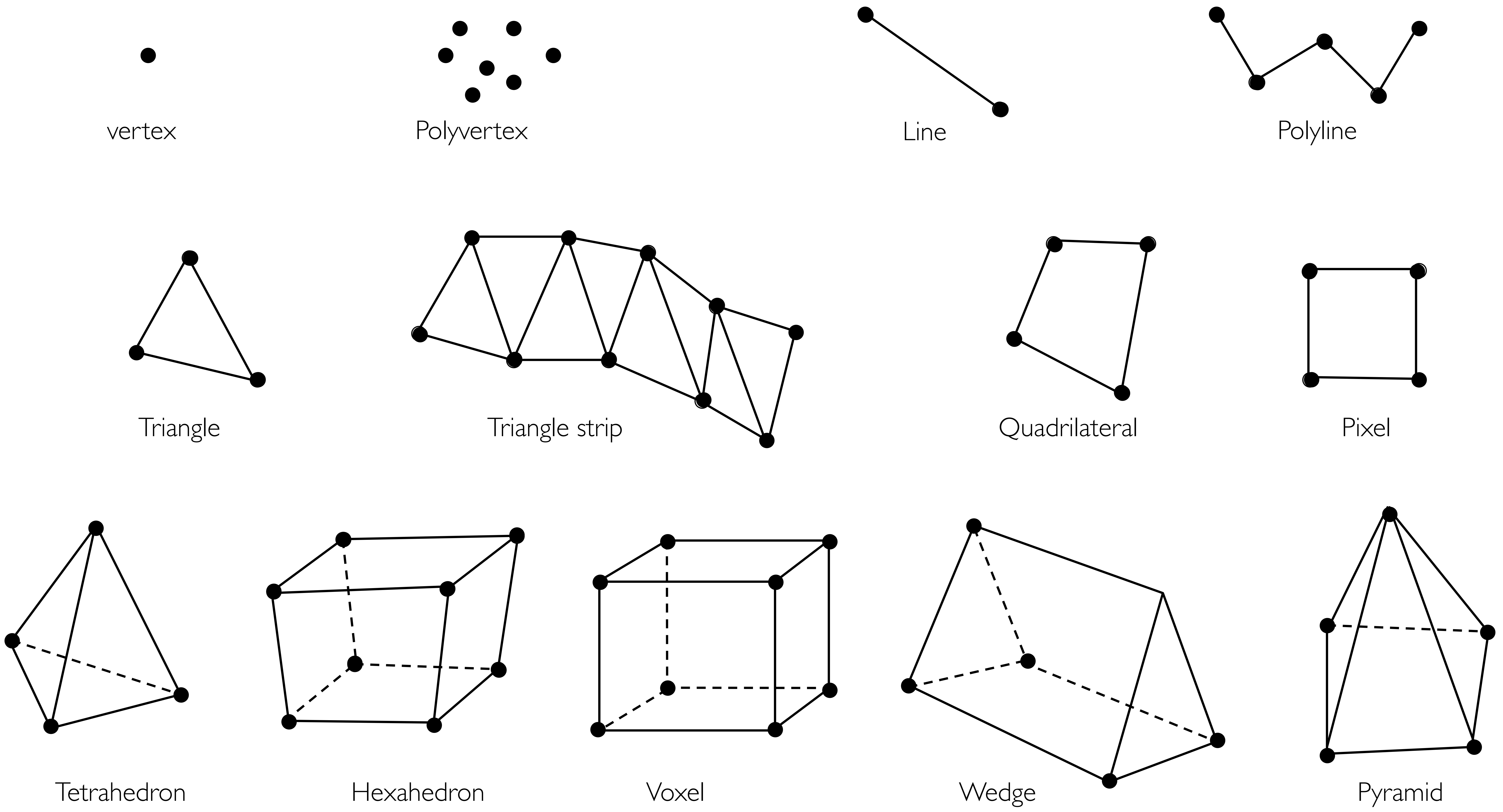
- Implicit control of execution (*lazy evaluation*): algorithms are only (re)executed when needed



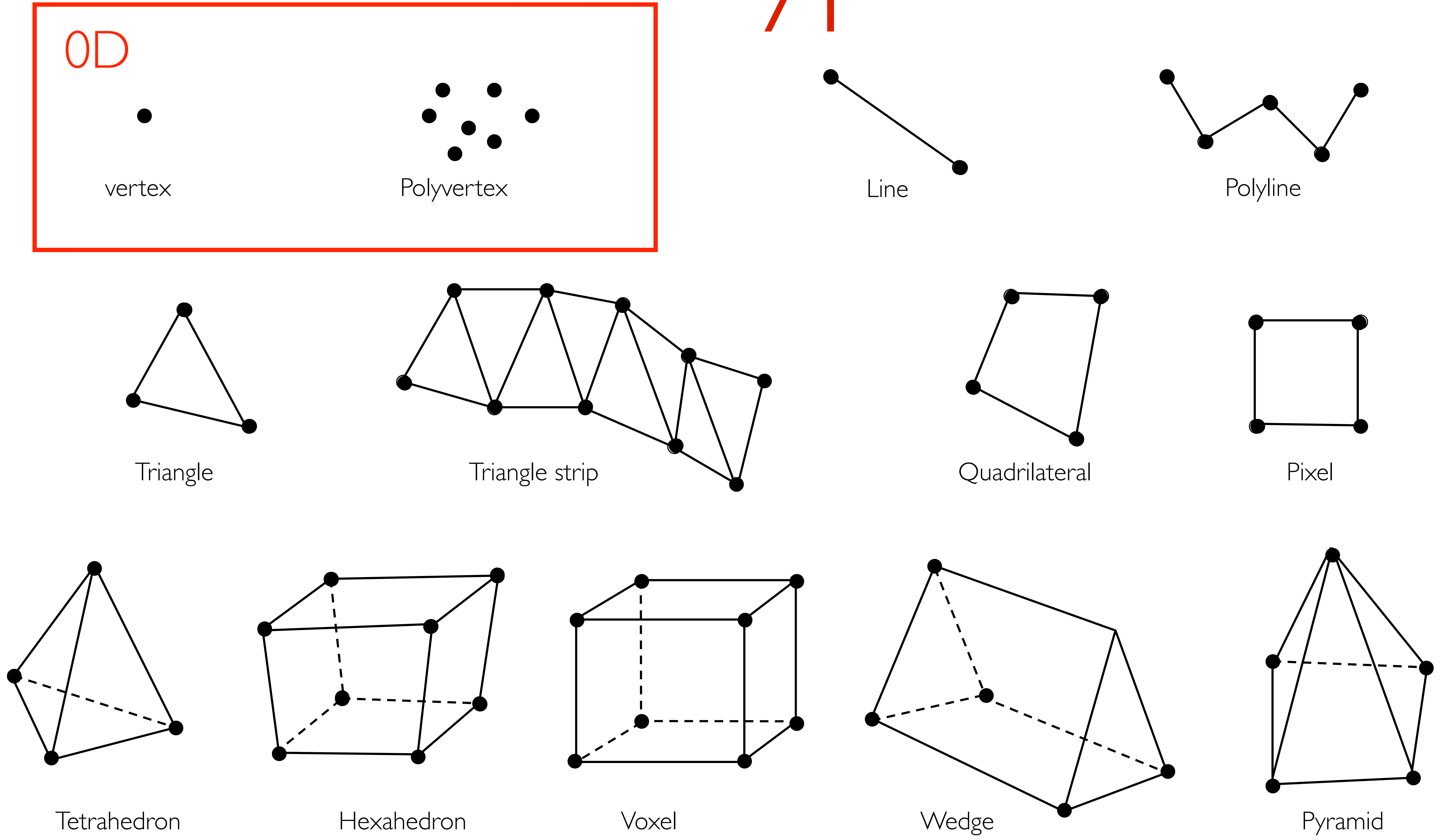
Outline

- Visualization pipeline
- Internal data representation
- Examples

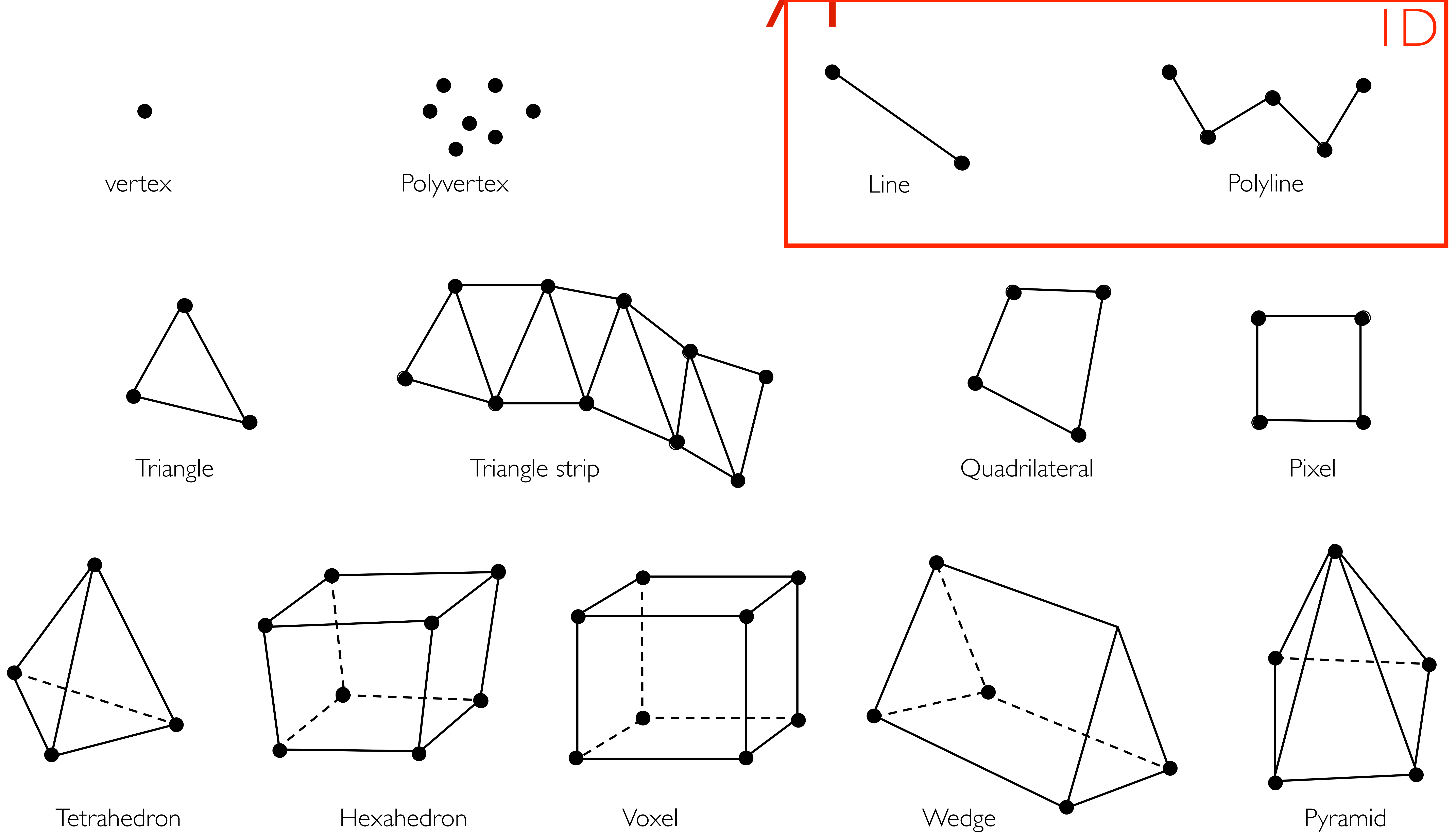
Cell Types


vertex
Polyvertex
Line
Polyline
Triangle
Triangle strip
Quadrilateral
Pixel
Tetrahedron
Hexahedron
Voxel
Wedge
Pyramid

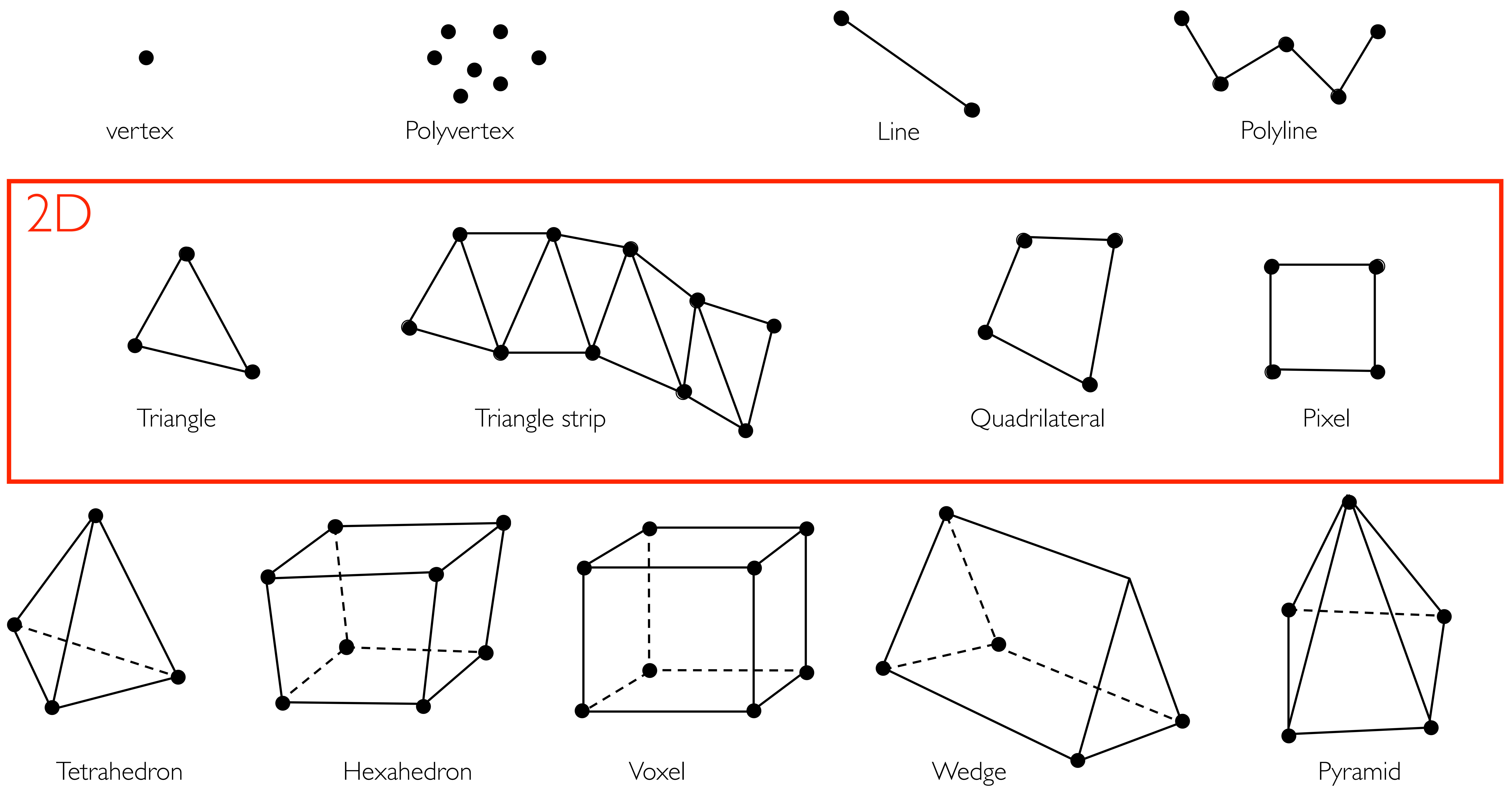
Cell Types


0D
vertex
Polyvertex
Line
Polyline
Triangle
Triangle strip
Quadrilateral
Pixel
Tetrahedron
Hexahedron
Voxel
Wedge
Pyramid

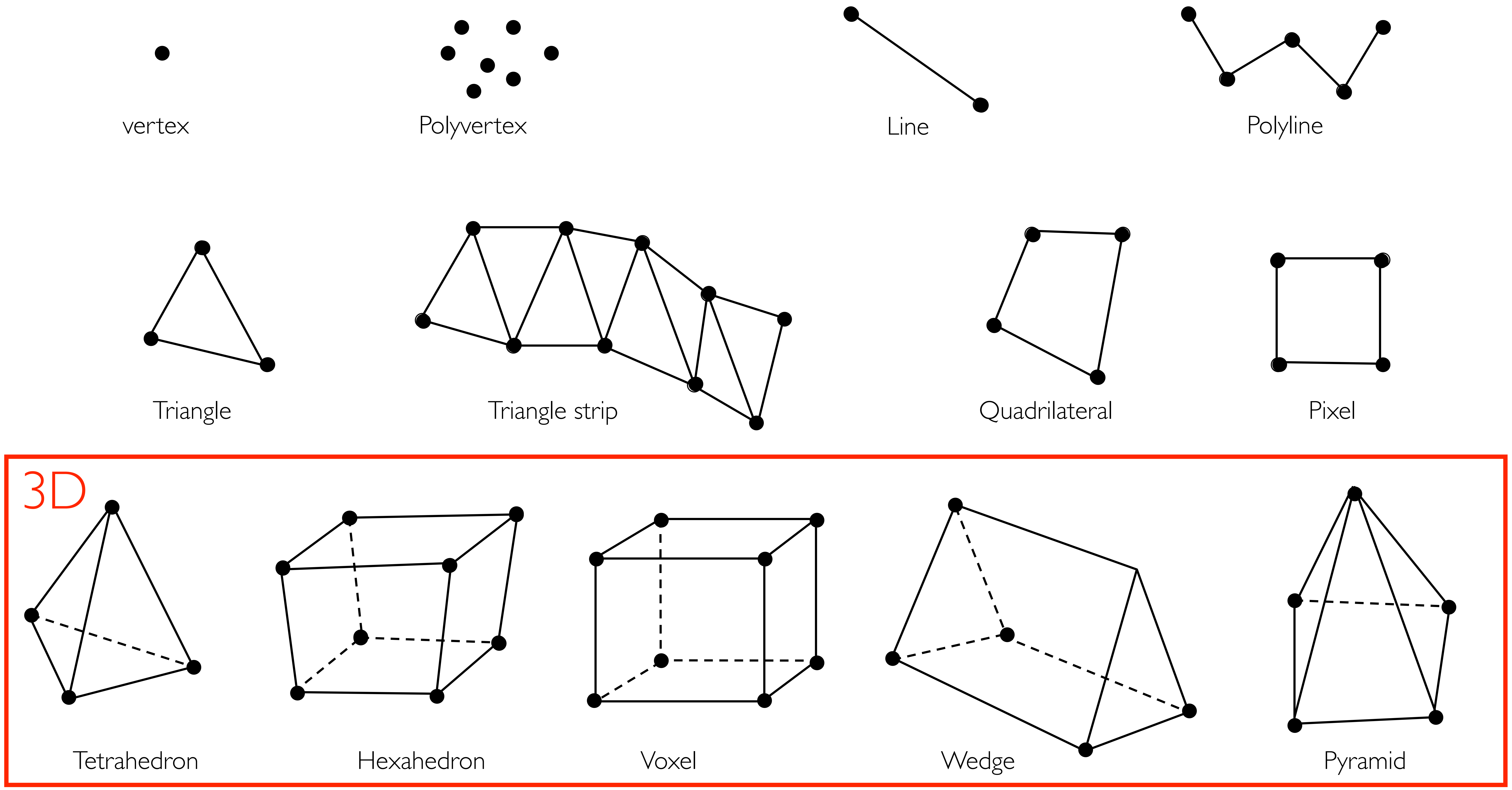
Cell Types


1D
vertex
Polyvertex
Line
Polyline
Triangle
Triangle strip
Quadrilateral
Pixel
Tetrahedron
Hexahedron
Voxel
Wedge
Pyramid

Cell Types

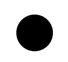

vertex
Polyvertex
Line
Polyline
2D
Triangle
Triangle strip
Quadrilateral
Pixel
Tetrahedron
Hexahedron
Voxel
Wedge
Pyramid

Cell Types

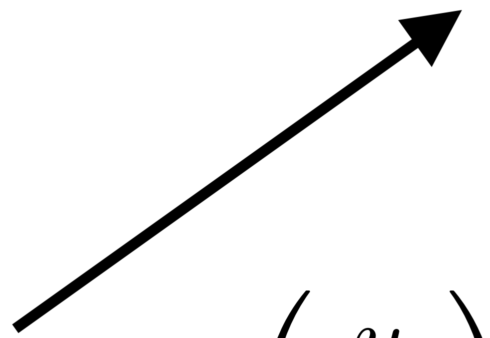

vertex
Polyvertex
Line
Polyline
Triangle
Triangle strip
Quadrilateral
Pixel
3D
Tetrahedron
Hexahedron
Voxel
Wedge
Pyramid

Data Attributes

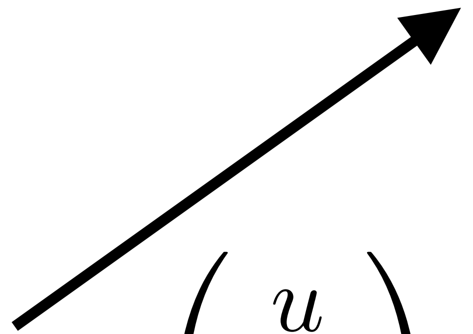
Cell-wise / point-wise (*vtkDataSetAttribute*)



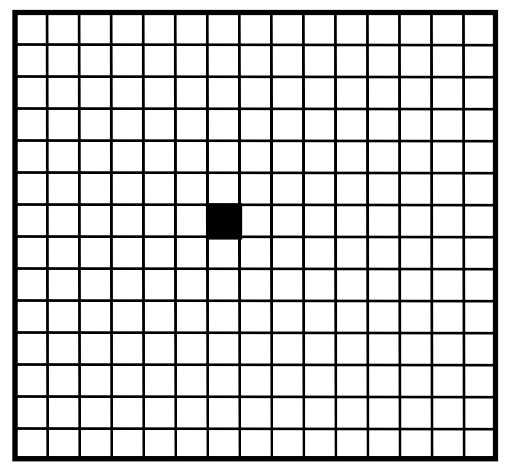
Scalar



3D vector $\begin{pmatrix} u \\ v \\ w \end{pmatrix}$



normal $\begin{pmatrix} u \\ v \\ w \end{pmatrix}, u^2 + v^2 + w^2 = 1$

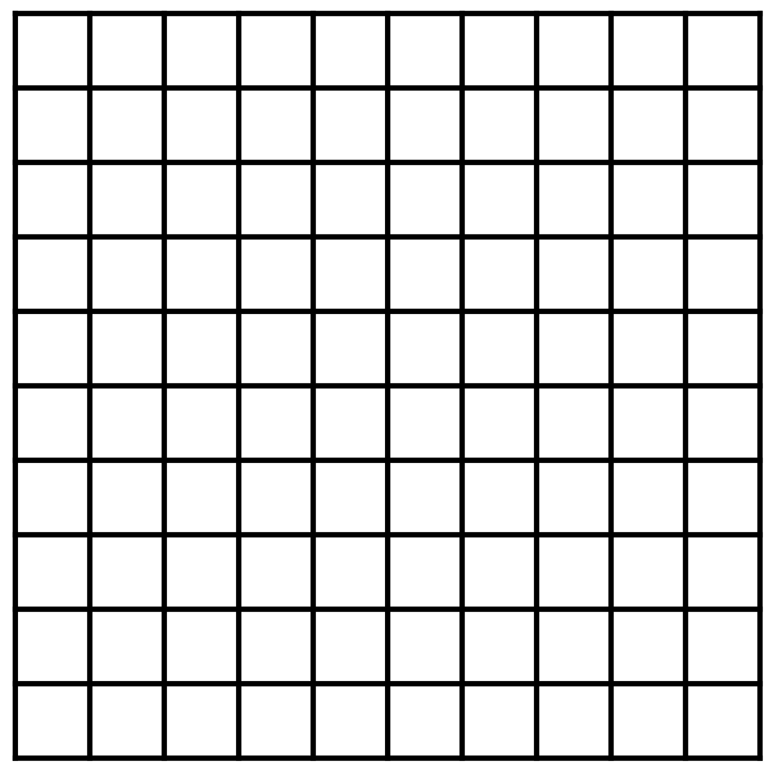


Texture coordinate (u,v) or (u,v,w)

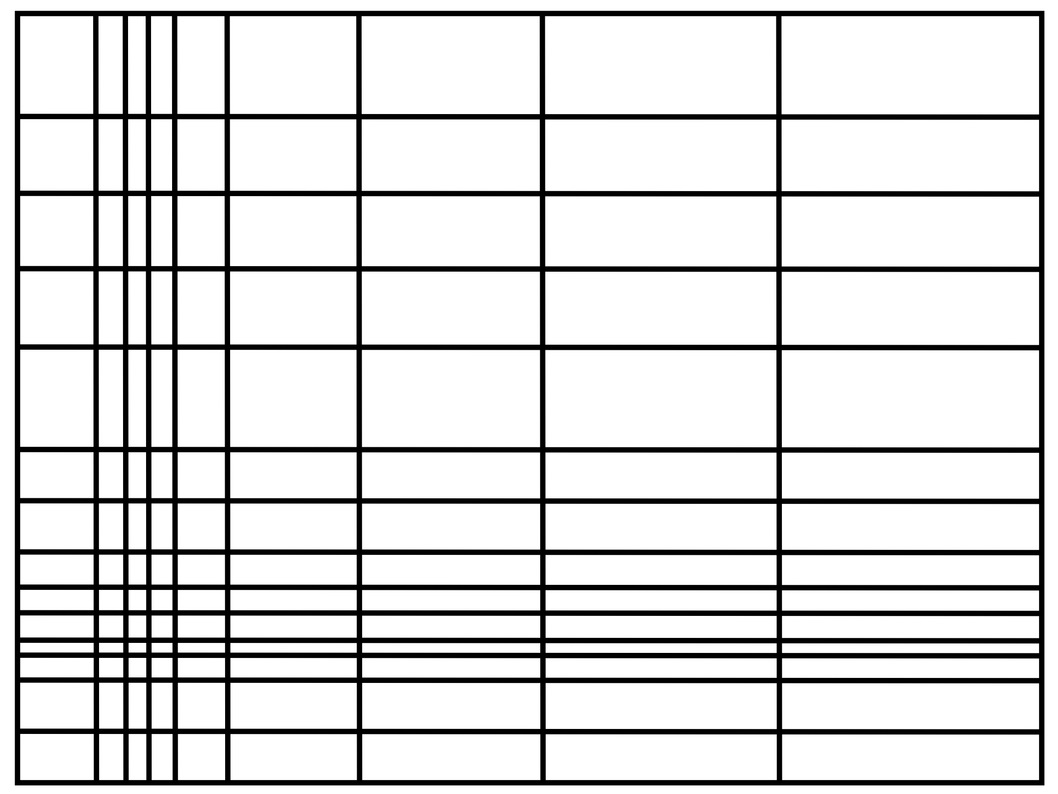
$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

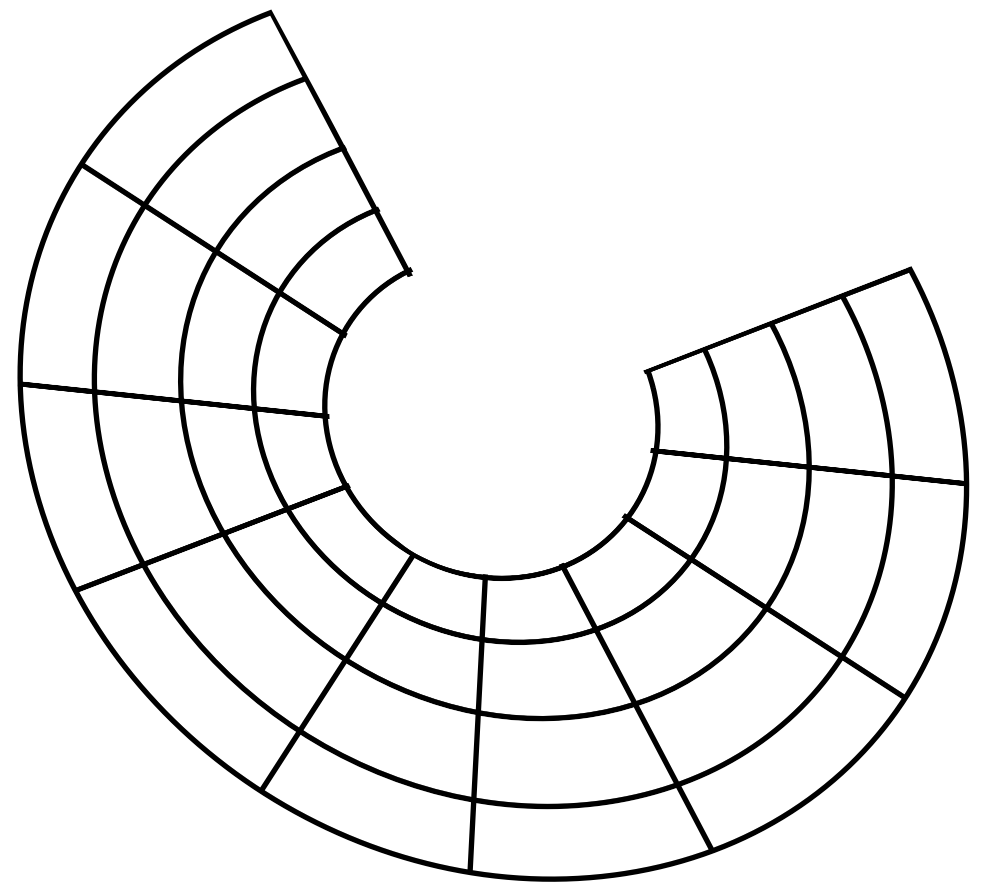
2nd order tensor (3x3 matrix)

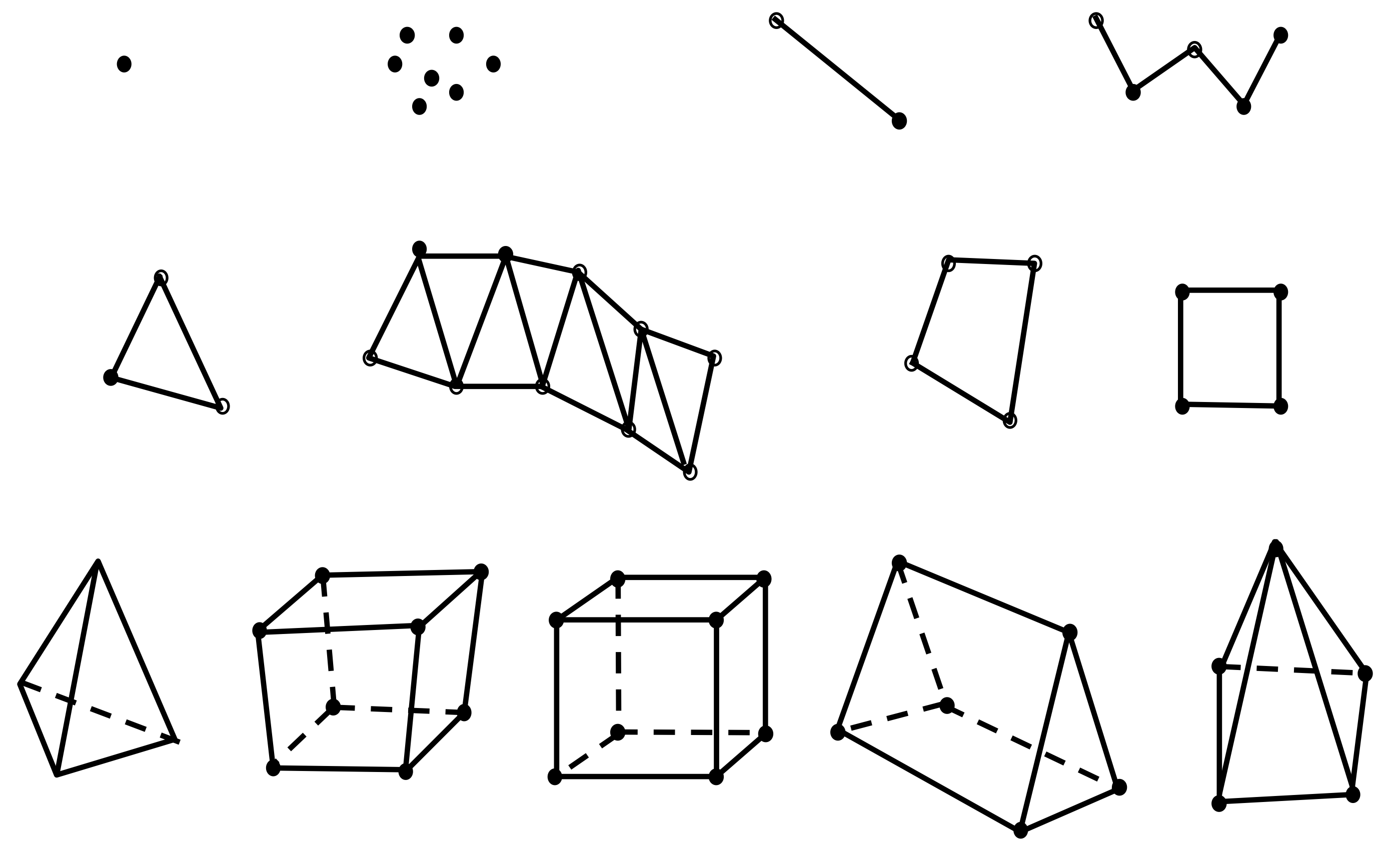
Dataset Types



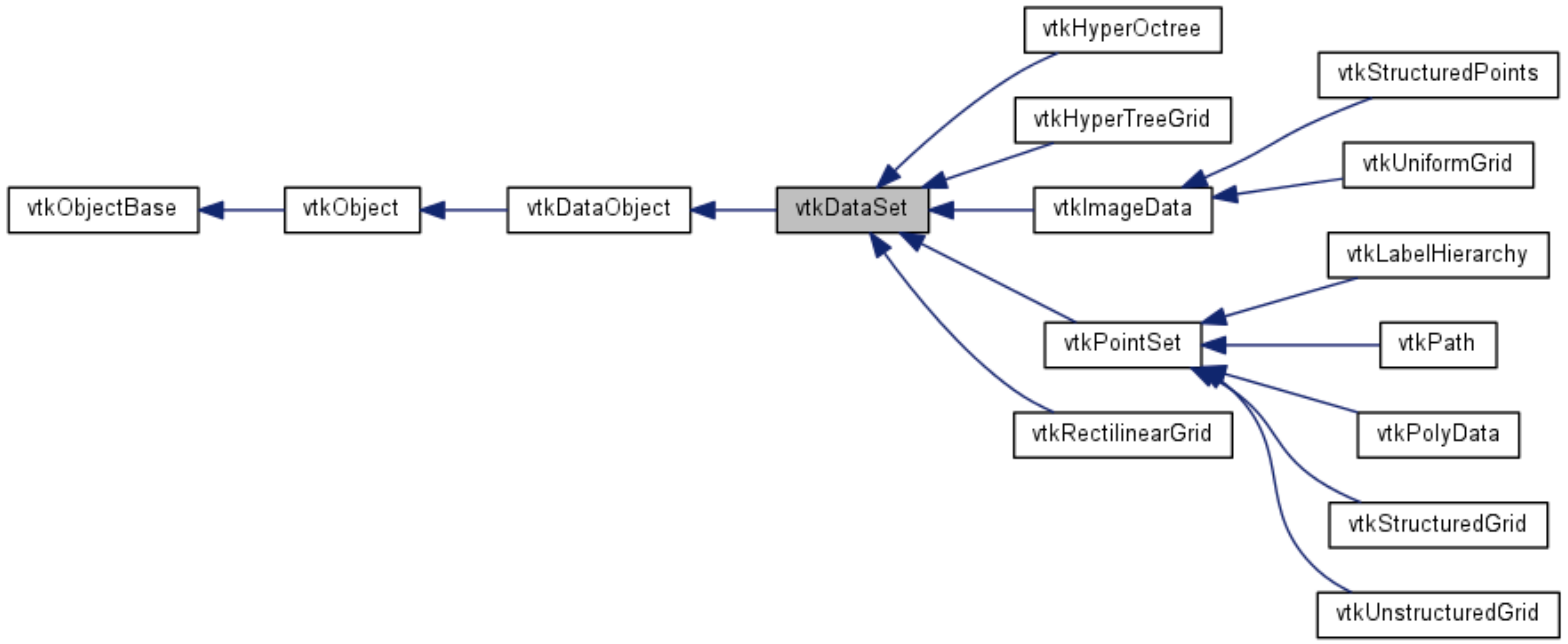
Image


Rectilinear grid

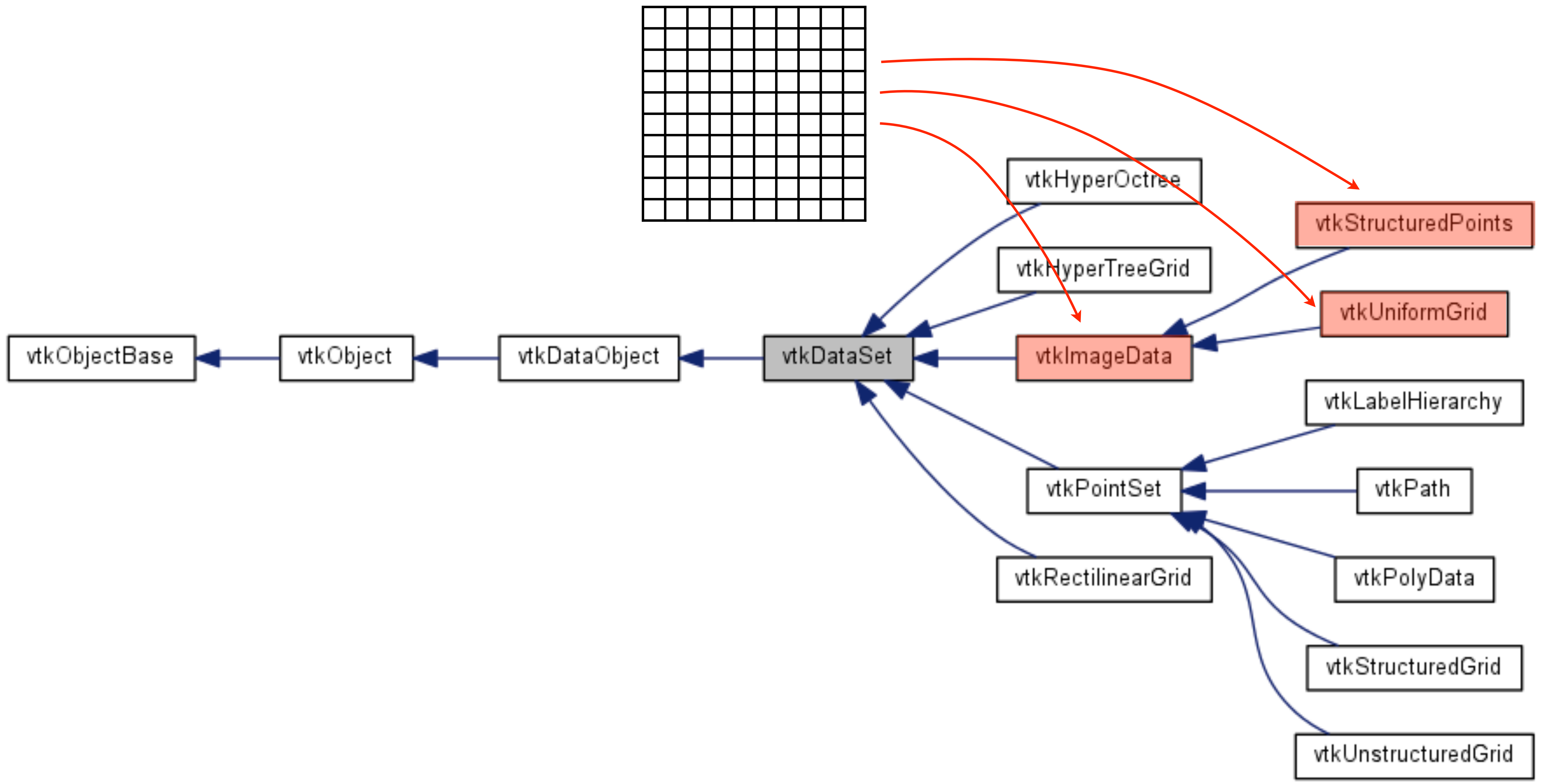

Structured (curvilinear) grid


Unstructured grid

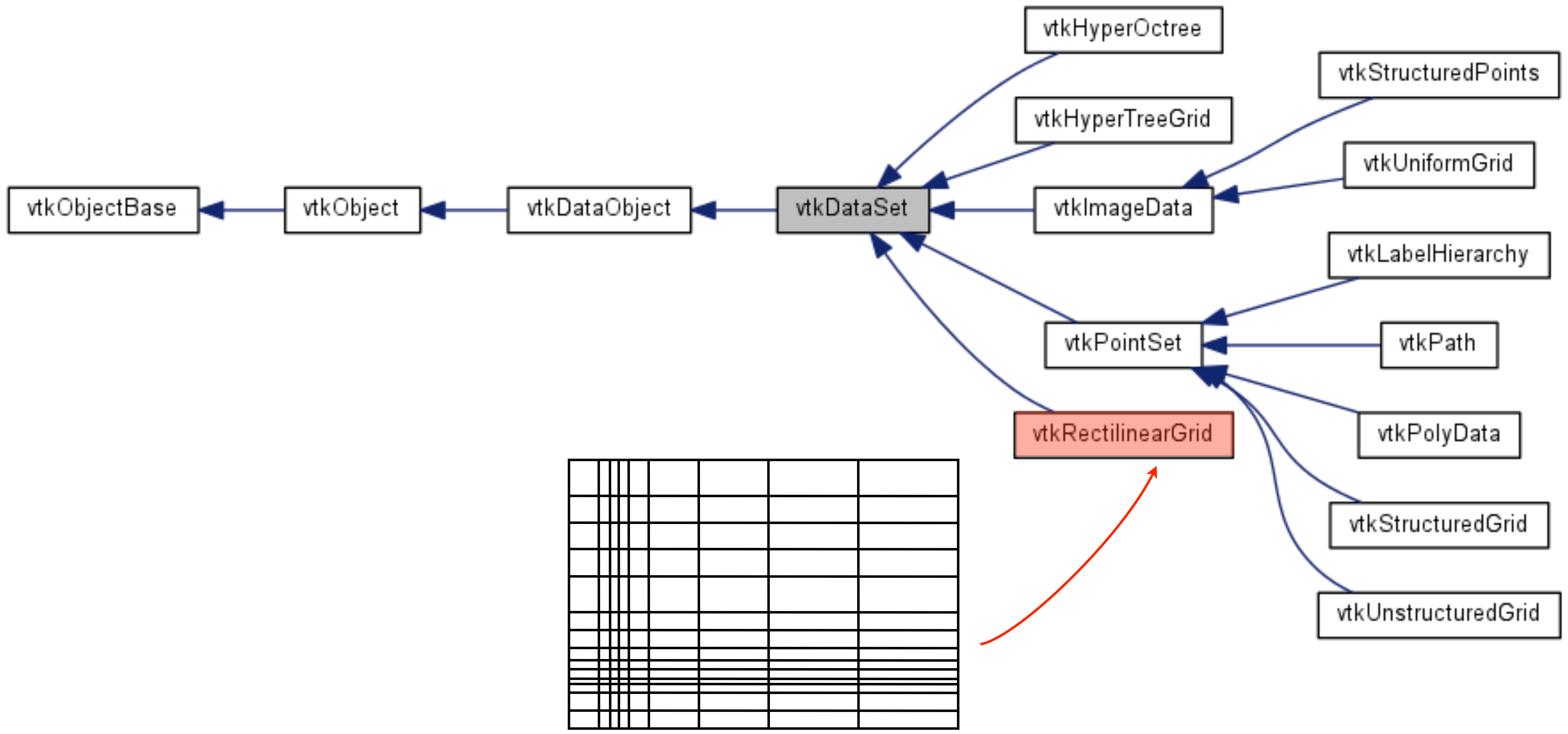
Dataset Hierarchy



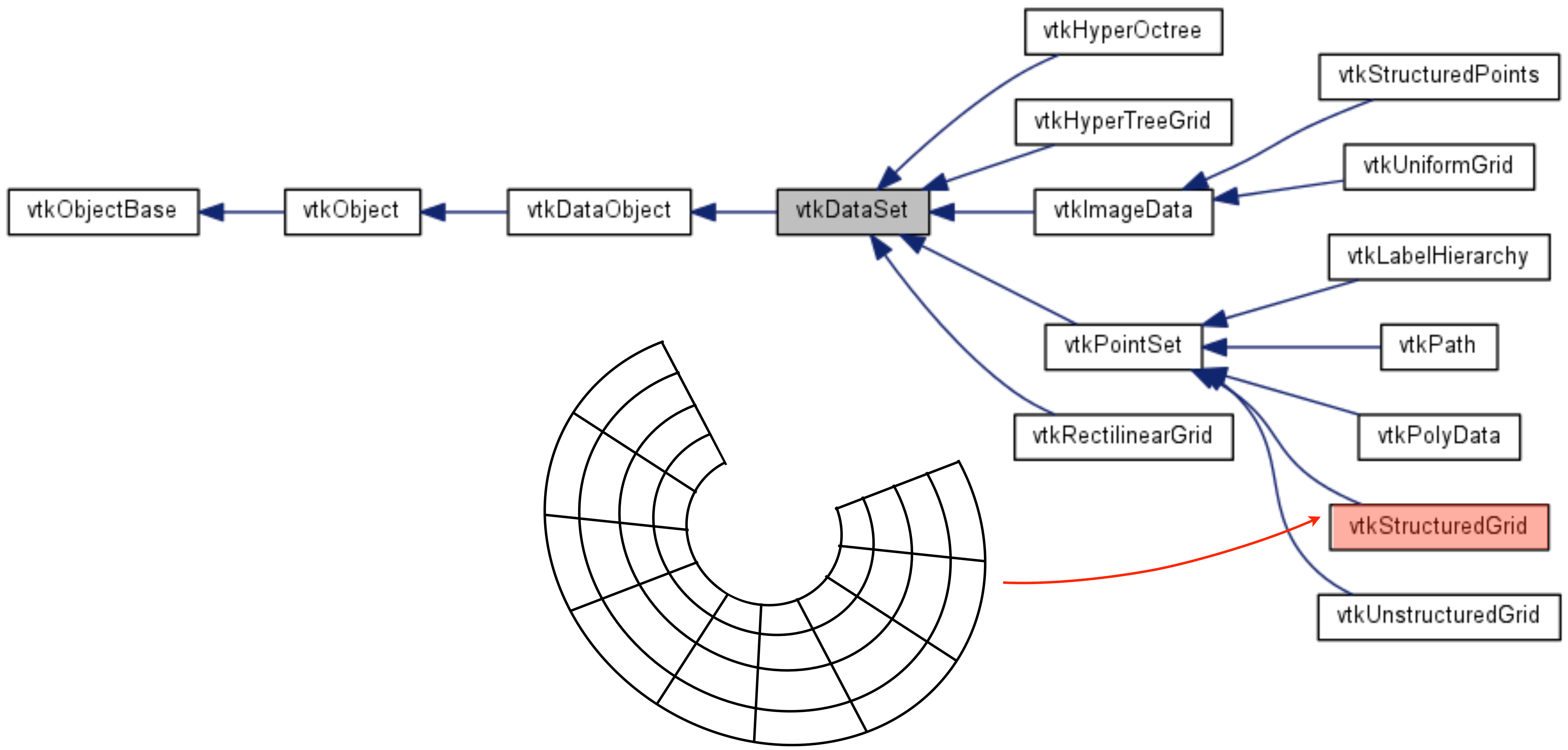
Dataset Hierarchy



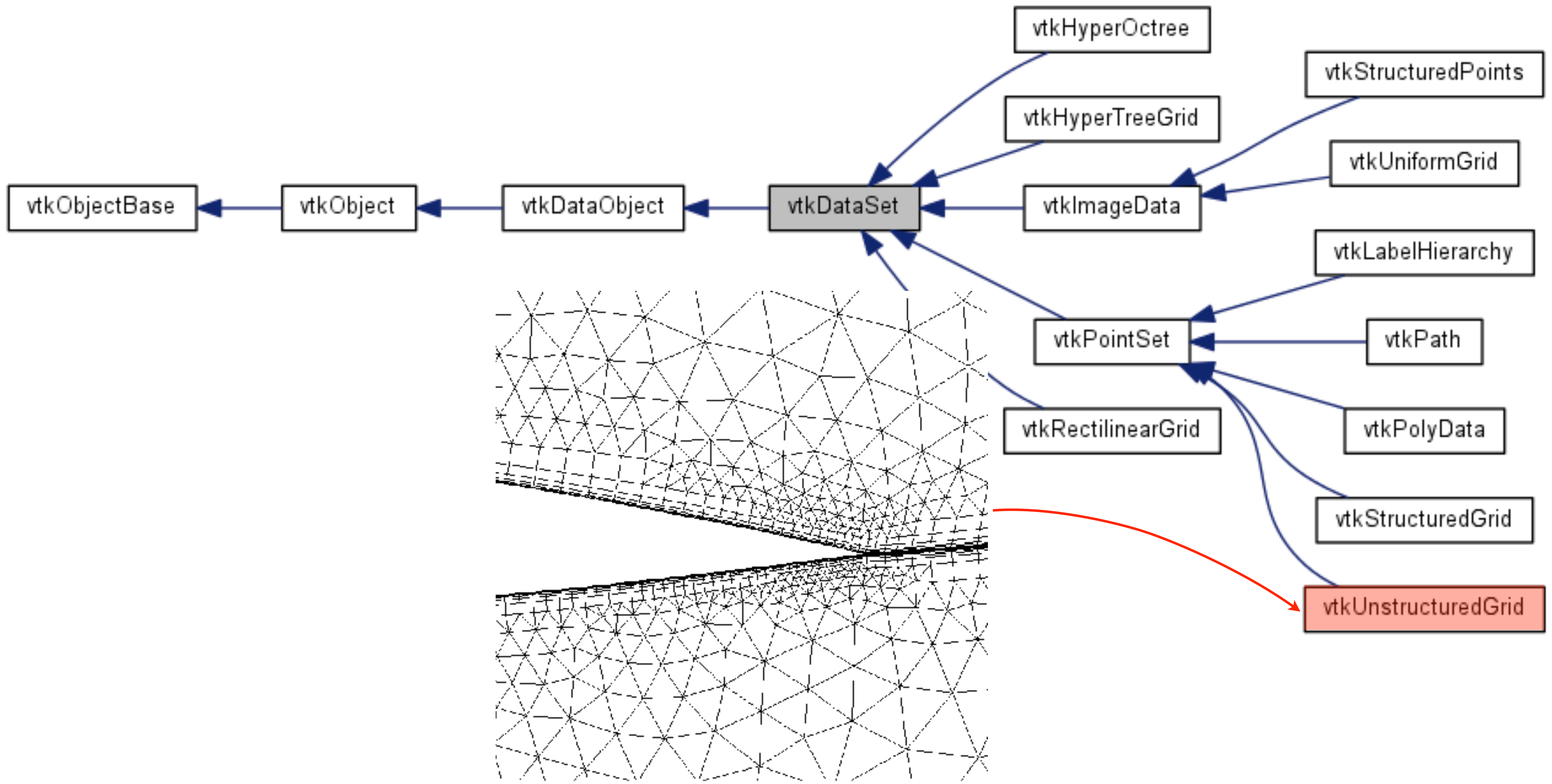
Dataset Hierarchy



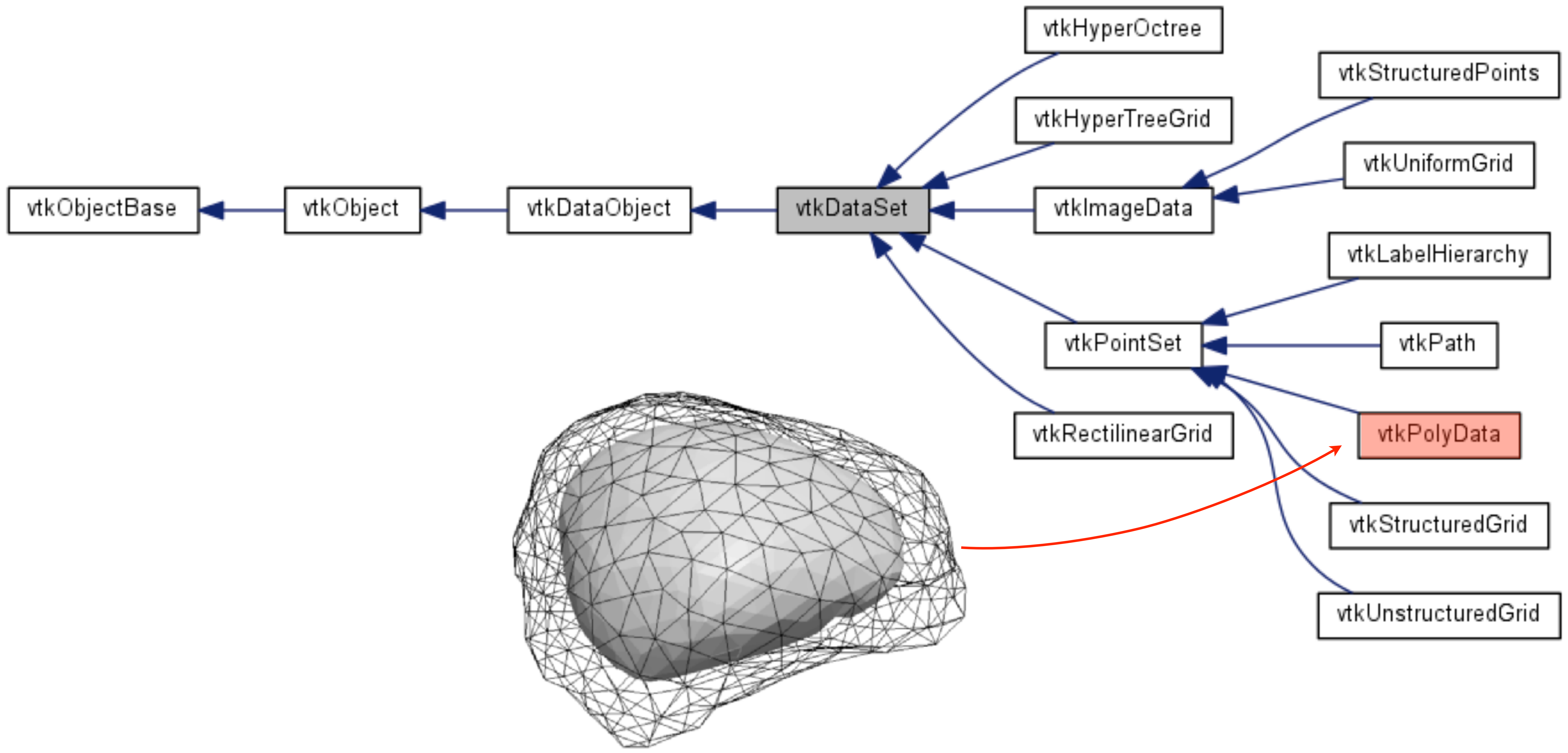
Dataset Hierarchy



Dataset Hierarchy



Dataset Hierarchy



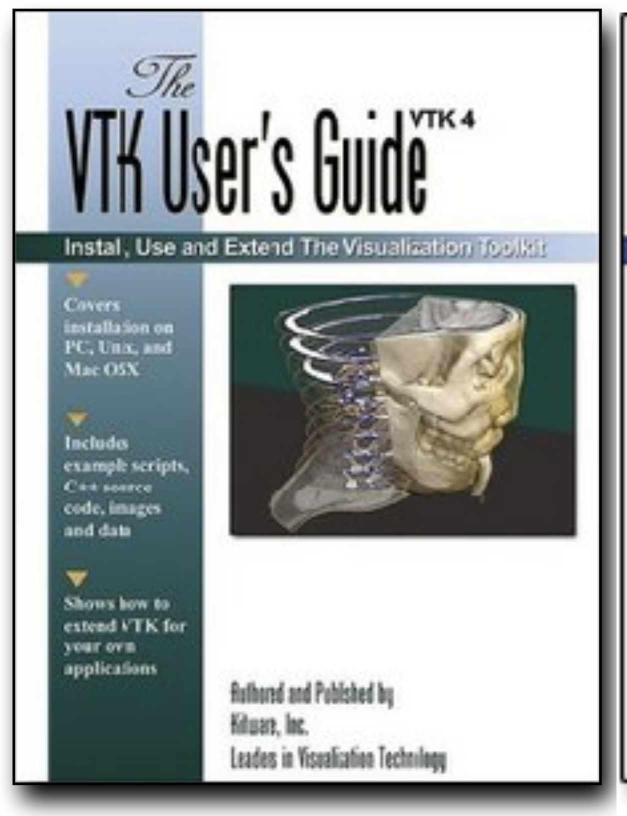
Outline

- Visualization pipeline
- Internal data representation
- Examples

Demos

Additional References

- VTK Wiki http://www.vtk.org/Wiki/VTK
- VTK Tutorial
  in source code under `Examples/Tutorial`
- VTK Examples
  in source code under `Examples/`, primarily `VisualizationAlgorithms, Rendering, Graphics, Geometric Modeling`
- *VTK User's Guide*
  Kitware Inc., ISBN 1-930934-0804
- *The Visualization Toolkit*
  An object-oriented Approach to 3D Graphics,
  3rd edition, W. Schroeder, K. Martin, B. Lorensen, Kitware
  ISBN 1-930934-07-6 (available as free [PDF](PDF))


The VTK User's Guide VTK 4
Install, Use and Extend The Visualization Toolkit
Authored and Published by Kitware, Inc. Leaders in Visualization Technology

The Visualization Toolkit 3rd Edition
An Object-Oriented Approach to 3D Graphics
Will Schroeder
Ken Martin
Bill Lorensen