Big Data Visual Analytics (CS 661)

Instructor: Soumya Dutta

Department of Computer Science and Engineering

Indian Institute of Technology Kanpur (IITK)

email: soumyad@cse.iitk.ac.in

Acknowledgements

- Some of the following slides are adapted from the excellent course materials and tutorials made available by:
  - Prof. Han-Wei Shen (The Ohio State University)
  - Prof. Klaus Mueller (State University of New York at Stony Brook)
  - David DeMarle (Intel)

Reading Materials for Lectures 1 & 2

- Visualization Analysis and Design by T. Munzner
  - Chapter 1, Chapter 2, Chapter 5
  - Book available online at IITK Library

Reading Materials for Lecture 3 (Today)

- The Visualization Toolkit by Will Schroeder, Ken Martin, Bill Lorensen
  - Chapter 1, Chapter 4, Chapter 5
  - Get the pdf: https://vtk.org/vtk-textbook/
- Reference for learning VTK:
  - VTK User's Guide
  - Get the pdf: https://vtk.org/vtk-users-guide/
  - Examples: https://kitware.github.io/vtk-examples/site/Python/

Scientific Data Analysis and Visualization

Imaging, Computer Graphics, and Visualization

Image processing

- Study of 2D pictures, or images
- Transform, extract features
- Analyze the data

Computer Graphics

- Process of creating images using a computer
- 2D paint-and-draw
- Sophisticated 3D rendering techniques, animation

Visualization

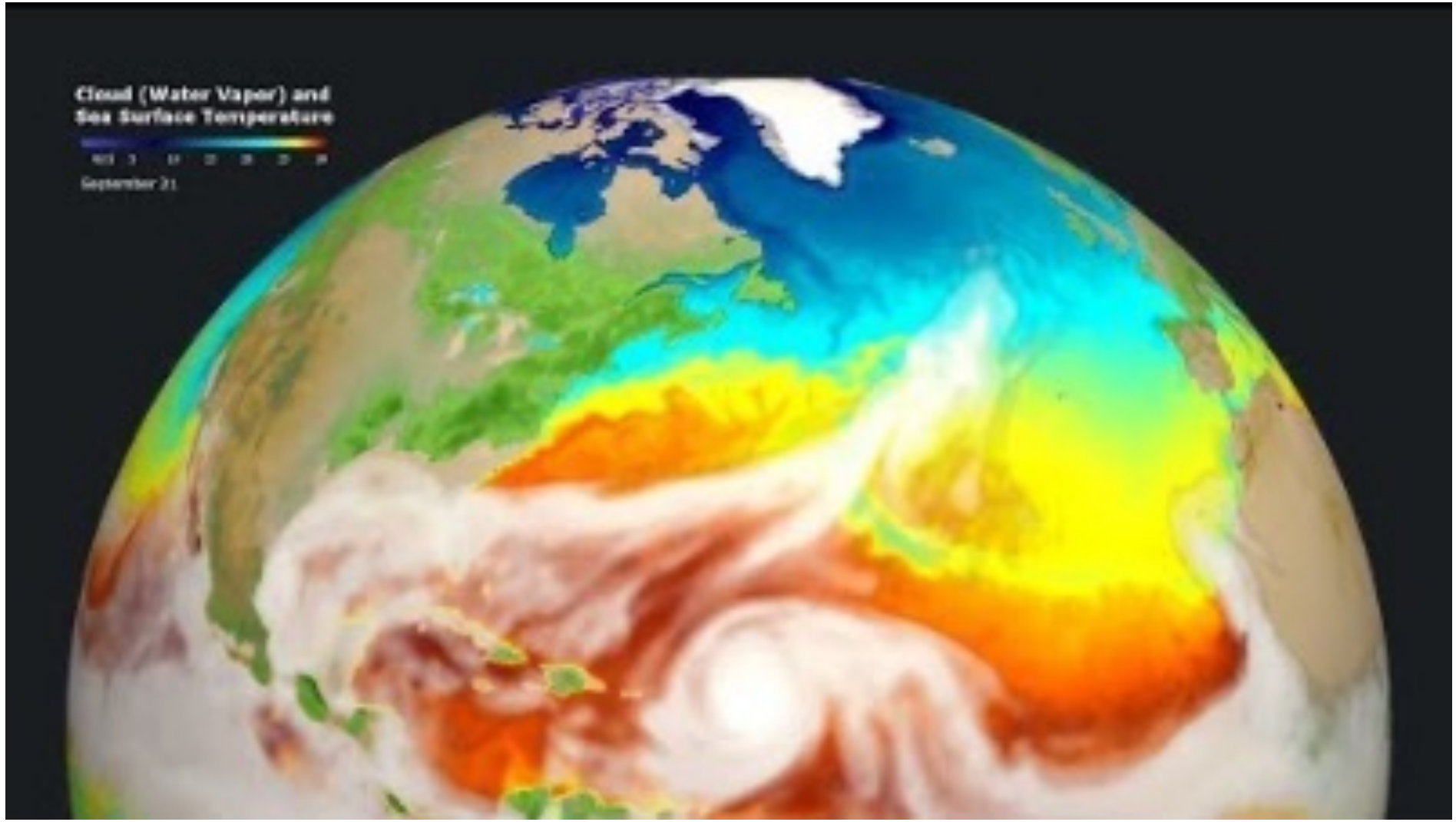
- Process of exploring, transforming, and viewing data as images, and plots
- Gain understanding and insight into the data

"the purpose of visualization is insight, not pictures" – Ben Shneiderman

Scientific Visualization (SciVis)

- Technique for comprehending data & knowledge extraction from the results of simulations, computations, or measurements
- Field in computer science that encompasses
  - data representation and processing algorithms
  - visual representations
  - user interface
  - other sensory presentation such as sound, touch, AR, VR
- Relatively (new) domain of research (~ 36 years)
  - Formal inception in 1987 by US NSF
  - “Visualization in Scientific Computing” by McCormick et al.

Example – Application in Climate Science



Example – Application in Asteroid Impact Assessment

Big Data in Scientific Visualization

- Big data describes datasets that are so large, complex, or rapidly changing that they push the very limits of our analytical capability[1]

Megascale (10^6 FLOPS) (1961)

↓

Gigascale (10^9 FLOPS) (1972)

↓

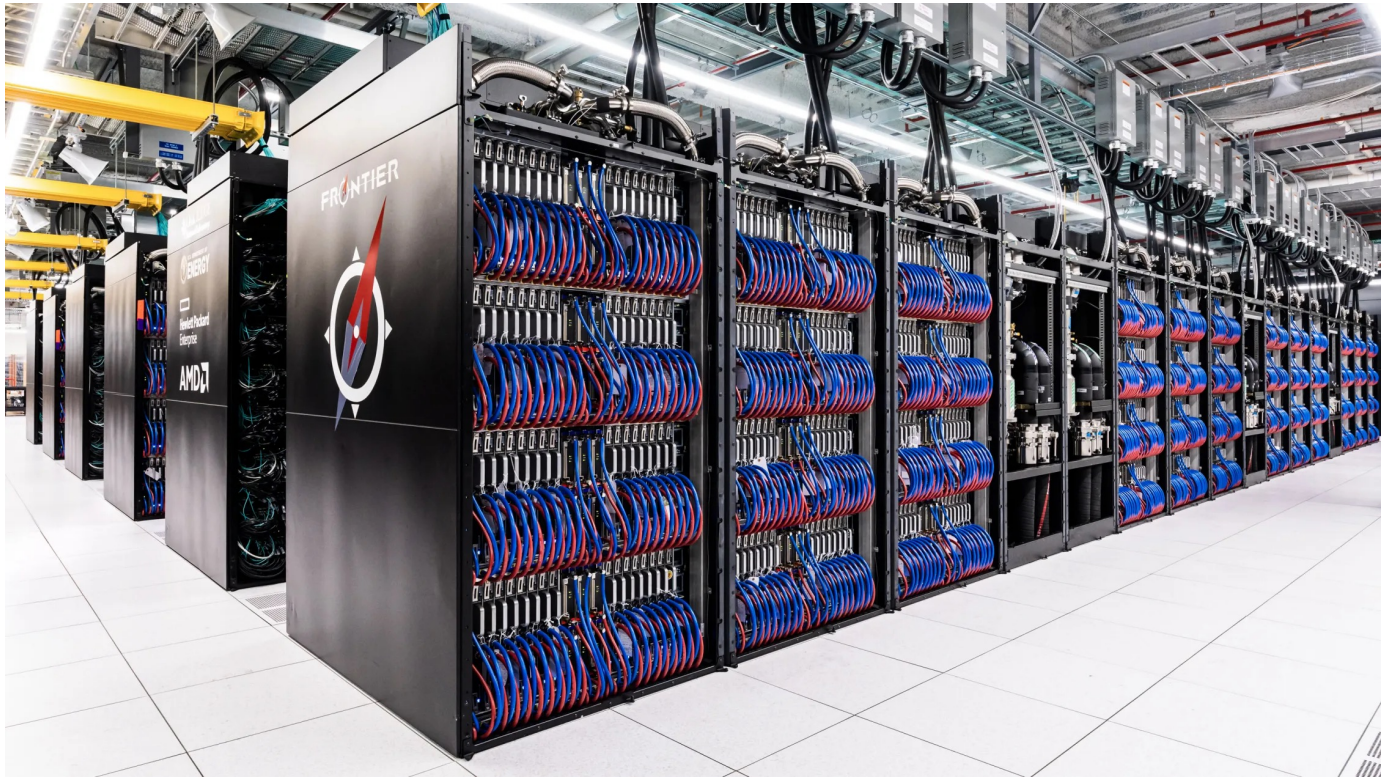
Terascale (10^{12} FLOPS) (1997)

↓

Petascale (10^{15} FLOPS) (2009)

↓

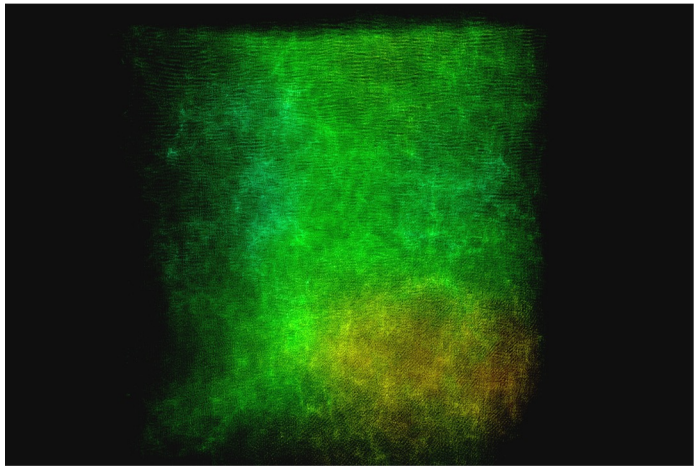
Exascale (10^{18} FLOPS) (2022)



FRONTIER World's fastest supercomputer at Oak Ridge National Laboratory, credit: OLCF

[1]NIST Big Data Public Working Group: (https://bigdatawg.nist.gov/_uploadfiles/NIST.SP.1500-1.pdf)

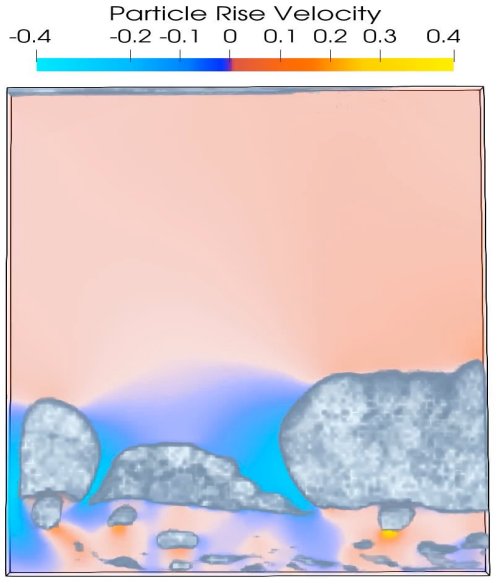
Big Data in Scientific Visualization



Exa-Star: Simulation of Universe

The target science includes simulations of astrophysical explosions (such as supernovae and neutron stars) to understand the cosmic origin and the fundamental physics.

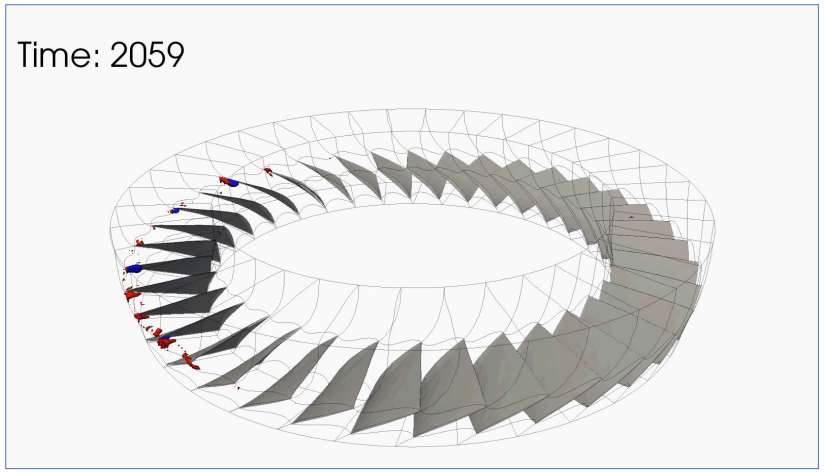
Targeted to simulate 8192^3 resolution grid with 6 scalar quantities, i.e., 4TB per time step



MFIX-Exa: Studies CLR

Deliver high-fidelity multiphase flow modeling capabilities for applications in reducing CO_2 emissions from fossil fuel power plants.

The simulation will use billions of particles to model the physics. A simulation with 200 million particles and 100 time steps will need 160 TB storage.



TURBO: Studies Engine Stall

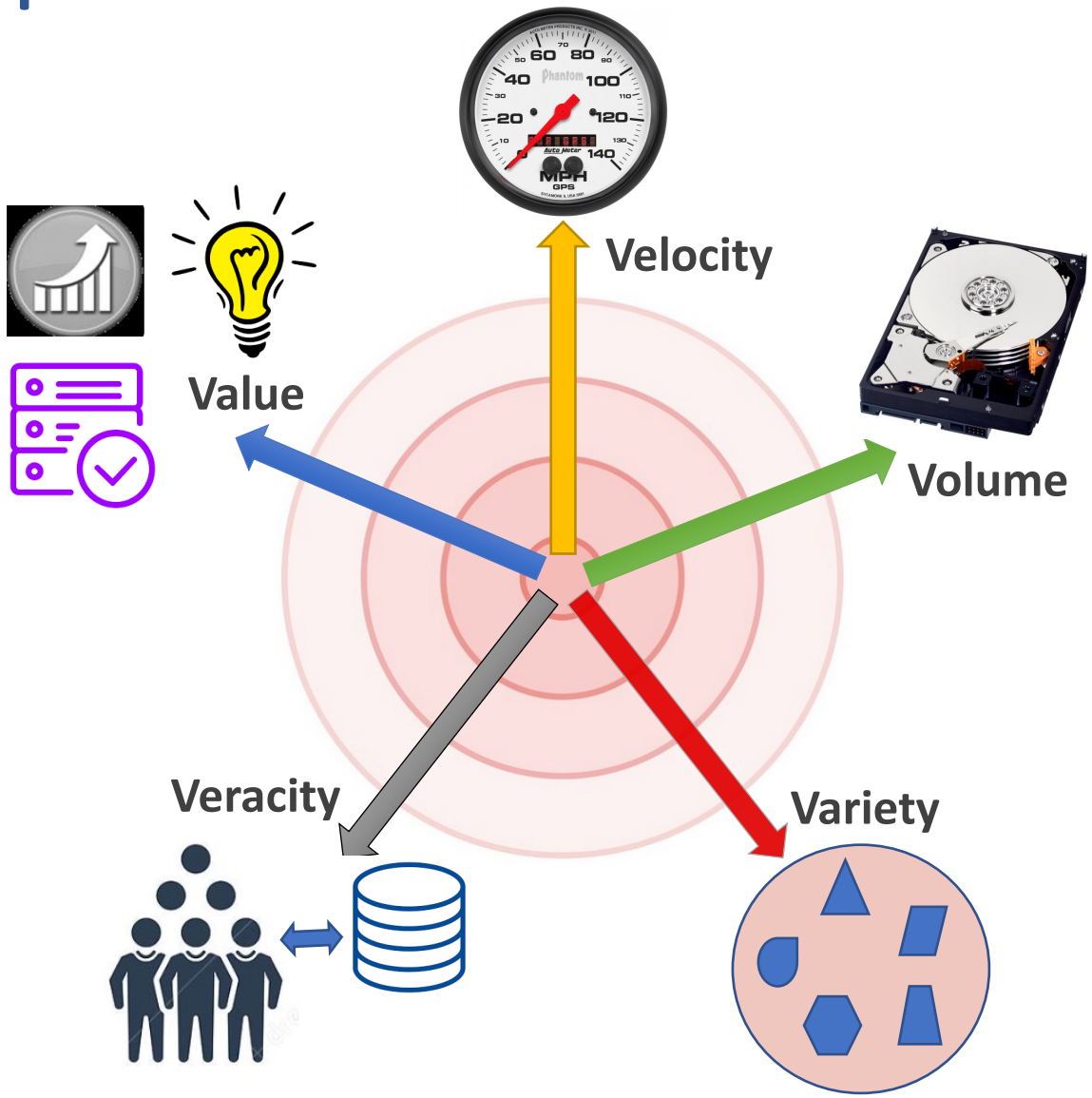
Enable understanding of rotating stall in Jet engines and develop stall detection as well as stall prevention measures.

Just a single revolution produces 5TB of data and hundreds of revolutions are needed to run.

Big Data Characterization

- The “5Vs” of Big Data

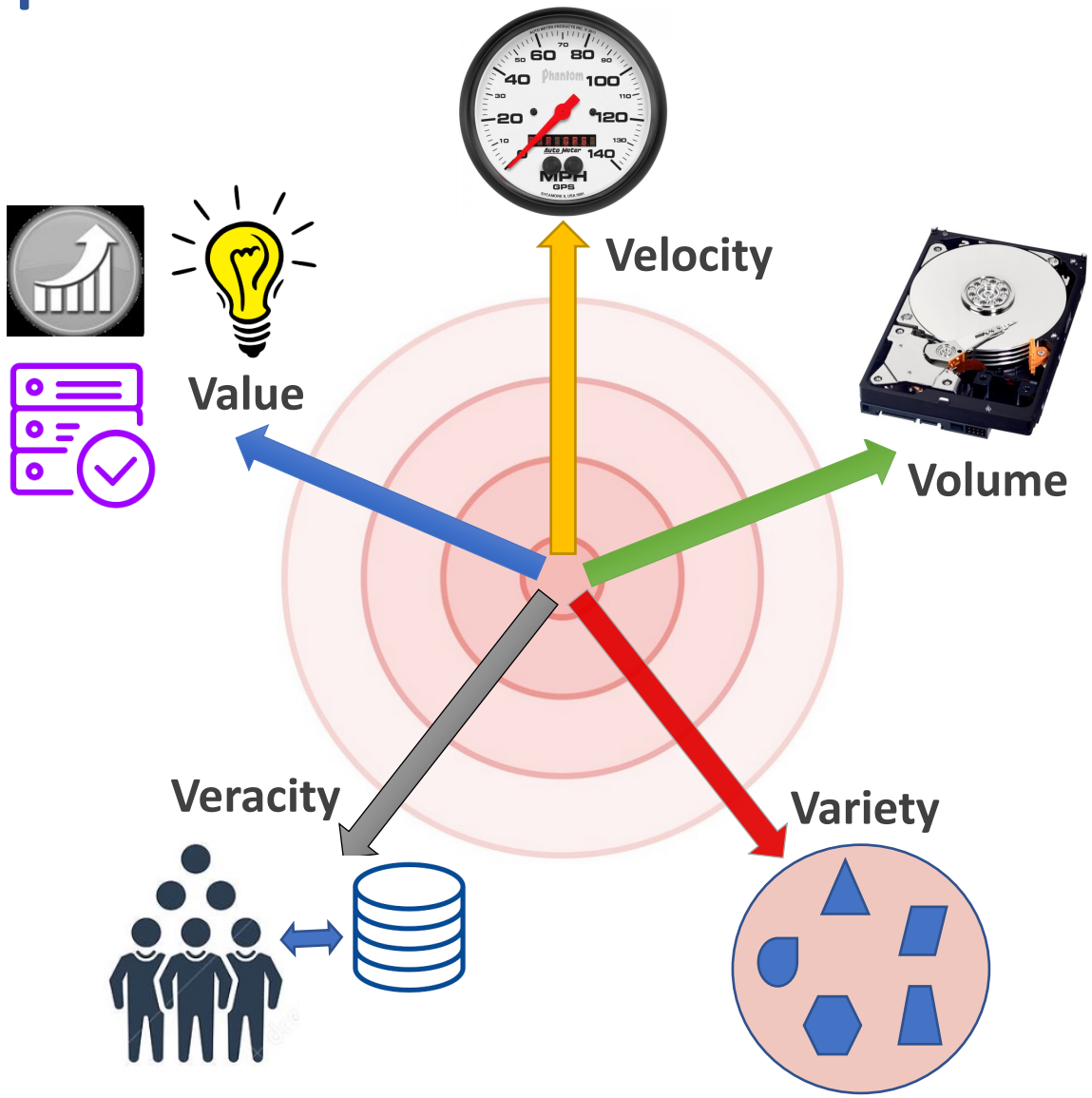
- Velocity: Rate at which data is generated
- Volume: The extreme size of the data
- Variety: Diverse types of structured and unstructured data
- Veracity: Quality and truthfulness of the data
- Value: Contributes to informed decision making



Big Data Characterization

- The “5Vs” of Big Data

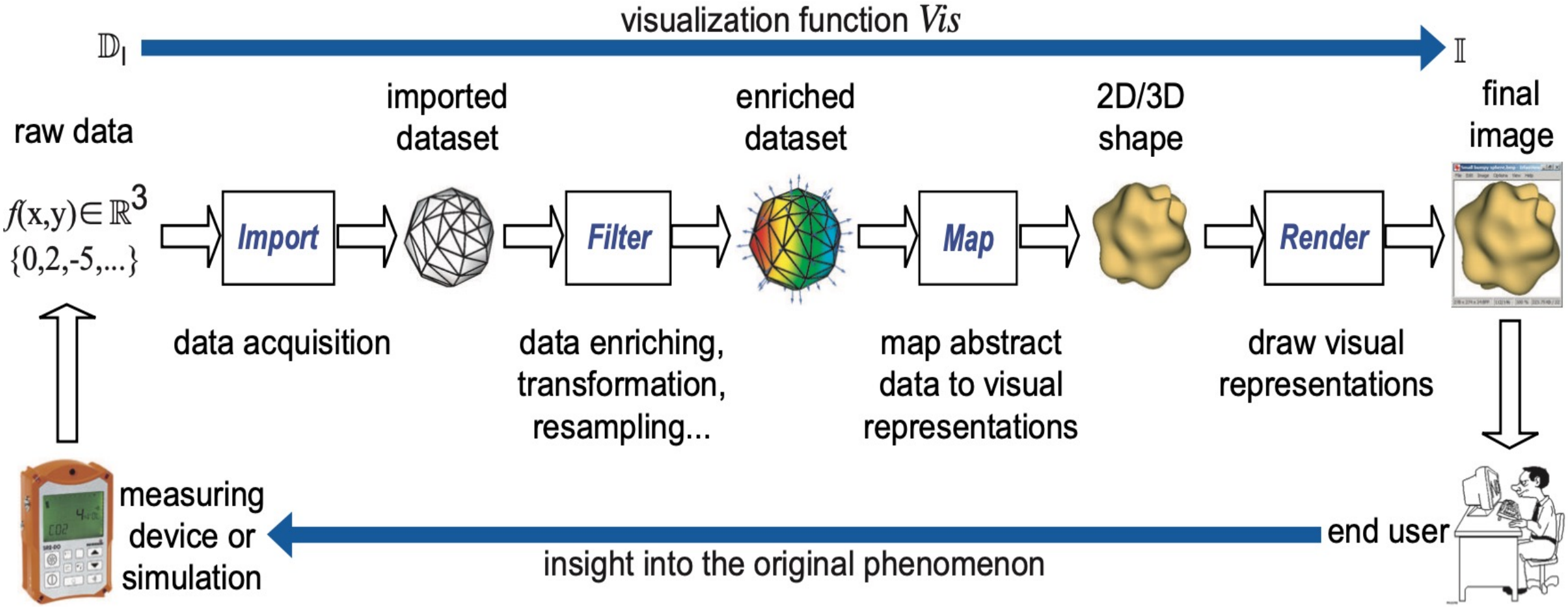
- Velocity: Rate at which data is generated
- Volume: The extreme size of the data
- Variety: Diverse types of structured and unstructured data
- Veracity: Quality and truthfulness of the data
- Value: Contributes to informed decision making



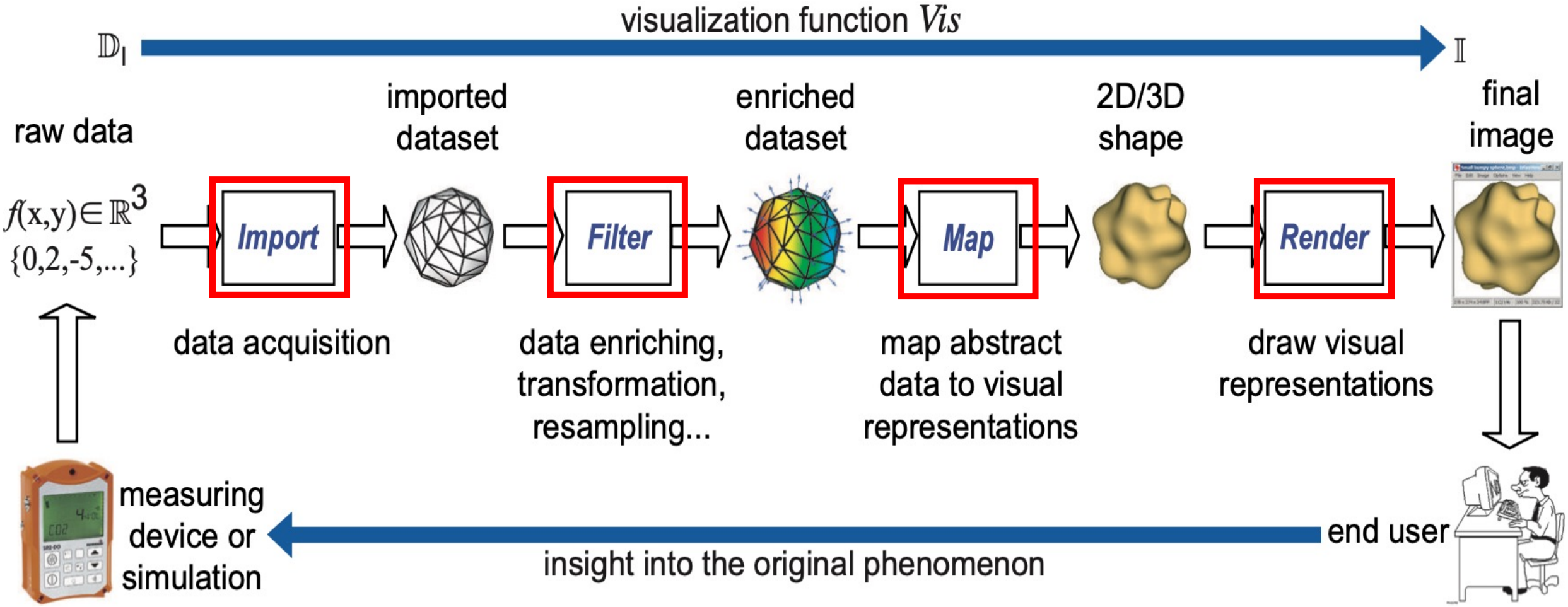
The 6th V: Visualization!

Visualization Pipeline

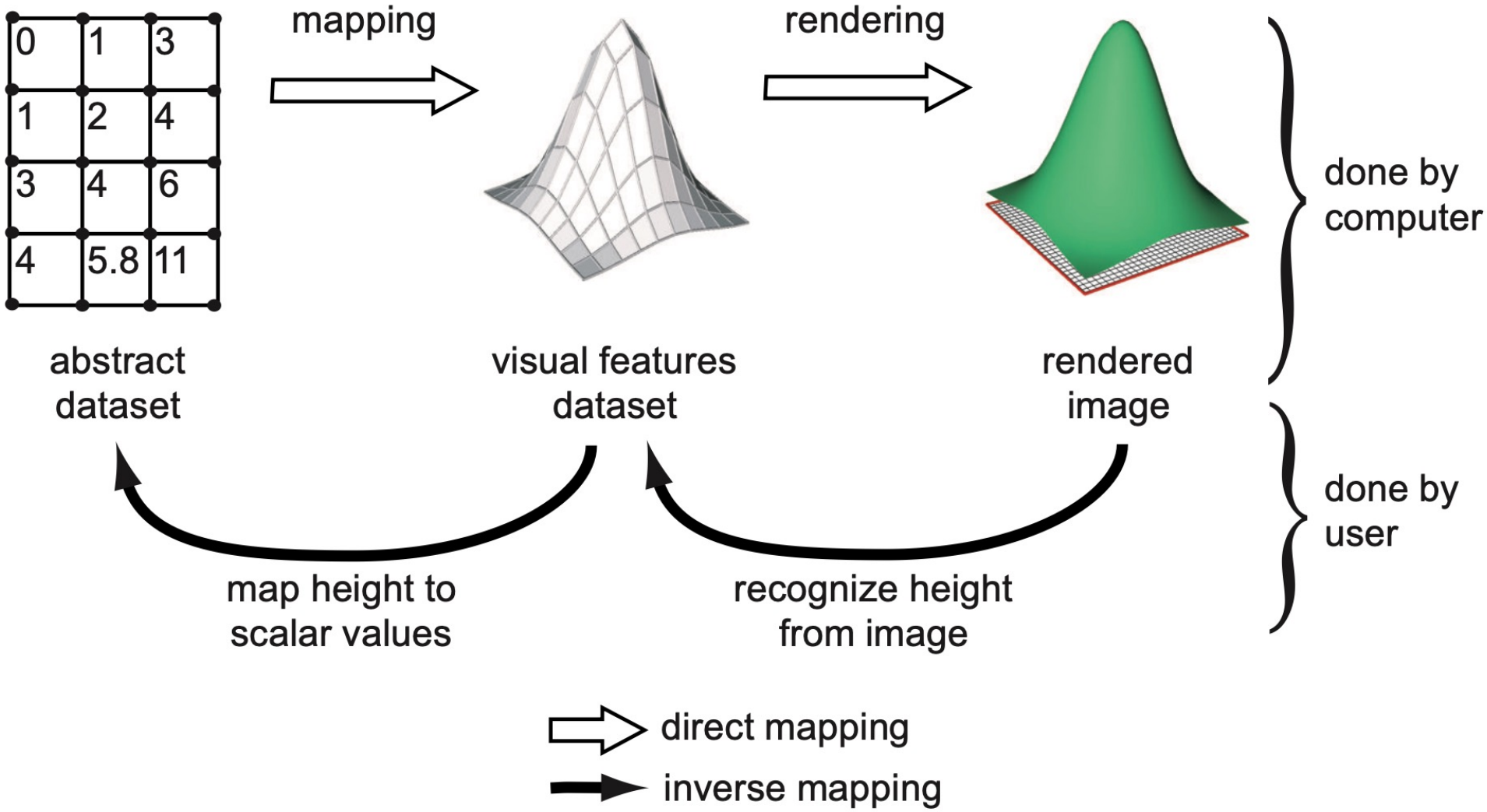
Visualization Pipeline



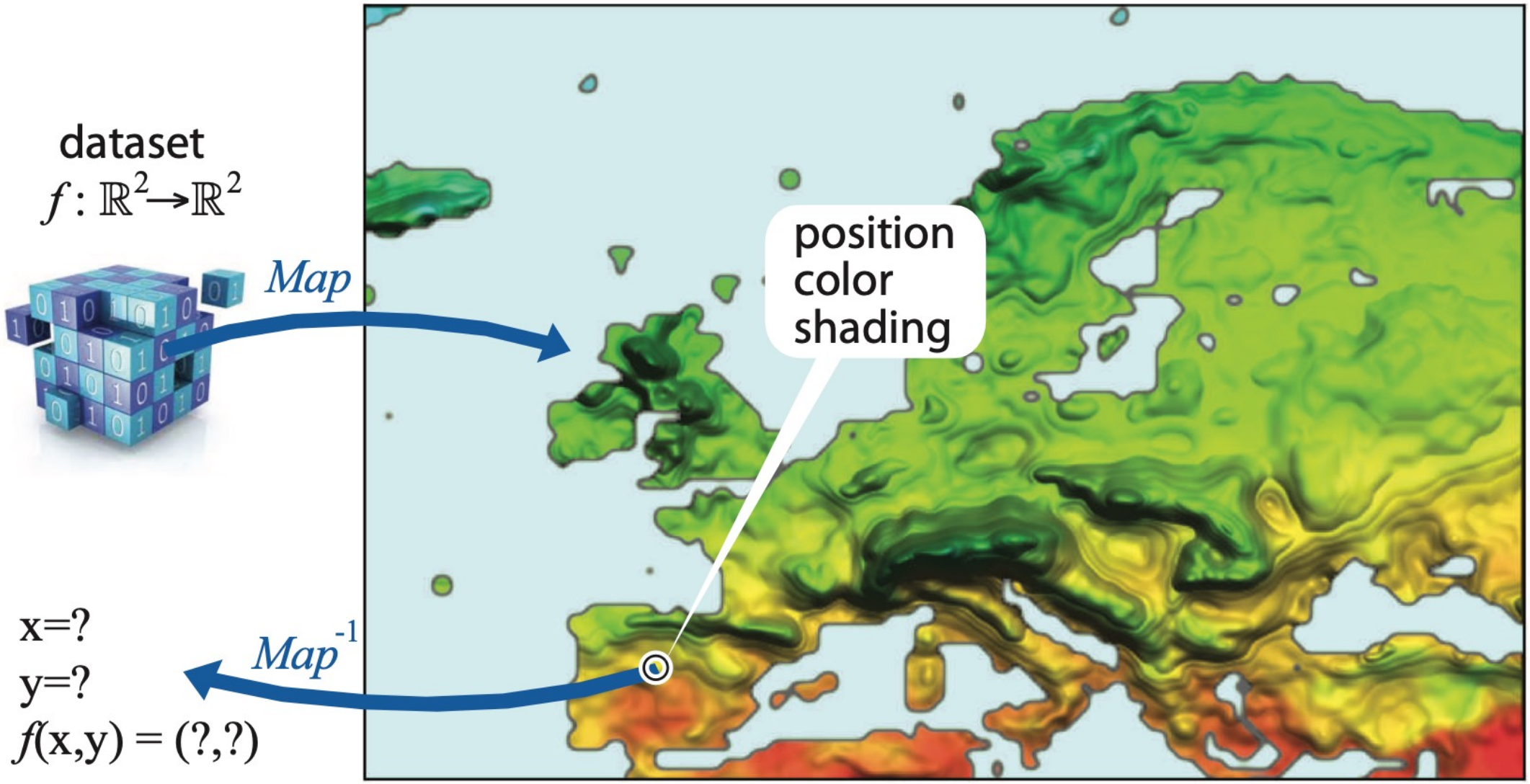
Visualization Pipeline



Direct and Inverse Mapping



Direct and Inverse Mapping



Data Representation & Scientific Data Model

Data Representation

- Scientific data is continuous in nature
  - Measure of physical quantities that are studied by various disciplines
- Mathematically, a continuous data is defined as a function

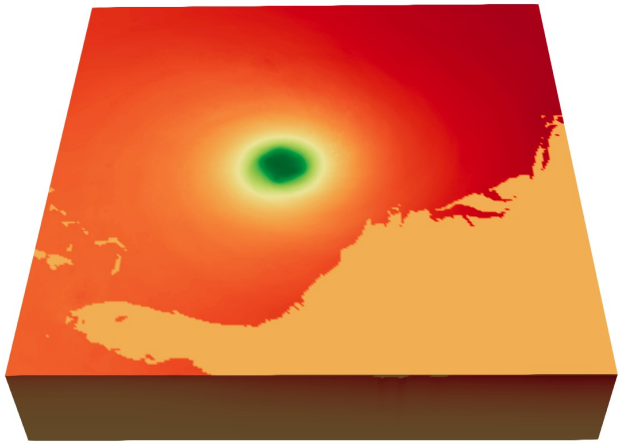
$$f : \mathrm{D} \rightarrow \mathrm{C}$$

- In practice, such data is sampled in a discrete form in computers for representation, manipulation, analysis, and visualization

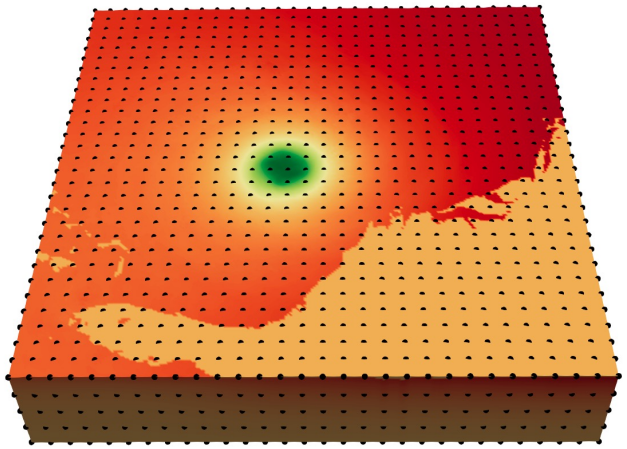
Scientific Dataset

- A scientific dataset is a related collection of data with a spatial context
- In practice, we deal with discrete datasets sampled from a continuous domain for digital representation

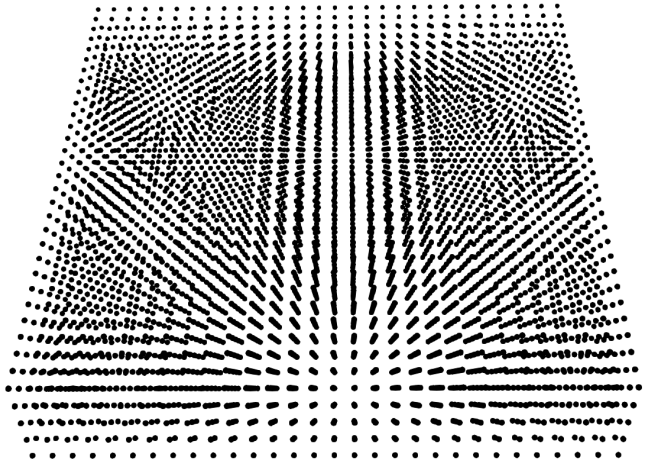
Scientific Dataset: Example



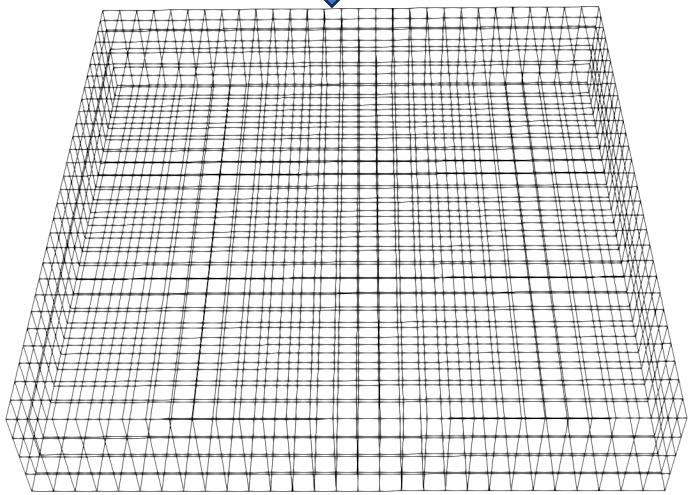
Continuous domain



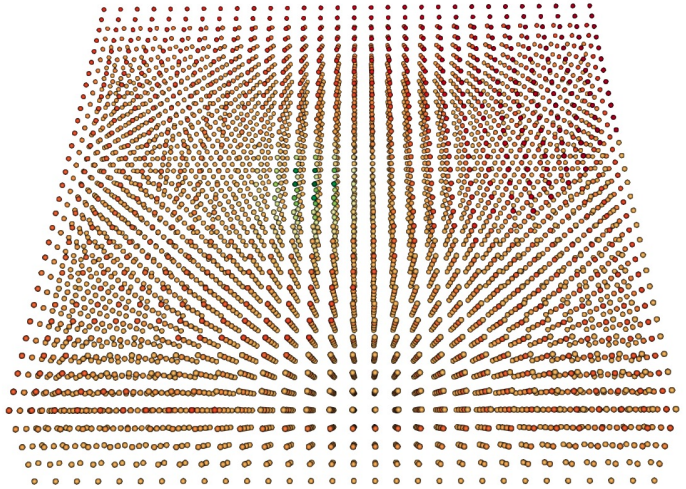
Discrete samples



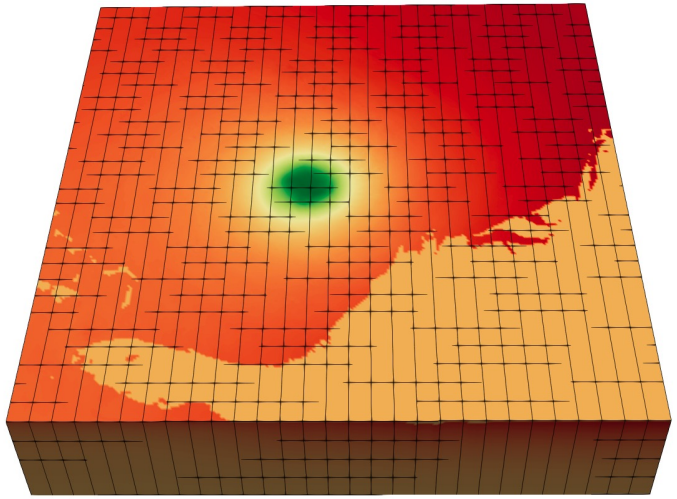
Discrete samples: Grid points



Grid points connected to form cells

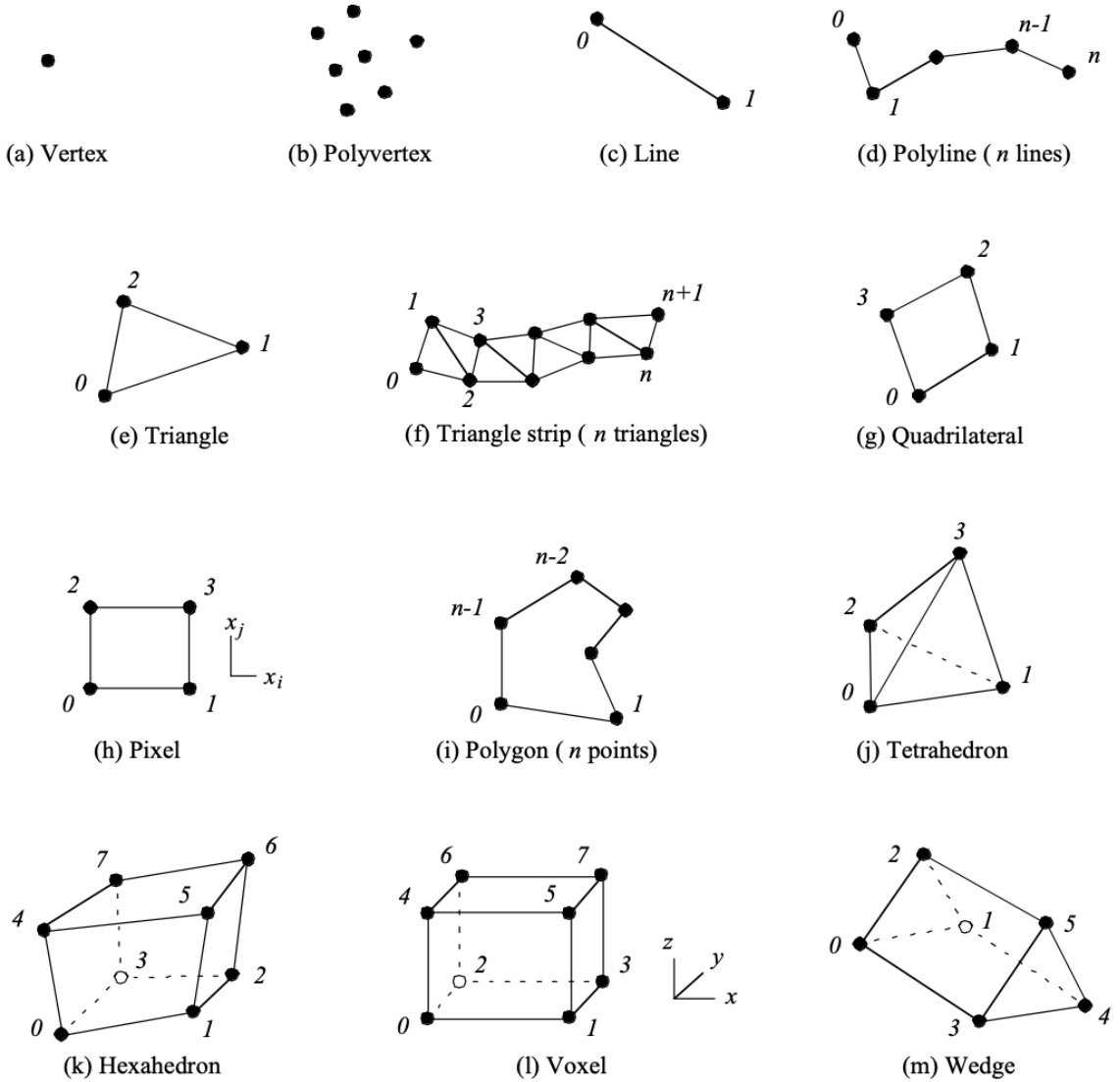


Data values are given at grid points

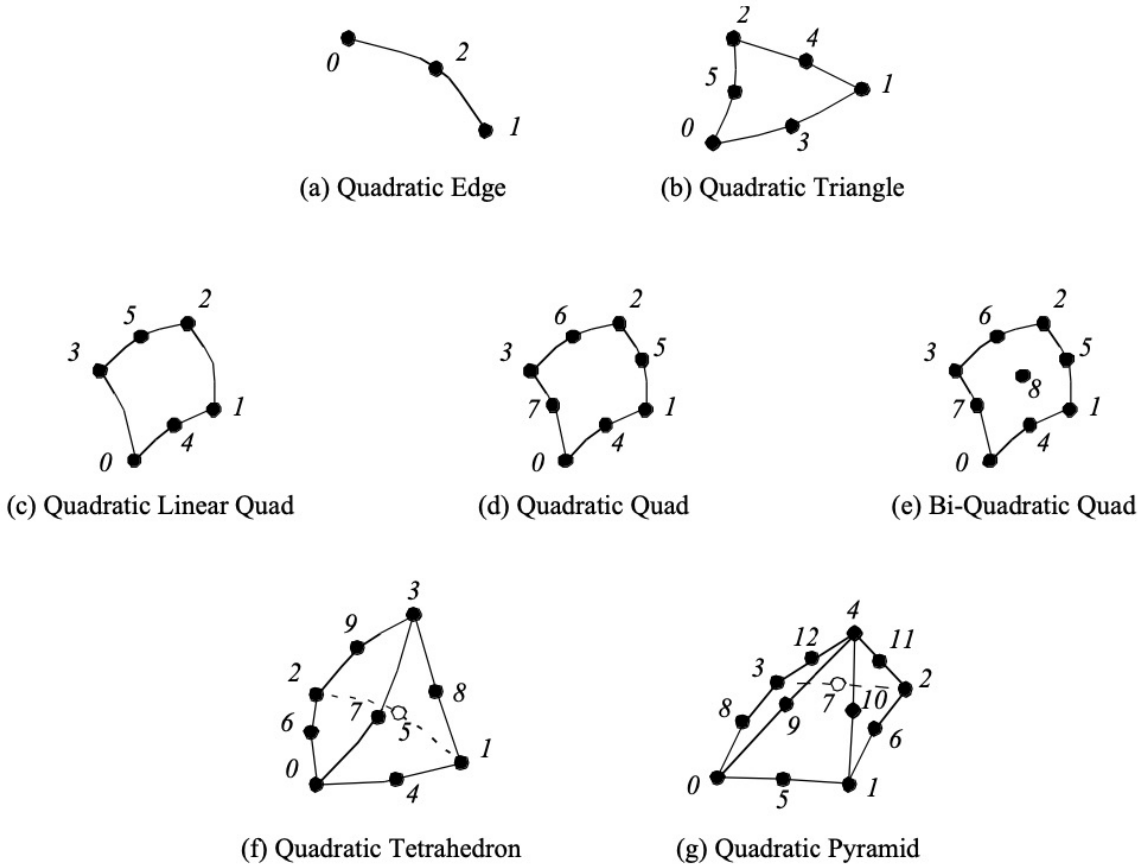


Reconstructed data from grid points

Scientific Dataset: Various Cell Types



Linear Cell Types

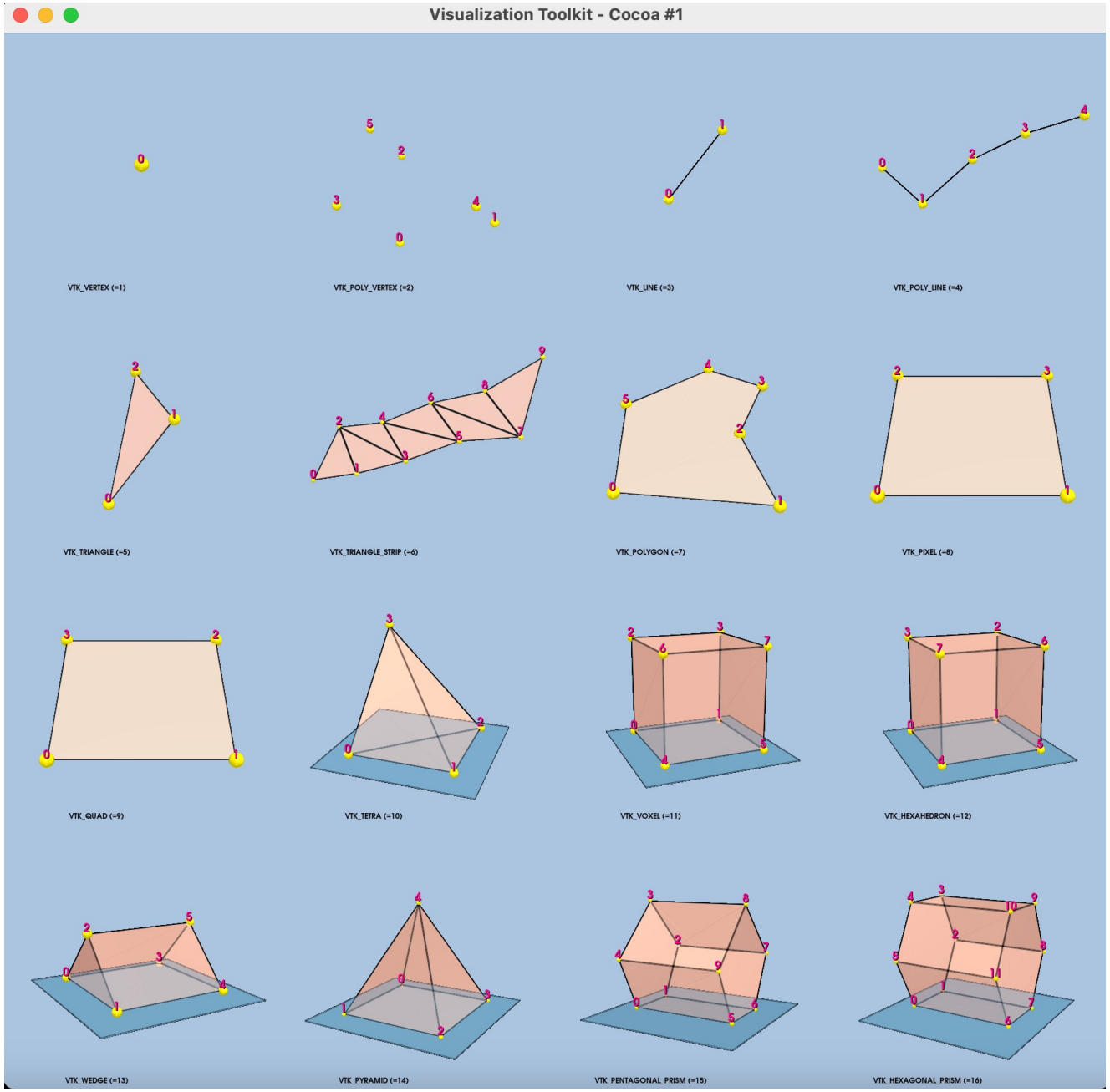


Non-linear Cells Types

Demo Linear Cell

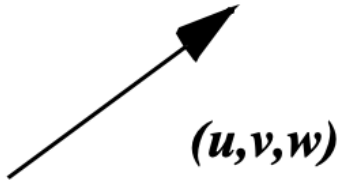
Code:

https://examples.vtk.org/site/Python/GeometricObjects/LinearCellsDemo/

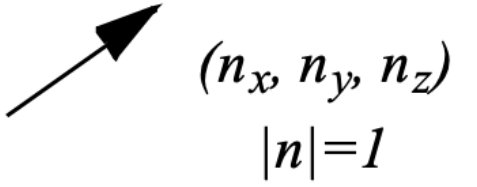


Scientific Dataset: Attribute Data

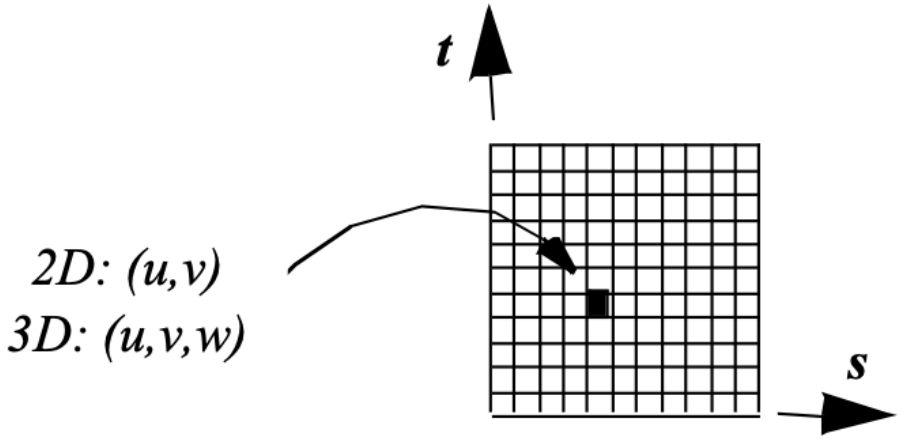
Scalar: single data value



Vector: 3D direction and magnitude



Normal: 3D direction



Texture coordinate: n-dimensional index into texture map

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

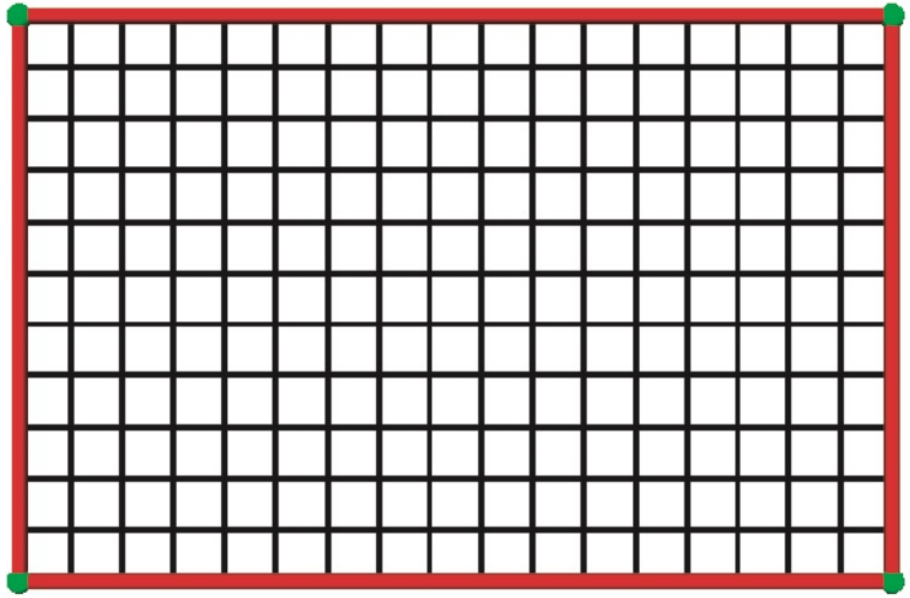
Tensor: nxn matrix

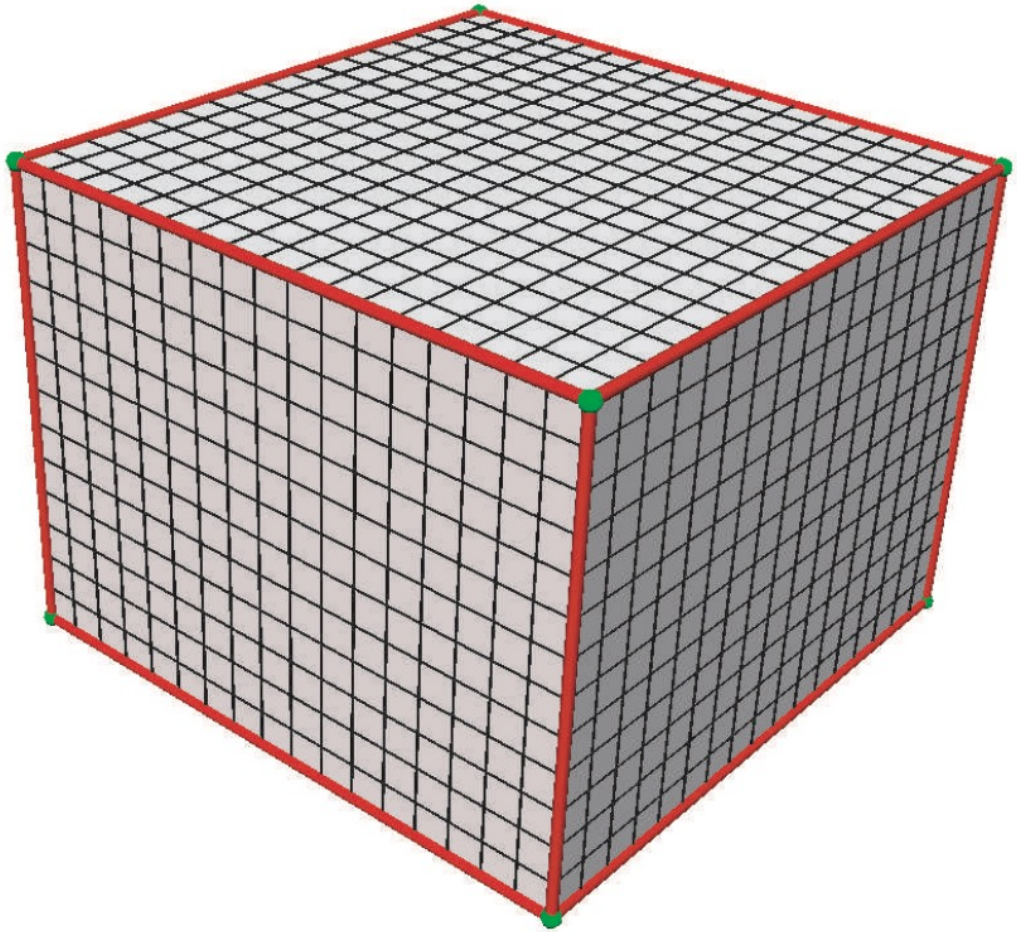
Scientific Dataset: Grid Types

- Uniform (regular) grid
- Rectilinear grid
- Structured grid
- Unstructured grid

Scientific Dataset: Uniform (regular) Grid

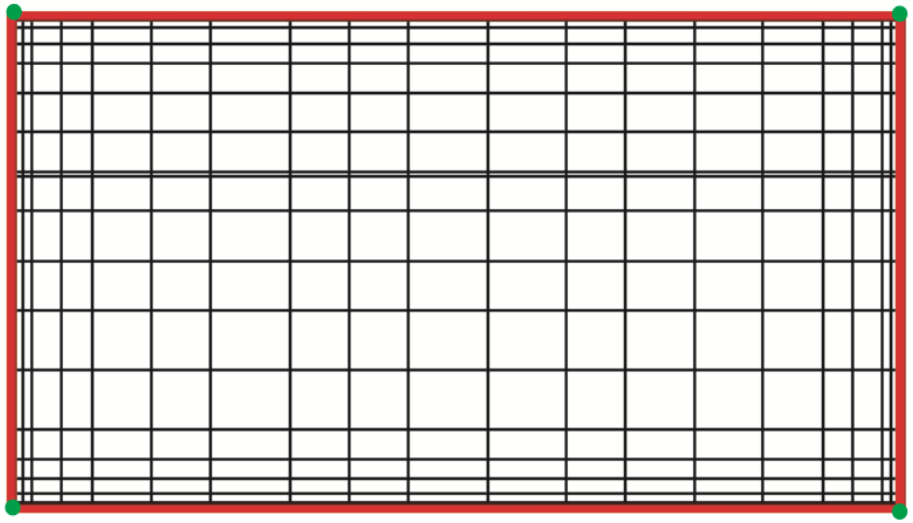
- Axis aligned box
- Sample points are equally spaced

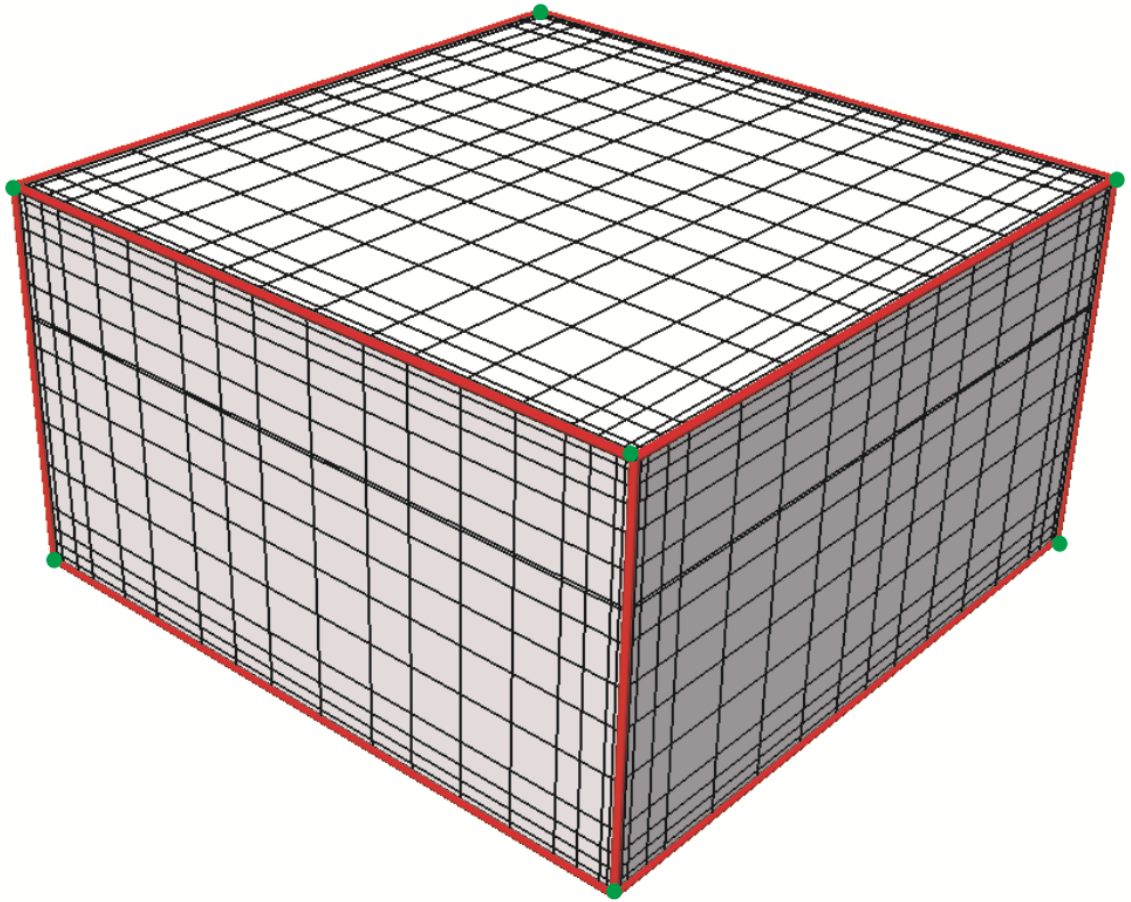




Scientific Dataset: Rectilinear Grid

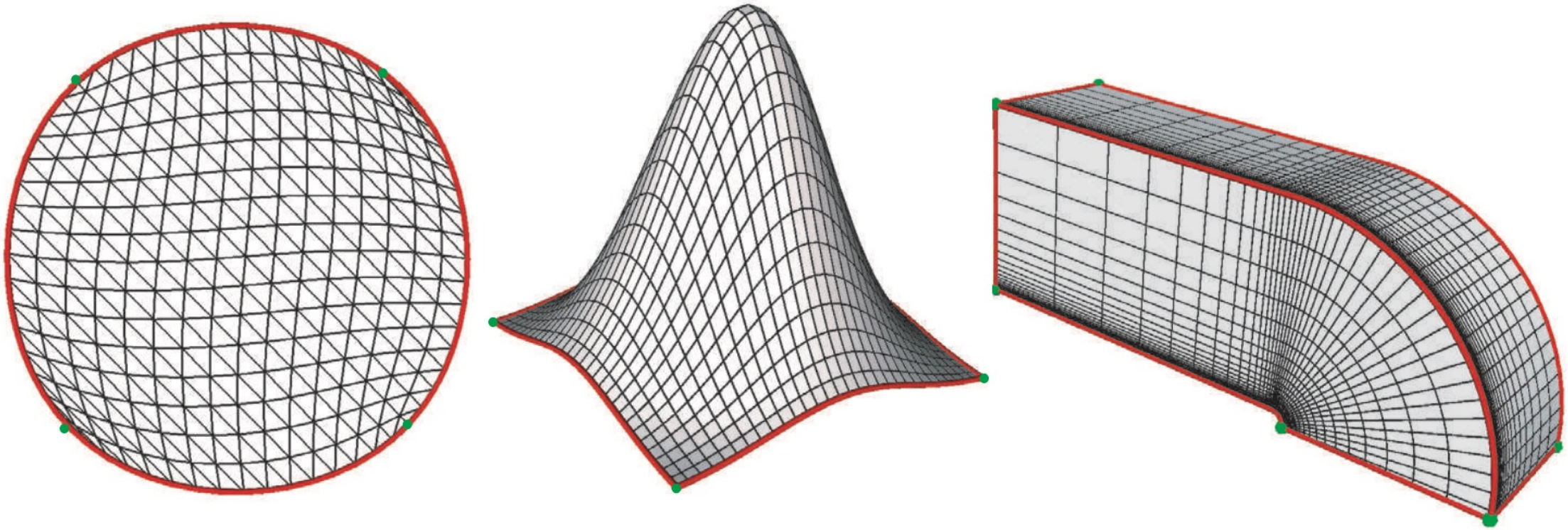
- Axis aligned box
- Sample points are nonequally spaced





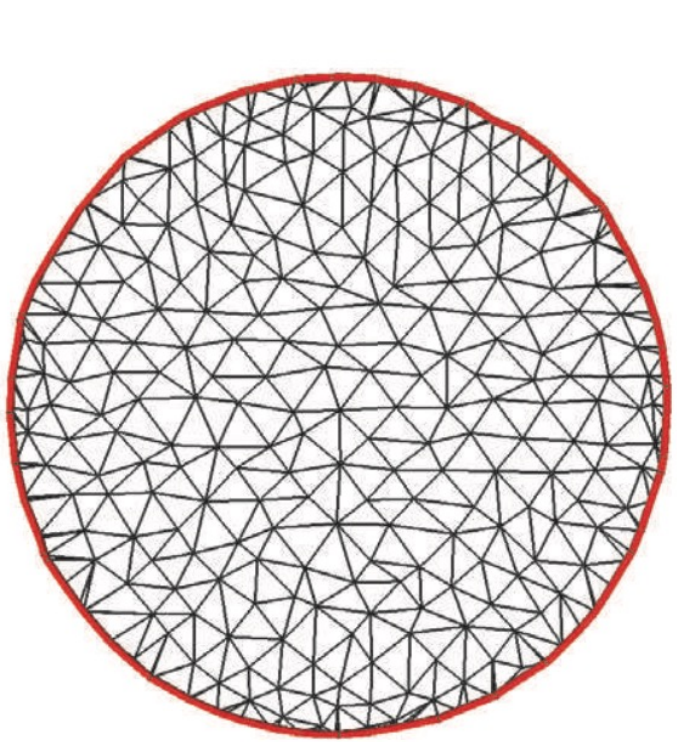
Scientific Dataset: Structured Grid

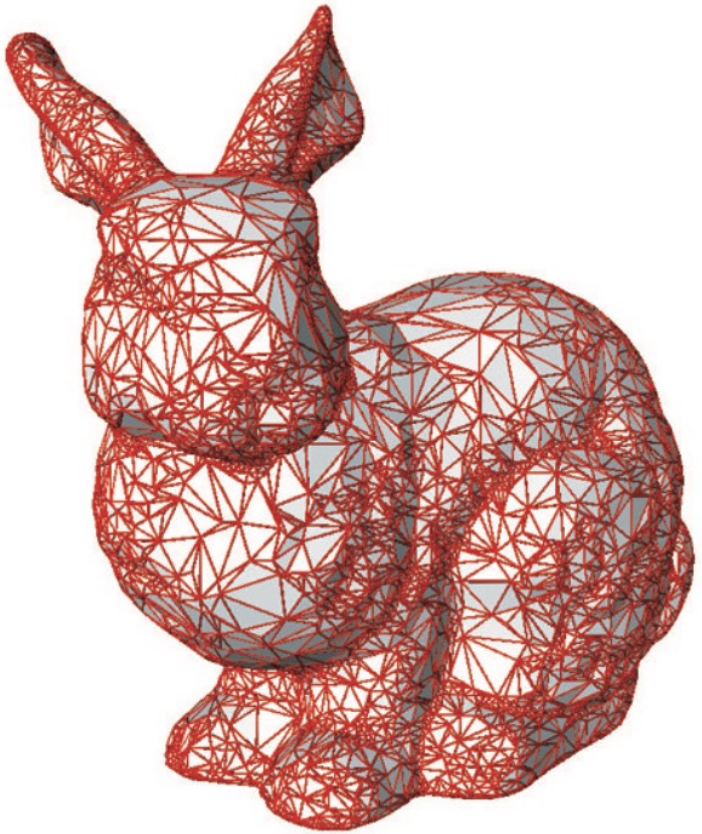
- Allow explicit placement of every sample point
- Yet preserve the matrix-like ordering
- Structured grids can be seen as a deformation of uniform/rectilinear grids



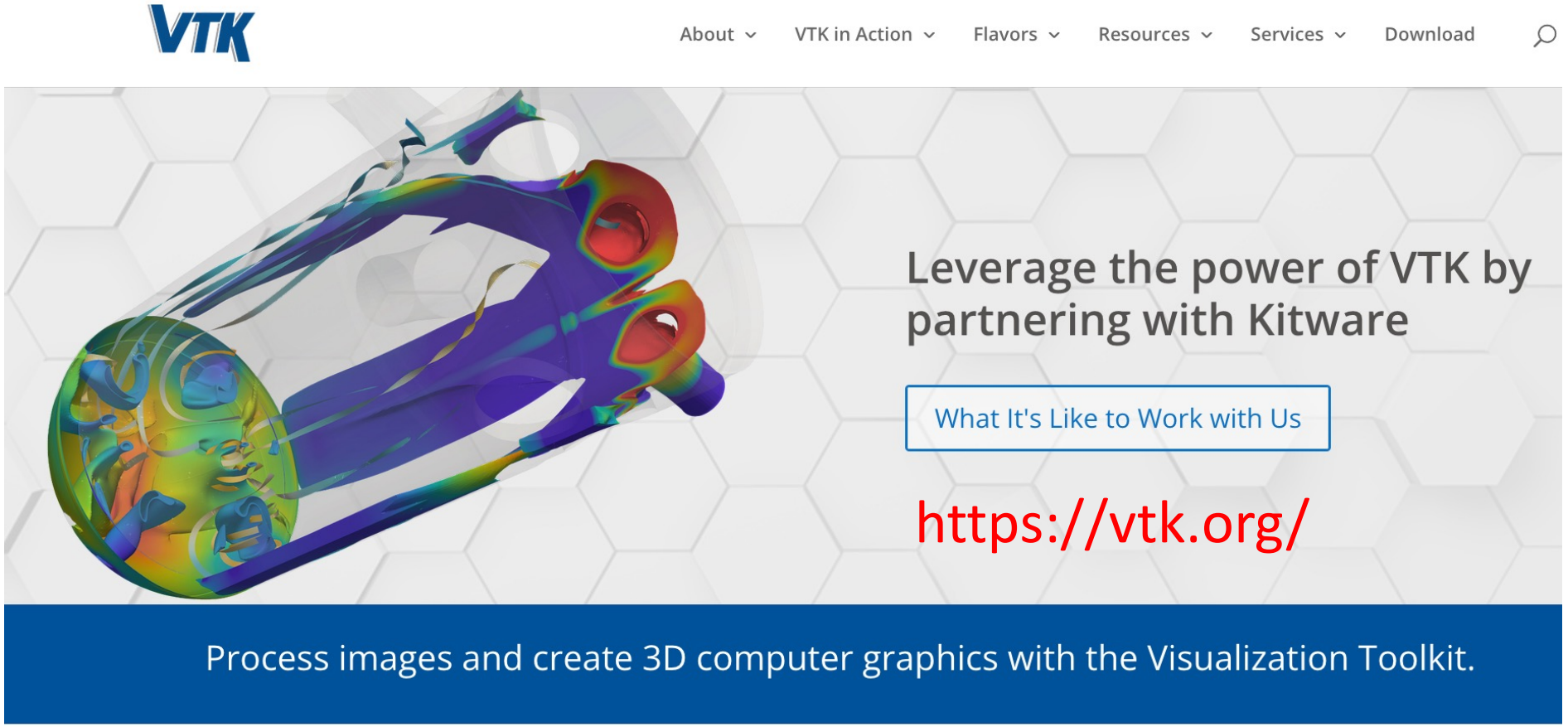
Scientific Dataset: Unstructured Grid

- Allow us to define both the sample points and cells explicitly
- Different cell types can be mixed
- Connectivity is explicitly specified



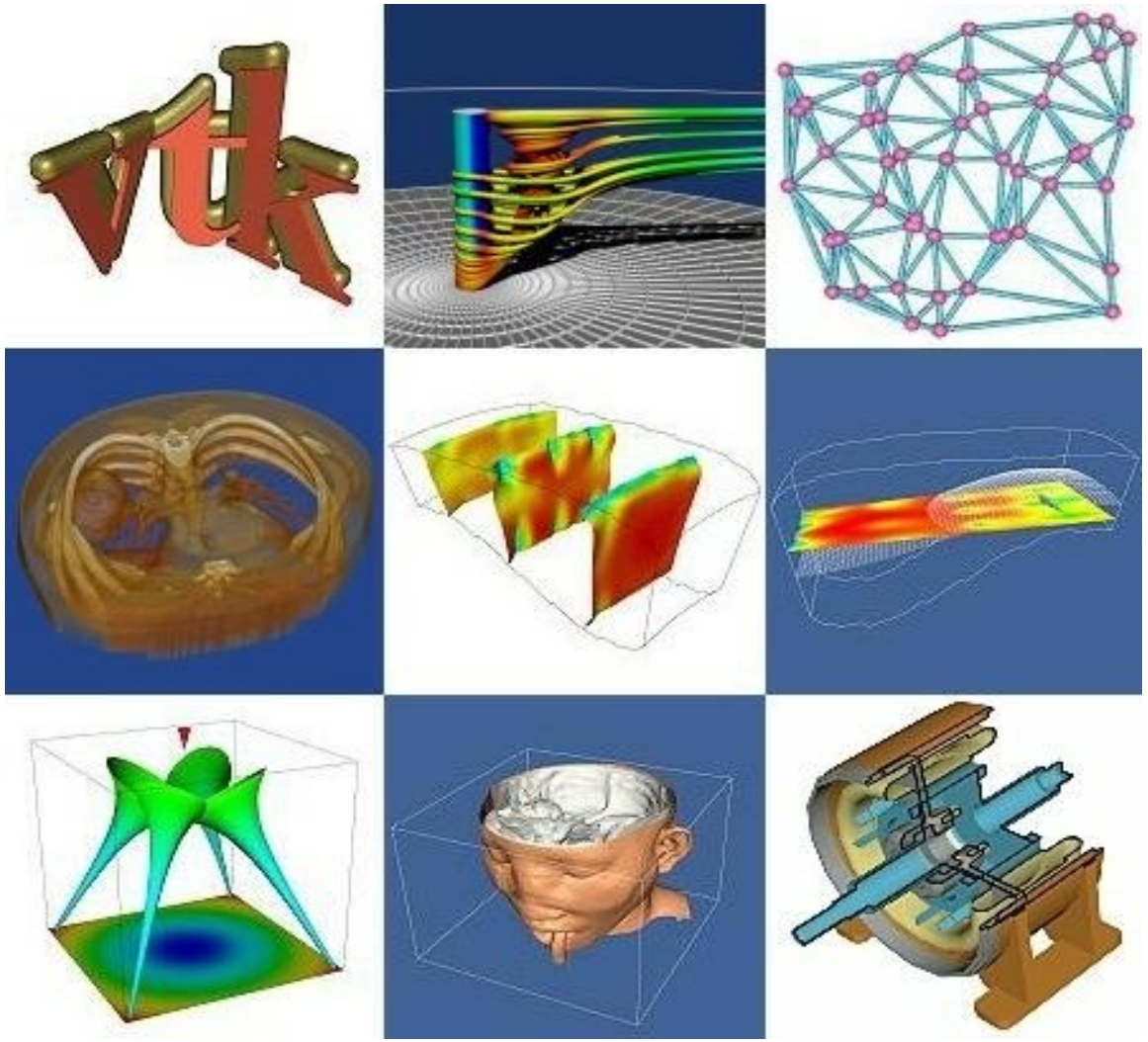


Visualization Toolkit (VTK)



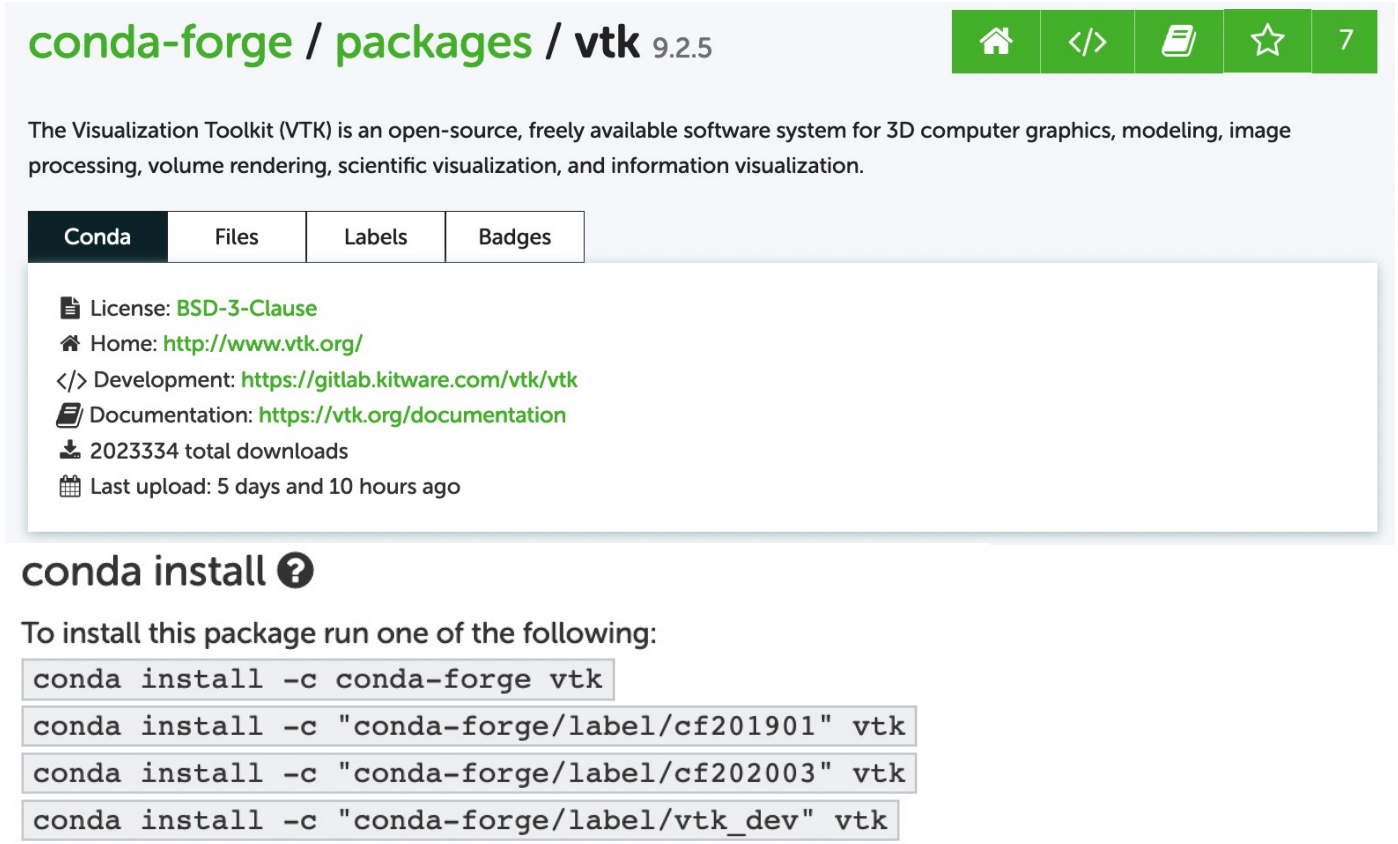
What is VTK?

- VTK: Visualization Toolkit
- An open source, freely available software library for 3D visualization, graphics, and image processing
- Support for hundreds of algorithms
- Object-oriented design with different interpreted language wrappers.



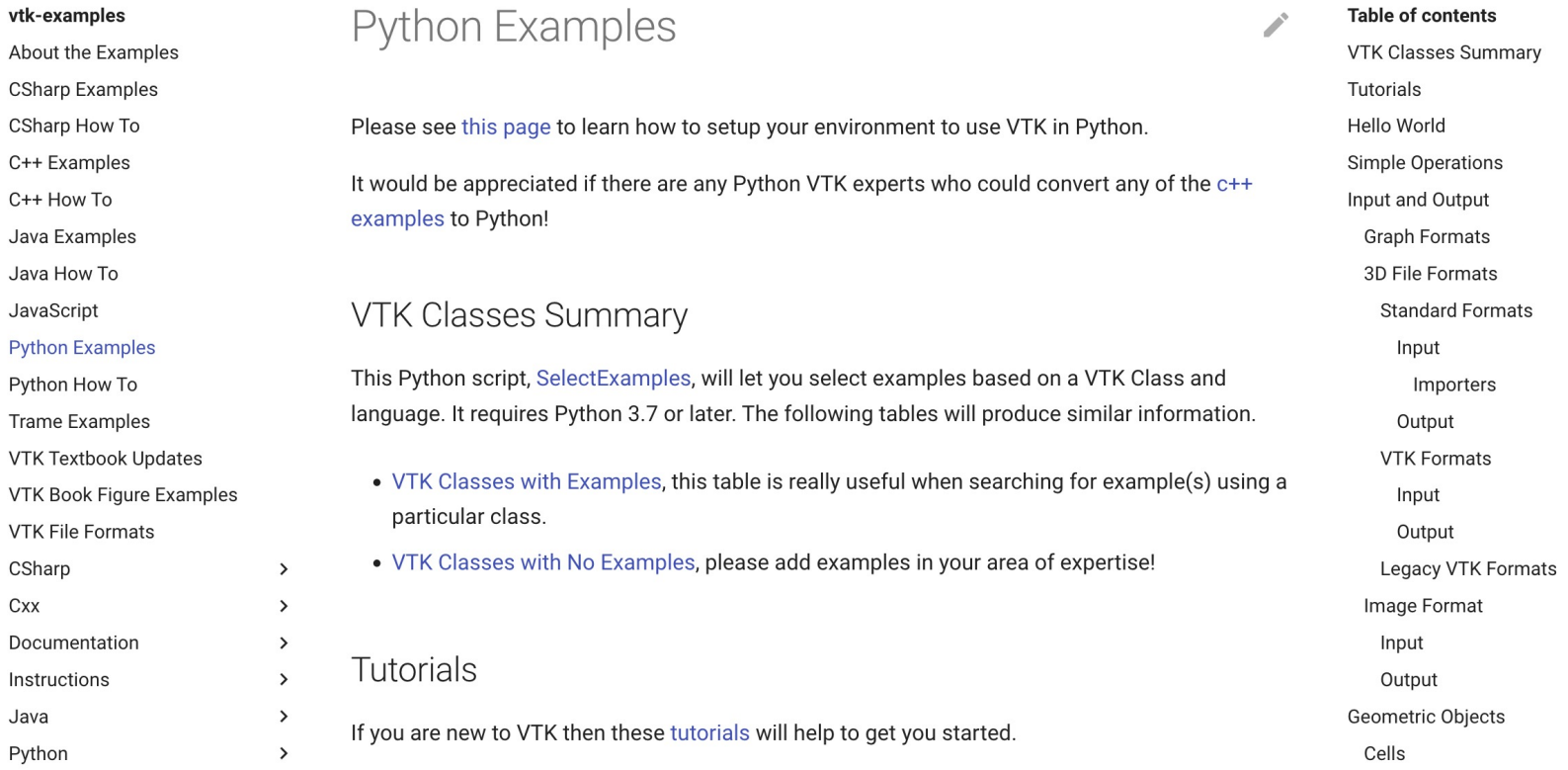
Installing VTK

- https://anaconda.org/conda-forge/vtk
- You can also use pip to install VTK

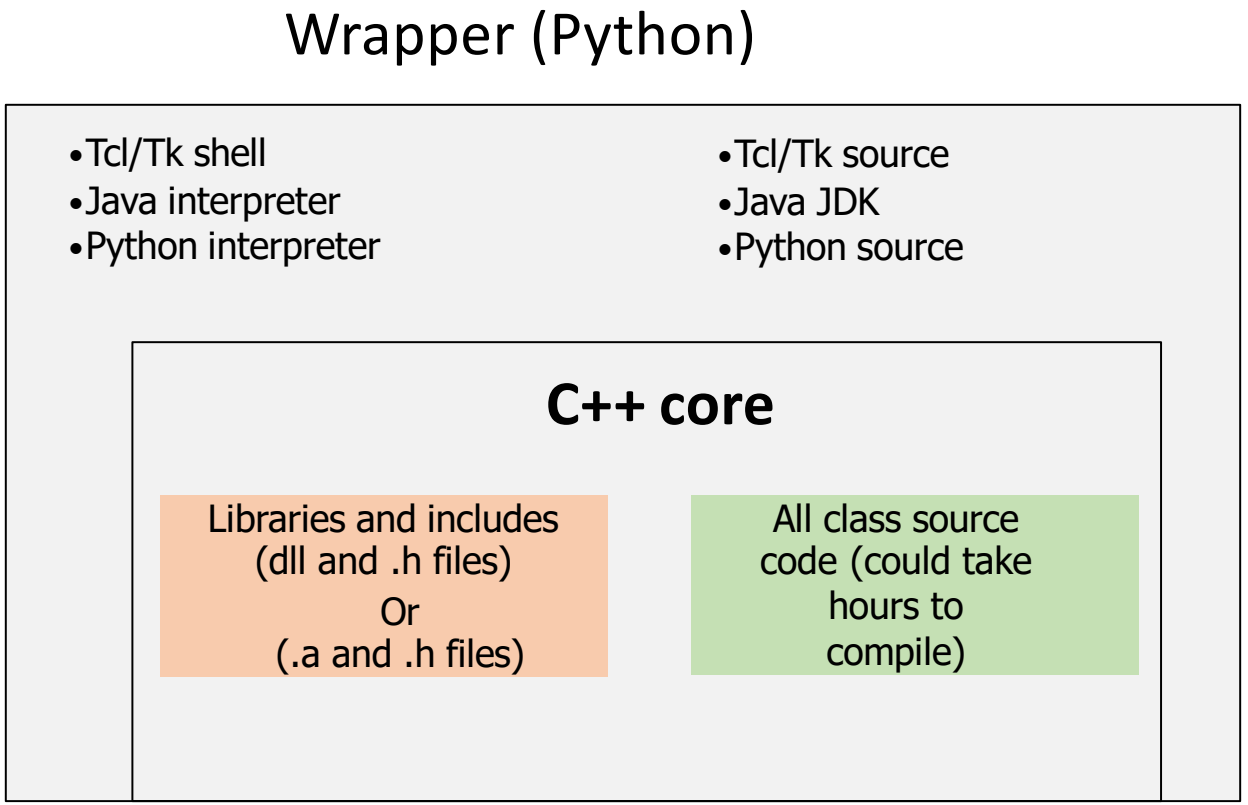


VTK: Resources

- Examples: https://kitware.github.io/vtk-examples/site/Python/
- VTK User's Guide: https://vtk.org/vtk-users-guide/
- VTK Textbook: https://vtk.org/vtk-textbook/



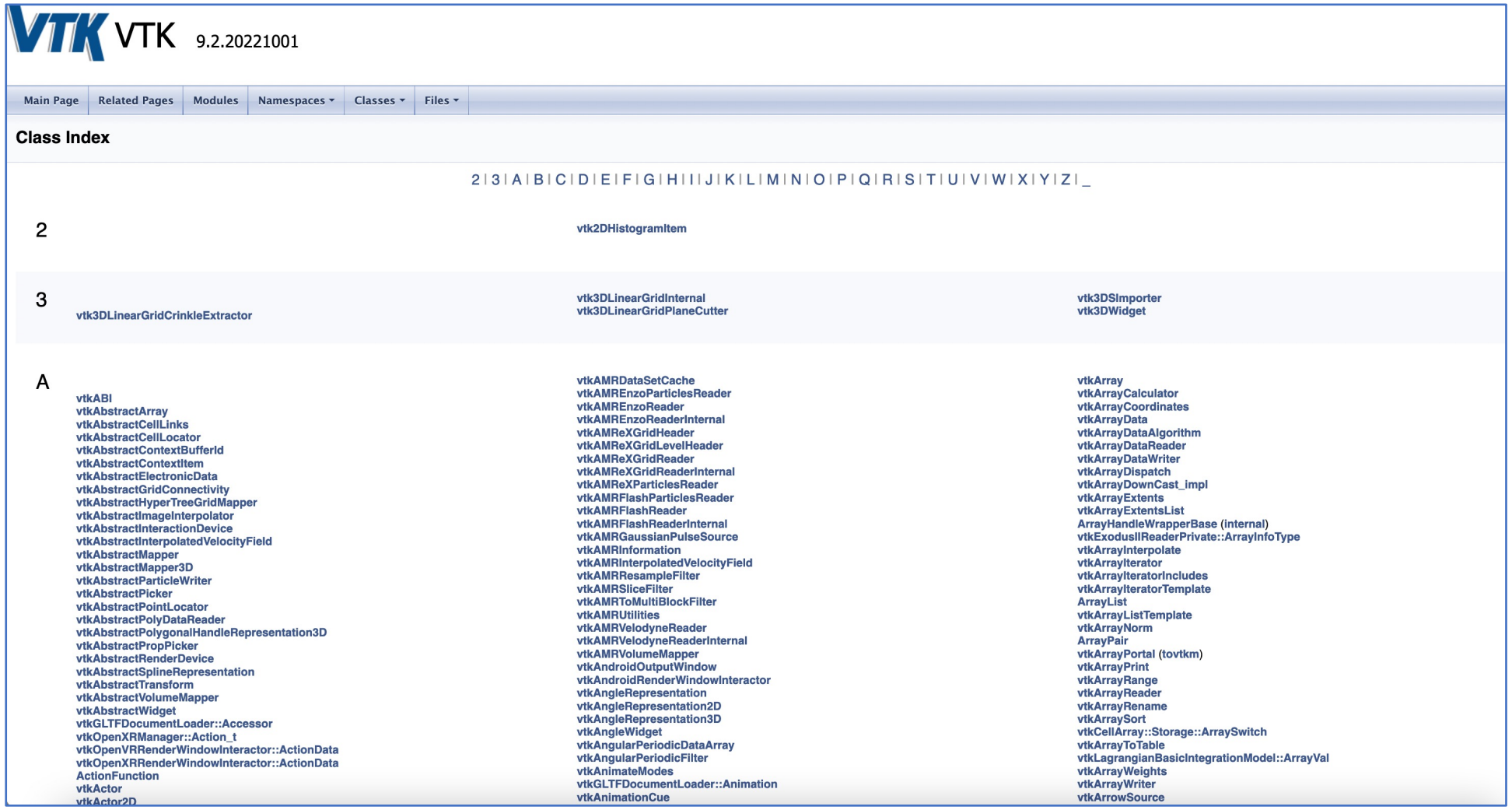
VTK System Architecture



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

Source code Installation: If you want to extend vtk

VTK classes



VTK 9.2.20221001

Main Page | Related Pages | Modules | Namespaces | Classes | Files

Class Index

2 | 3 | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | _

2

vtk2DHistogramItem

3

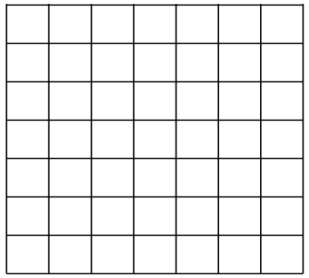
vtk3DLinearGridCrinkleExtractor
vtk3DLinearGridInternal
vtk3DLinearGridPlaneCutter
vtk3DSImporter
vtk3DWidget

A

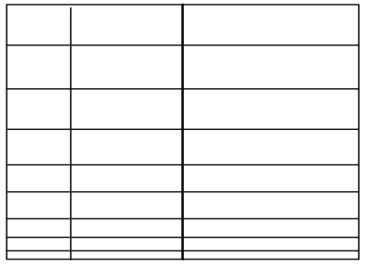
vtkABI
vtkAbstractArray
vtkAbstractCellLinks
vtkAbstractCellLocator
vtkAbstractContextBufferId
vtkAbstractContextItem
vtkAbstractElectronicData
vtkAbstractGridConnectivity
vtkAbstractHyperTreeGridMapper
vtkAbstractImageInterpolator
vtkAbstractInteractionDevice
vtkAbstractInterpolatedVelocityField
vtkAbstractMapper
vtkAbstractMapper3D
vtkAbstractParticleWriter
vtkAbstractPicker
vtkAbstractPointLocator
vtkAbstractPolyDataReader
vtkAbstractPolygonalHandleRepresentation3D
vtkAbstractPropPicker
vtkAbstractRenderDevice
vtkAbstractSplineRepresentation
vtkAbstractTransform
vtkAbstractVolumeMapper
vtkAbstractWidget
vtkGLTFDocumentLoader::Accessor
vtkOpenXRManager::Action_t
vtkOpenVRRenderWindowInteractor::ActionData
vtkOpenXRRenderWindowInteractor::ActionData
ActionFunction
vtkActor
vtkActor2D
vtkAMRDataSetCache
vtkAMREnzoParticlesReader
vtkAMREnzoReader
vtkAMREnzoReaderInternal
vtkAMReXGridHeader
vtkAMReXGridLevelHeader
vtkAMReXGridReader
vtkAMReXGridReaderInternal
vtkAMReXParticlesReader
vtkAMRFlashParticlesReader
vtkAMRFlashReader
vtkAMRFlashReaderInternal
vtkAMRGaussianPulseSource
vtkAMRInformation
vtkAMRInterpolatedVelocityField
vtkAMRResampleFilter
vtkAMRSliceFilter
vtkAMRToMultiBlockFilter
vtkAMRUtilities
vtkAMRVelodyneReader
vtkAMRVelodyneReaderInternal
vtkAMRVolumeMapper
vtkAndroidOutputWindow
vtkAndroidRenderWindowInteractor
vtkAngleRepresentation
vtkAngleRepresentation2D
vtkAngleRepresentation3D
vtkAngleWidget
vtkAngularPeriodicDataArray
vtkAngularPeriodicFilter
vtkAnimateModes
vtkGLTFDocumentLoader::Animation
vtkAnimationCue
vtkArray
vtkArrayCalculator
vtkArrayCoordinates
vtkArrayData
vtkArrayDataAlgorithm
vtkArrayDataReader
vtkArrayDataWriter
vtkArrayDispatch
vtkArrayDownCast_impl
vtkArrayExtents
vtkArrayExtentsList
ArrayHandleWrapperBase (internal)
vtkExodusIIReaderPrivate::ArrayInfoType
vtkArrayInterpolate
vtkArrayIterator
vtkArrayIteratorIncludes
vtkArrayIteratorTemplate
ArrayList
vtkArrayListTemplate
vtkArrayNorm
ArrayPair
vtkArrayPortal (tovtkm)
vtkArrayPrint
vtkArrayRange
vtkArrayReader
vtkArrayRename
vtkArraySort
vtkCellArray::Storage::ArraySwitch
vtkArrayToTable
vtkLagrangianBasicIntegrationModel::ArrayVal
vtkArrayWeights
vtkArrayWriter
vtkArrowSource

(https://vtk.org/doc/nightly/html/classes.html)

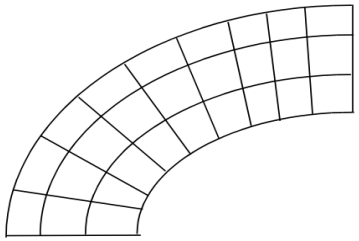
VTK Data Types



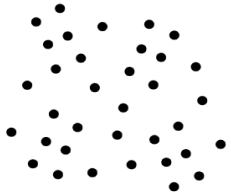
(a) Image Data
(vtkImageData)



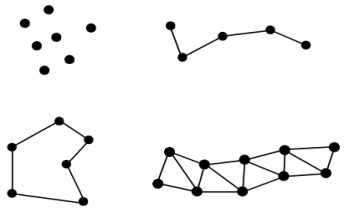
(b) Rectilinear Grid
(vtkRectilinearGrid)



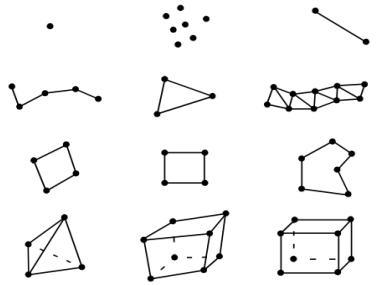
(c) Structured Grid
(vtkStructuredGrid)



(d) Unstructured Points
(use vtkPolyData)

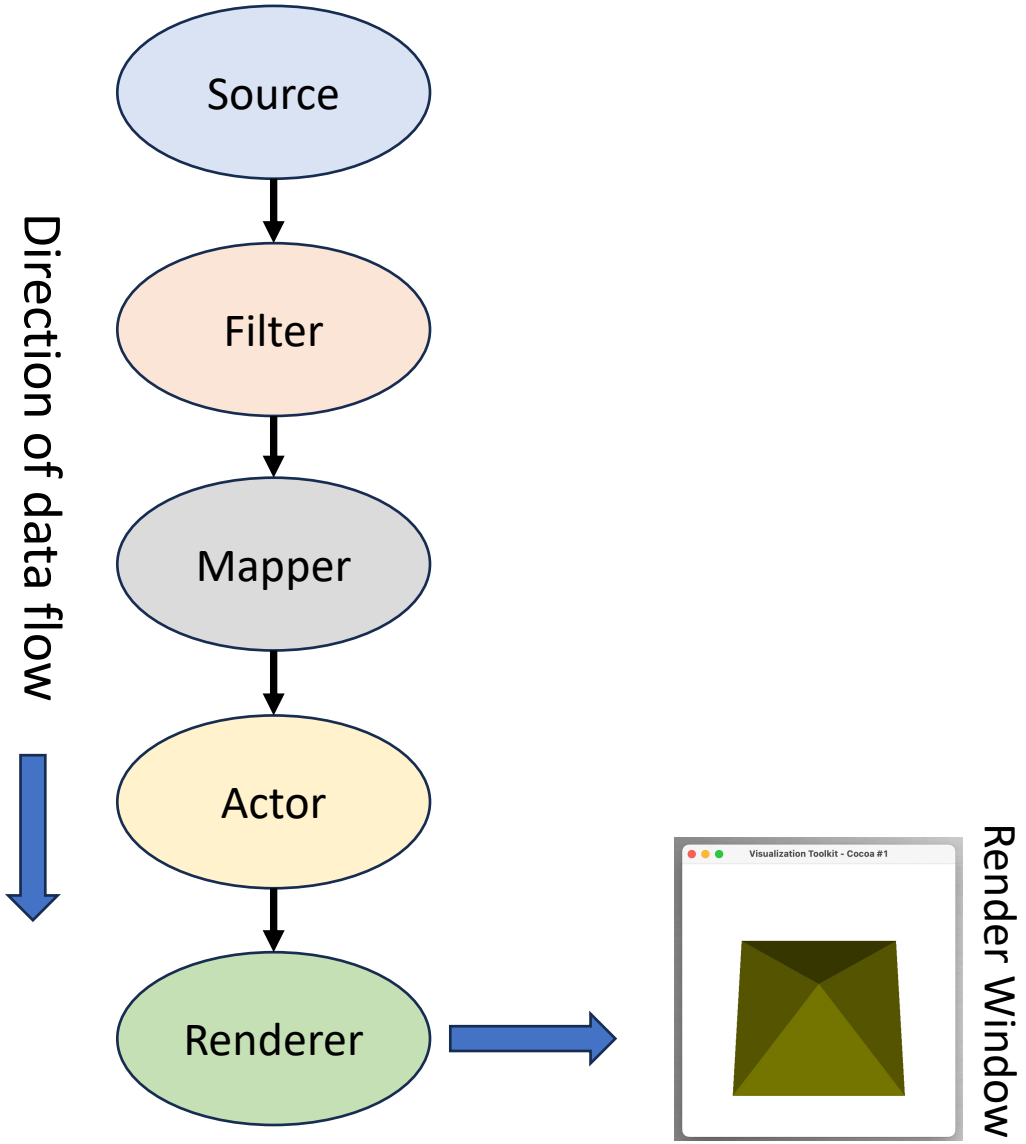


(e) Polygonal Data
(vtkPolyData)

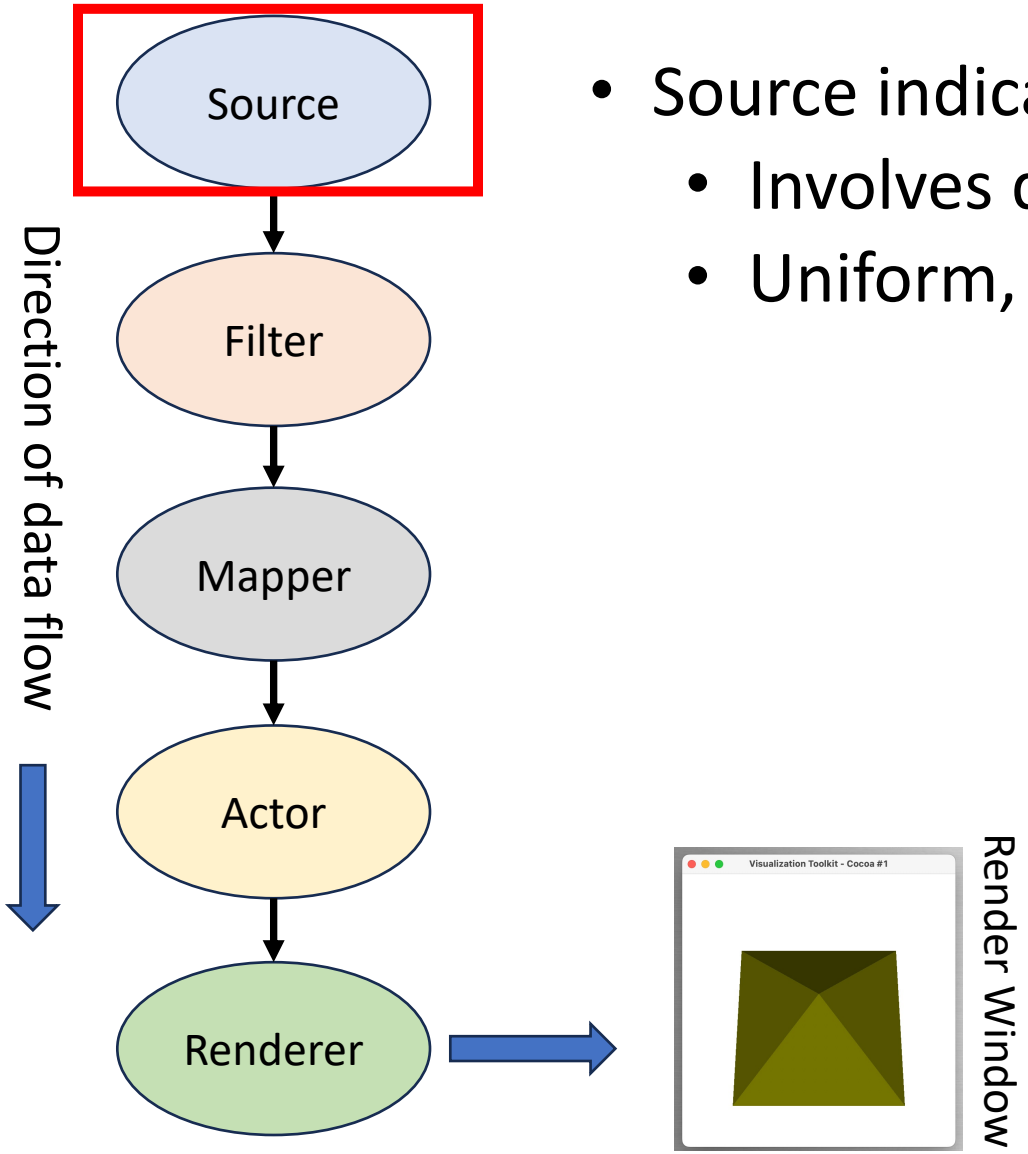


(f) Unstructured Grid
(vtkUnstructuredGrid)

VTK Pipeline Execution

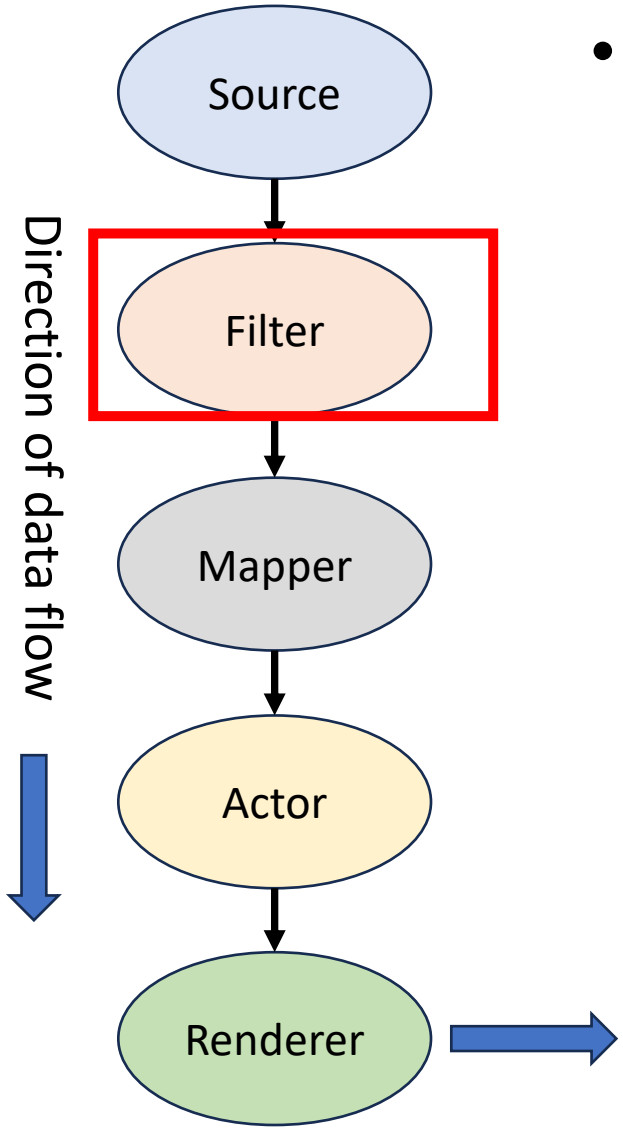


VTK Pipeline Execution: Source

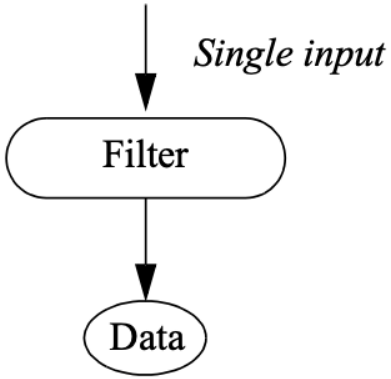


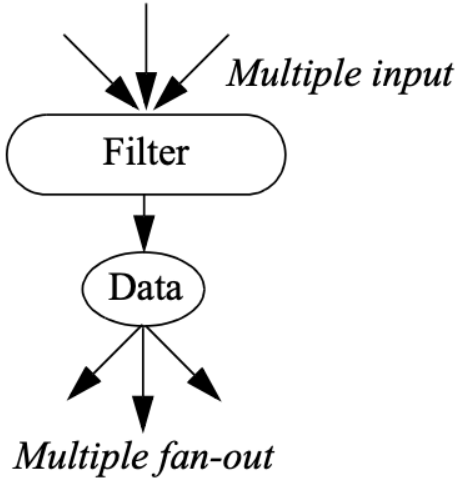
- Source indicates the dataset source
  - Involves data loader of various types
  - Uniform, structured, rectilinear, etc.

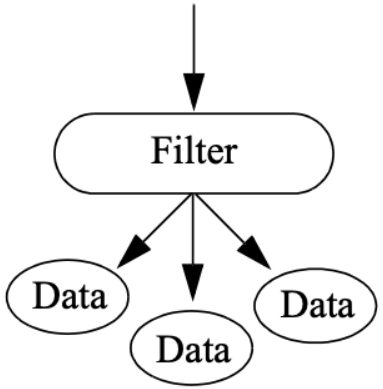
VTK Pipeline Execution: Filter

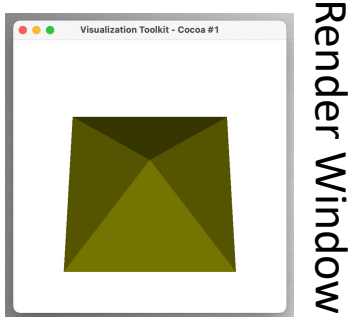


- Filters are another name of algorithms in VTK
  - threshold, connected component, surface extraction, volume render, etc.

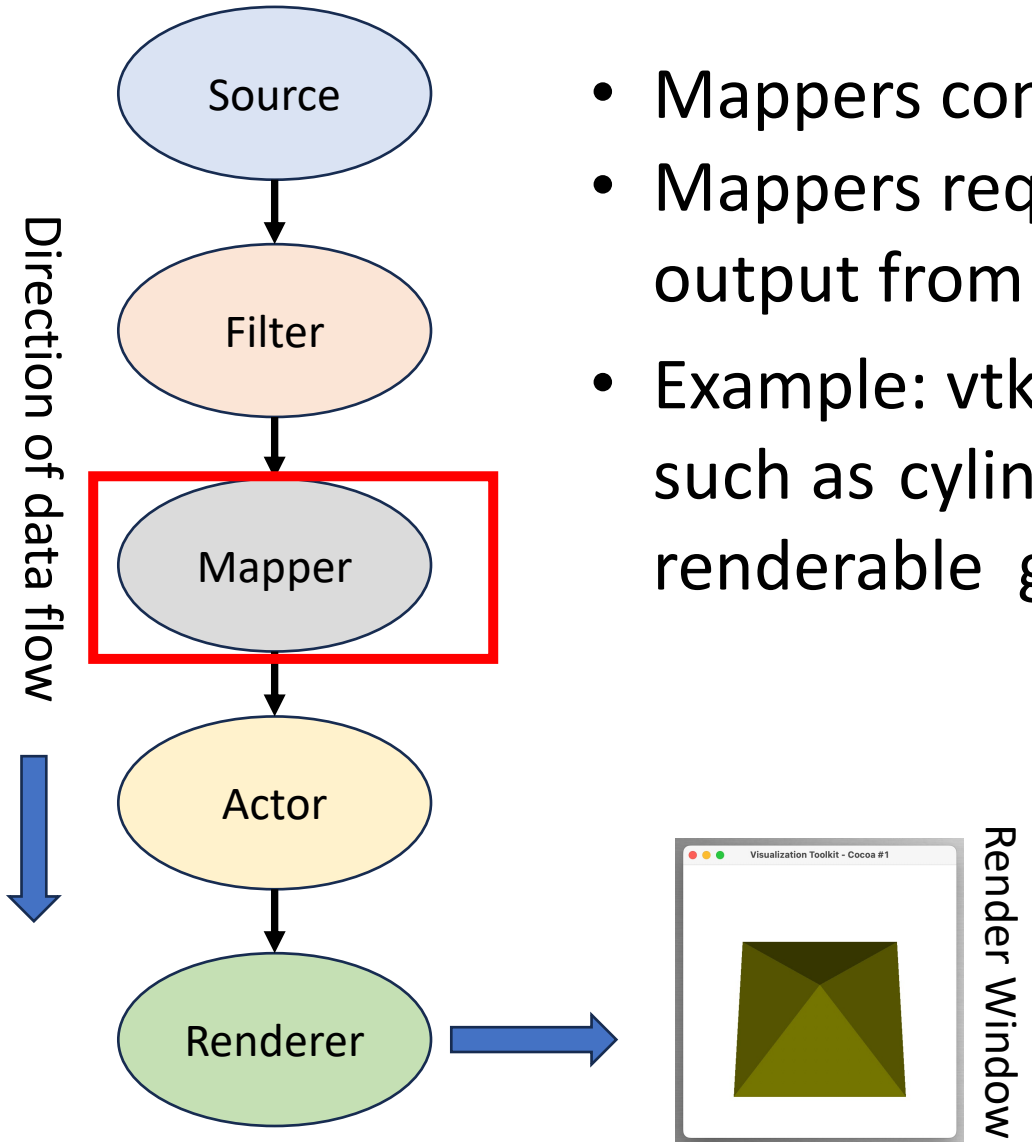






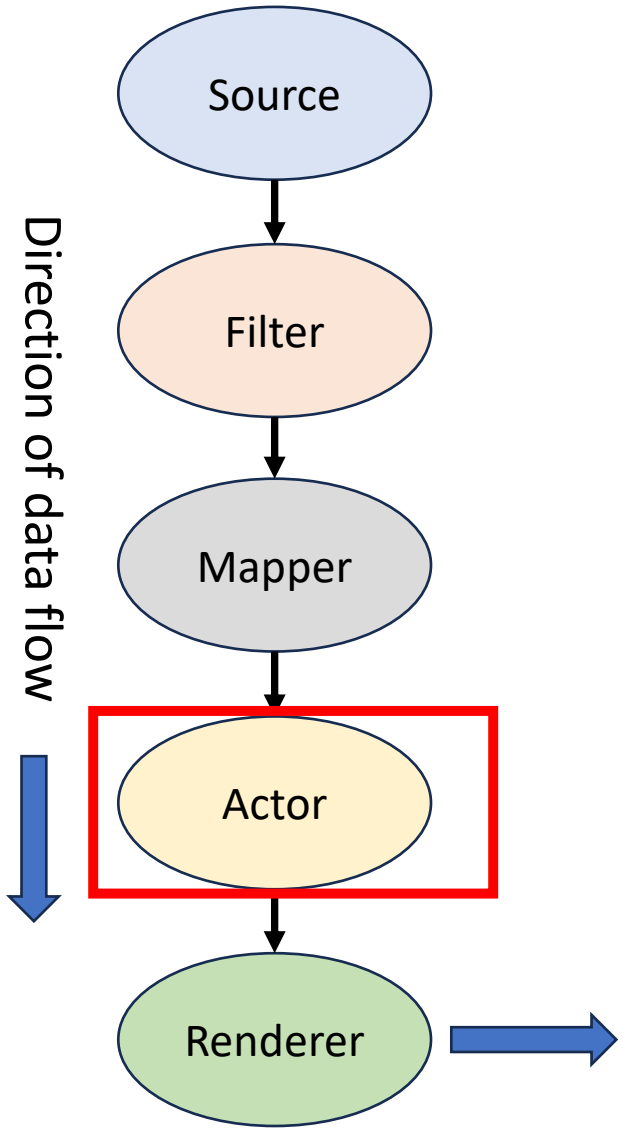


VTK Pipeline Execution: Mapper

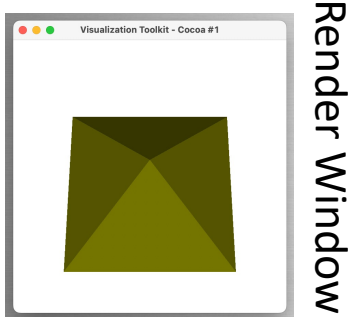


- Mappers convert data into graphical primitives
- Mappers require one or more input data objects, output from Filters
- Example: vtkPolyDataMapper, which takes geometry such as cylinder or cone as input and convert it to renderable geometry

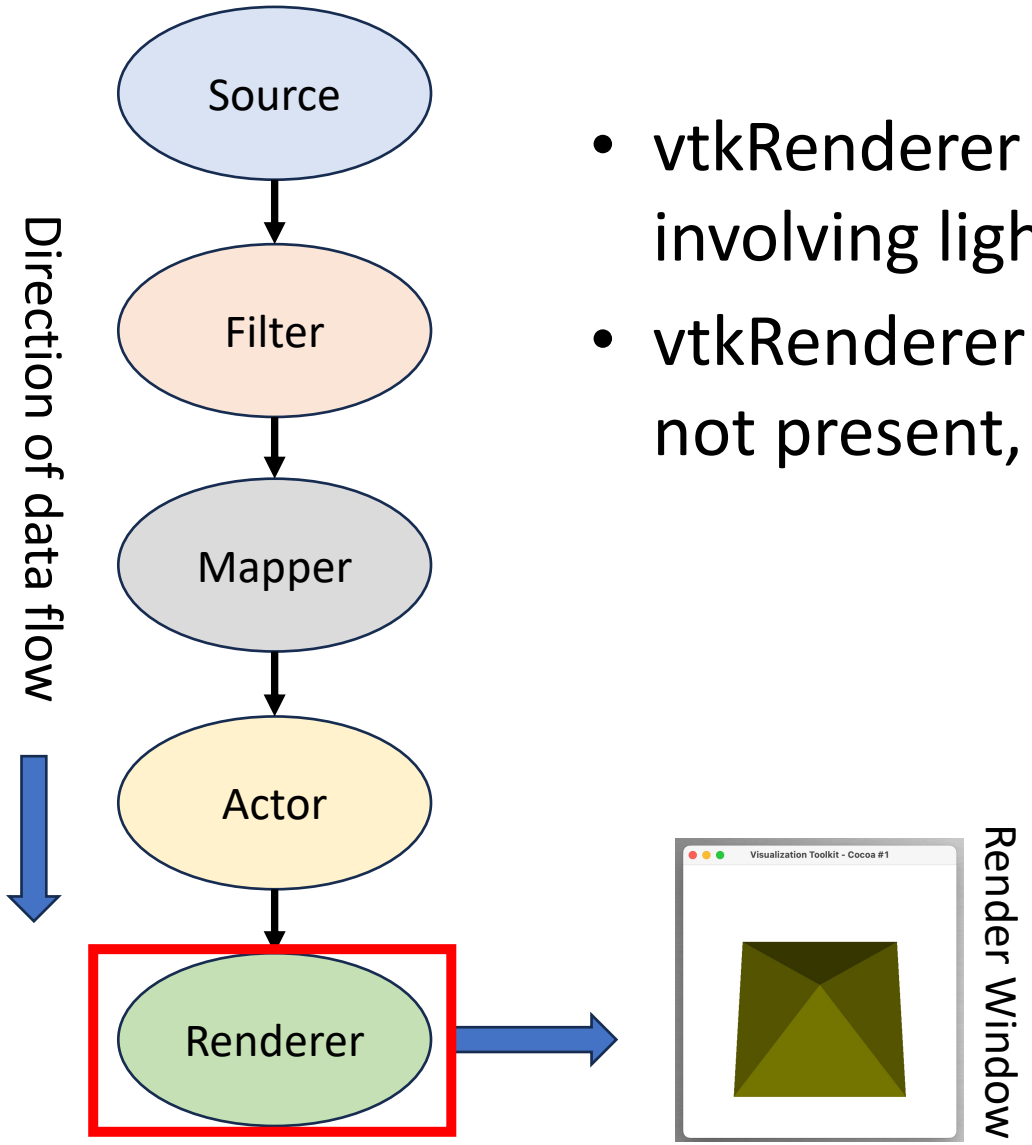
VTK Pipeline Execution: Actor



- Actors represent graphical data or objects with various properties for rendering
- A VTK actor contains
  - object properties (color, shading type, etc.)
  - geometry
  - transformations
- VTK actors need to work together with lights (vtkLight) and camera (vtkCamera) to make a scene

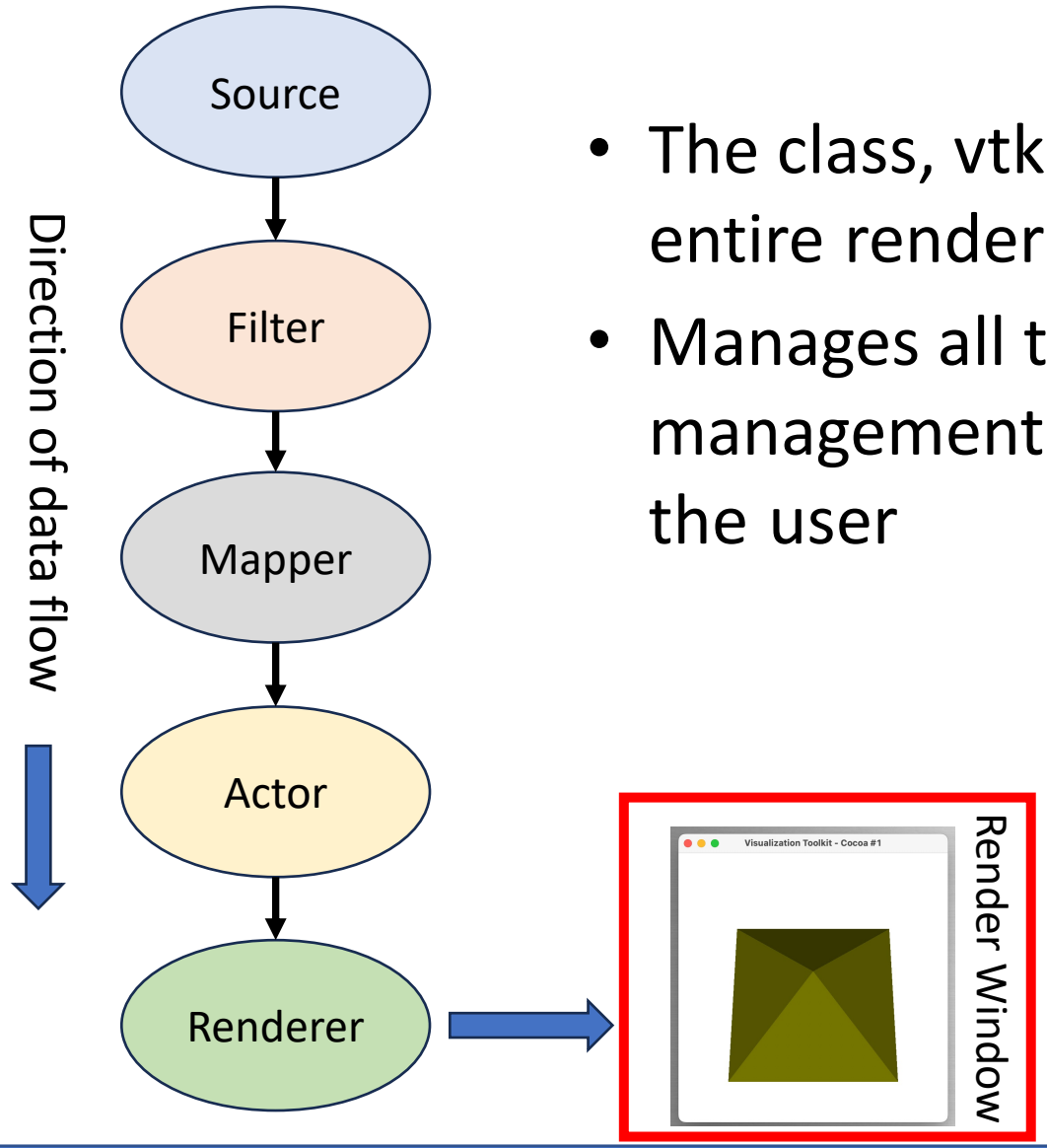


VTK Pipeline Execution: Renderer



- vtkRenderer coordinates the rendering process involving lights, camera, and actors
- vtkRenderer creates a default camera and light if not present, but needs to have at least one actor

VTK Pipeline Execution: Render Window



- The class, vtkRenderWindow ties the entire rendering process together
- Manages all the platform dependent window management issues and hide the details from the user

A Simple VTK Program

```
## Import VTK
from vtk import *


### Data
########################
p0 = [1.0, 1.0, 1.0]
p1 = [-1.0, 1.0, 1.0]
p2 = [-1.0, -1.0, 1.0]
p3 = [1.0, -1.0, 1.0]
p4 = [0.0, 0.0, 0.0]

### Create geometric primitives
###############################
points = vtkPoints()
points.InsertNextPoint(p0)
points.InsertNextPoint(p1)
points.InsertNextPoint(p2)
points.InsertNextPoint(p3)
points.InsertNextPoint(p4)

### Create Pyramid geometry
###############################
pyramid = vtkPyramid()
pyramid.GetPointIds().SetId(0, 0)
pyramid.GetPointIds().SetId(1, 1)
pyramid.GetPointIds().SetId(2, 2)
pyramid.GetPointIds().SetId(3, 3)
pyramid.GetPointIds().SetId(4, 4)

cells = vtkCellArray()
cells.InsertNextCell(pyramid)

### Create a grid and attach the geometry to it
###############################################
ug = vtkUnstructuredGrid()
ug.SetPoints(points)
ug.InsertNextCell(pyramid.GetCellType(), pyramid.GetPointIds())

### Create a mapper and give the grid to it
###########################################
mapper = vtkDataSetMapper()
mapper.SetInputData(ug)
```

A Simple VTK Program

```
## Import VTK
from vtk import *


### Data
########################
p0 = [1.0, 1.0, 1.0]
p1 = [-1.0, 1.0, 1.0]
p2 = [-1.0, -1.0, 1.0]
p3 = [1.0, -1.0, 1.0]
p4 = [0.0, 0.0, 0.0]

### Create geometric primitives
###############################
points = vtkPoints()
points.InsertNextPoint(p0)
points.InsertNextPoint(p1)
points.InsertNextPoint(p2)
points.InsertNextPoint(p3)
points.InsertNextPoint(p4)

### Create Pyramid geometry
###############################
pyramid = vtkPyramid()
pyramid.GetPointIds().SetId(0, 0)
pyramid.GetPointIds().SetId(1, 1)
pyramid.GetPointIds().SetId(2, 2)
pyramid.GetPointIds().SetId(3, 3)
pyramid.GetPointIds().SetId(4, 4)

cells = vtkCellArray()
cells.InsertNextCell(pyramid)

### Create a grid and attach the geometry to it
###############################################
ug = vtkUnstructuredGrid()
ug.SetPoints(points)
ug.InsertNextCell(pyramid.GetCellType(), pyramid.GetPointIds())

### Create a mapper and give the grid to it
###########################################
mapper = vtkDataSetMapper()
mapper.SetInputData(ug)
```

A Simple VTK Program

```
## Import VTK
from vtk import *


### Data
########################
p0 = [1.0, 1.0, 1.0]
p1 = [-1.0, 1.0, 1.0]
p2 = [-1.0, -1.0, 1.0]
p3 = [1.0, -1.0, 1.0]
p4 = [0.0, 0.0, 0.0]

### Create geometric primitives
################################
points = vtkPoints()
points.InsertNextPoint(p0)
points.InsertNextPoint(p1)
points.InsertNextPoint(p2)
points.InsertNextPoint(p3)
points.InsertNextPoint(p4)

### Create Pyramid geometry
################################
pyramid = vtkPyramid()
pyramid.GetPointIds().SetId(0, 0)
pyramid.GetPointIds().SetId(1, 1)
pyramid.GetPointIds().SetId(2, 2)
pyramid.GetPointIds().SetId(3, 3)
pyramid.GetPointIds().SetId(4, 4)

cells = vtkCellArray()
cells.InsertNextCell(pyramid)

### Create a grid and attach the geometry to it
############################################
ug = vtkUnstructuredGrid()
ug.SetPoints(points)
ug.InsertNextCell(pyramid.GetCellType(), pyramid.GetPointIds())

### Create a mapper and give the grid to it
########################################
mapper = vtkDataSetMapper()
mapper.SetInputData(ug)
```

A Simple VTK Program

```
## Import VTK
from vtk import *


### Data
########################
p0 = [1.0, 1.0, 1.0]
p1 = [-1.0, 1.0, 1.0]
p2 = [-1.0, -1.0, 1.0]
p3 = [1.0, -1.0, 1.0]
p4 = [0.0, 0.0, 0.0]

### Create geometric primitives
################################
points = vtkPoints()
points.InsertNextPoint(p0)
points.InsertNextPoint(p1)
points.InsertNextPoint(p2)
points.InsertNextPoint(p3)
points.InsertNextPoint(p4)

### Create Pyramid geometry
################################
pyramid = vtkPyramid()
pyramid.GetPointIds().SetId(0, 0)
pyramid.GetPointIds().SetId(1, 1)
pyramid.GetPointIds().SetId(2, 2)
pyramid.GetPointIds().SetId(3, 3)
pyramid.GetPointIds().SetId(4, 4)

cells = vtkCellArray()
cells.InsertNextCell(pyramid)

### Create a grid and attach the geometry to it
############################################
ug = vtkUnstructuredGrid()
ug.SetPoints(points)
ug.InsertNextCell(pyramid.GetCellType(), pyramid.GetPointIds())

### Create a mapper and give the grid to it
########################################
mapper = vtkDataSetMapper()
mapper.SetInputData(ug)
```

A Simple VTK Program

```
## Import VTK
from vtk import *


### Data
########################
p0 = [1.0, 1.0, 1.0]
p1 = [-1.0, 1.0, 1.0]
p2 = [-1.0, -1.0, 1.0]
p3 = [1.0, -1.0, 1.0]
p4 = [0.0, 0.0, 0.0]

### Create geometric primitives
###############################
points = vtkPoints()
points.InsertNextPoint(p0)
points.InsertNextPoint(p1)
points.InsertNextPoint(p2)
points.InsertNextPoint(p3)
points.InsertNextPoint(p4)

### Create Pyramid geometry
###############################
pyramid = vtkPyramid()
pyramid.GetPointIds().SetId(0, 0)
pyramid.GetPointIds().SetId(1, 1)
pyramid.GetPointIds().SetId(2, 2)
pyramid.GetPointIds().SetId(3, 3)
pyramid.GetPointIds().SetId(4, 4)

cells = vtkCellArray()
cells.InsertNextCell(pyramid)

### Create a grid and attach the geometry to it
###############################################
ug = vtkUnstructuredGrid()
ug.SetPoints(points)
ug.InsertNextCell(pyramid.GetCellType(), pyramid.GetPointIds())

### Create a mapper and give the grid to it
###########################################
mapper = vtkDataSetMapper()
mapper.SetInputData(ug)
```

A Simple VTK Program

```
## Import VTK
from vtk import *


### Data
########################
p0 = [1.0, 1.0, 1.0]
p1 = [-1.0, 1.0, 1.0]
p2 = [-1.0, -1.0, 1.0]
p3 = [1.0, -1.0, 1.0]
p4 = [0.0, 0.0, 0.0]

### Create geometric primitives
################################
points = vtkPoints()
points.InsertNextPoint(p0)
points.InsertNextPoint(p1)
points.InsertNextPoint(p2)
points.InsertNextPoint(p3)
points.InsertNextPoint(p4)

### Create Pyramid geometry
################################
pyramid = vtkPyramid()
pyramid.GetPointIds().SetId(0, 0)
pyramid.GetPointIds().SetId(1, 1)
pyramid.GetPointIds().SetId(2, 2)
pyramid.GetPointIds().SetId(3, 3)
pyramid.GetPointIds().SetId(4, 4)

cells = vtkCellArray()
cells.InsertNextCell(pyramid)

### Create a grid and attach the geometry to it
###############################################
ug = vtkUnstructuredGrid()
ug.SetPoints(points)
ug.InsertNextCell(pyramid.GetCellType(), pyramid.GetPointIds())

### Create a mapper and give the grid to it
###########################################
mapper = vtkDataSetMapper()
mapper.SetInputData(ug)
```

```
### Create an actor and give the mapper to it
#############################################
actor = vtkActor()
actor.SetMapper(mapper)
## specify color for Pyramid
actor.GetProperty().SetColor(0.5,0.5,0)

### Create a renderer and give the actor to it
##############################################
renderer = vtkRenderer()
renderer.AddActor(actor)
renderer.SetBackground(1,1,1)

### Create a renderwindow and attach it with the renderer and interactor
#########################################################################
renderWindow = vtkRenderWindow()
renderWindow.SetSize(800,800)
renderWindow.AddRenderer(renderer)
renderWindowInteractor = vtkRenderWindowInteractor()
renderWindowInteractor.SetRenderWindow(renderWindow)

# Create a nice view
#######################
renderer.ResetCamera()
renderer.GetActiveCamera().Azimuth(180)
renderer.GetActiveCamera().Elevation(-20)
renderer.ResetCameraClippingRange()

### Finally render the Pyradim object
#####################################
renderWindow.Render()
renderWindowInteractor.Start()
```

A Simple VTK Program

```
## Import VTK
from vtk import *


### Data
########################
p0 = [1.0, 1.0, 1.0]
p1 = [-1.0, 1.0, 1.0]
p2 = [-1.0, -1.0, 1.0]
p3 = [1.0, -1.0, 1.0]
p4 = [0.0, 0.0, 0.0]

### Create geometric primitives
################################
points = vtkPoints()
points.InsertNextPoint(p0)
points.InsertNextPoint(p1)
points.InsertNextPoint(p2)
points.InsertNextPoint(p3)
points.InsertNextPoint(p4)

### Create Pyramid geometry
################################
pyramid = vtkPyramid()
pyramid.GetPointIds().SetId(0, 0)
pyramid.GetPointIds().SetId(1, 1)
pyramid.GetPointIds().SetId(2, 2)
pyramid.GetPointIds().SetId(3, 3)
pyramid.GetPointIds().SetId(4, 4)

cells = vtkCellArray()
cells.InsertNextCell(pyramid)

### Create a grid and attach the geometry to it
###############################################
ug = vtkUnstructuredGrid()
ug.SetPoints(points)
ug.InsertNextCell(pyramid.GetCellType(), pyramid.GetPointIds())

### Create a mapper and give the grid to it
###########################################
mapper = vtkDataSetMapper()
mapper.SetInputData(ug)
```

```
### Create an actor and give the mapper to it
#############################################
actor = vtkActor()
actor.SetMapper(mapper)
## specify color for Pyramid
actor.GetProperty().SetColor(0.5,0.5,0)

### Create a renderer and give the actor to it
##############################################
renderer = vtkRenderer()
renderer.AddActor(actor)
renderer.SetBackground(1,1,1)

### Create a renderwindow and attach it with the renderer and interactor
#########################################################################
renderWindow = vtkRenderWindow()
renderWindow.SetSize(800,800)
renderWindow.AddRenderer(renderer)
renderWindowInteractor = vtkRenderWindowInteractor()
renderWindowInteractor.SetRenderWindow(renderWindow)

# Create a nice view
#####################
renderer.ResetCamera()
renderer.GetActiveCamera().Azimuth(180)
renderer.GetActiveCamera().Elevation(-20)
renderer.ResetCameraClippingRange()

### Finally render the Pyradim object
#####################################
renderWindow.Render()
renderWindowInteractor.Start()
```

A Simple VTK Program

```
## Import VTK
from vtk import *


### Data
########################
p0 = [1.0, 1.0, 1.0]
p1 = [-1.0, 1.0, 1.0]
p2 = [-1.0, -1.0, 1.0]
p3 = [1.0, -1.0, 1.0]
p4 = [0.0, 0.0, 0.0]

### Create geometric primitives
###############################
points = vtkPoints()
points.InsertNextPoint(p0)
points.InsertNextPoint(p1)
points.InsertNextPoint(p2)
points.InsertNextPoint(p3)
points.InsertNextPoint(p4)

### Create Pyramid geometry
###############################
pyramid = vtkPyramid()
pyramid.GetPointIds().SetId(0, 0)
pyramid.GetPointIds().SetId(1, 1)
pyramid.GetPointIds().SetId(2, 2)
pyramid.GetPointIds().SetId(3, 3)
pyramid.GetPointIds().SetId(4, 4)

cells = vtkCellArray()
cells.InsertNextCell(pyramid)

### Create a grid and attach the geometry to it
###############################################
ug = vtkUnstructuredGrid()
ug.SetPoints(points)
ug.InsertNextCell(pyramid.GetCellType(), pyramid.GetPointIds())

### Create a mapper and give the grid to it
###########################################
mapper = vtkDataSetMapper()
mapper.SetInputData(ug)
```

```
### Create an actor and give the mapper to it
#############################################
actor = vtkActor()
actor.SetMapper(mapper)
## specify color for Pyramid
actor.GetProperty().SetColor(0.5,0.5,0)

### Create a renderer and give the actor to it
##############################################
renderer = vtkRenderer()
renderer.AddActor(actor)
renderer.SetBackground(1,1,1)

### Create a renderwindow and attach it with the renderer and interactor
#########################################################################
renderWindow = vtkRenderWindow()
renderWindow.SetSize(800,800)
renderWindow.AddRenderer(renderer)
renderWindowInteractor = vtkRenderWindowInteractor()
renderWindowInteractor.SetRenderWindow(renderWindow)

# Create a nice view
#####################
renderer.ResetCamera()
renderer.GetActiveCamera().Azimuth(180)
renderer.GetActiveCamera().Elevation(-20)
renderer.ResetCameraClippingRange()

### Finally render the Pyradim object
#####################################
renderWindow.Render()
renderWindowInteractor.Start()
```

A Simple VTK Program

```
## Import VTK
from vtk import *


### Data
########################
p0 = [1.0, 1.0, 1.0]
p1 = [-1.0, 1.0, 1.0]
p2 = [-1.0, -1.0, 1.0]
p3 = [1.0, -1.0, 1.0]
p4 = [0.0, 0.0, 0.0]

### Create geometric primitives
################################
points = vtkPoints()
points.InsertNextPoint(p0)
points.InsertNextPoint(p1)
points.InsertNextPoint(p2)
points.InsertNextPoint(p3)
points.InsertNextPoint(p4)

### Create Pyramid geometry
################################
pyramid = vtkPyramid()
pyramid.GetPointIds().SetId(0, 0)
pyramid.GetPointIds().SetId(1, 1)
pyramid.GetPointIds().SetId(2, 2)
pyramid.GetPointIds().SetId(3, 3)
pyramid.GetPointIds().SetId(4, 4)

cells = vtkCellArray()
cells.InsertNextCell(pyramid)

### Create a grid and attach the geometry to it
###############################################
ug = vtkUnstructuredGrid()
ug.SetPoints(points)
ug.InsertNextCell(pyramid.GetCellType(), pyramid.GetPointIds())

### Create a mapper and give the grid to it
###########################################
mapper = vtkDataSetMapper()
mapper.SetInputData(ug)
```

```
### Create an actor and give the mapper to it
#############################################
actor = vtkActor()
actor.SetMapper(mapper)
## specify color for Pyramid
actor.GetProperty().SetColor(0.5,0.5,0)

### Create a renderer and give the actor to it
##############################################
renderer = vtkRenderer()
renderer.AddActor(actor)
renderer.SetBackground(1,1,1)

### Create a renderwindow and attach it with the renderer and interactor
#########################################################################
renderWindow = vtkRenderWindow()
renderWindow.SetSize(800,800)
renderWindow.AddRenderer(renderer)
renderWindowInteractor = vtkRenderWindowInteractor()
renderWindowInteractor.SetRenderWindow(renderWindow)

# Create a nice view
#####################
renderer.ResetCamera()
renderer.GetActiveCamera().Azimuth(180)
renderer.GetActiveCamera().Elevation(-20)
renderer.ResetCameraClippingRange()

### Finally render the Pyradim object
#####################################
renderWindow.Render()
renderWindowInteractor.Start()
```

A Simple VTK Program

```
## Import VTK
from vtk import *


### Data
########################
p0 = [1.0, 1.0, 1.0]
p1 = [-1.0, 1.0, 1.0]
p2 = [-1.0, -1.0, 1.0]
p3 = [1.0, -1.0, 1.0]
p4 = [0.0, 0.0, 0.0]

### Create geometric primitives
################################
points = vtkPoints()
points.InsertNextPoint(p0)
points.InsertNextPoint(p1)
points.InsertNextPoint(p2)
points.InsertNextPoint(p3)
points.InsertNextPoint(p4)

### Create Pyramid geometry
################################
pyramid = vtkPyramid()
pyramid.GetPointIds().SetId(0, 0)
pyramid.GetPointIds().SetId(1, 1)
pyramid.GetPointIds().SetId(2, 2)
pyramid.GetPointIds().SetId(3, 3)
pyramid.GetPointIds().SetId(4, 4)

cells = vtkCellArray()
cells.InsertNextCell(pyramid)

### Create a grid and attach the geometry to it
###############################################
ug = vtkUnstructuredGrid()
ug.SetPoints(points)
ug.InsertNextCell(pyramid.GetCellType(), pyramid.GetPointIds())

### Create a mapper and give the grid to it
###########################################
mapper = vtkDataSetMapper()
mapper.SetInputData(ug)
```

```
### Create an actor and give the mapper to it
#############################################
actor = vtkActor()
actor.SetMapper(mapper)
## specify color for Pyramid
actor.GetProperty().SetColor(0.5,0.5,0)

### Create a renderer and give the actor to it
##############################################
renderer = vtkRenderer()
renderer.AddActor(actor)
renderer.SetBackground(1,1,1)

### Create a renderwindow and attach it with the renderer and interactor
#########################################################################
renderWindow = vtkRenderWindow()
renderWindow.SetSize(800,800)
renderWindow.AddRenderer(renderer)
renderWindowInteractor = vtkRenderWindowInteractor()
renderWindowInteractor.SetRenderWindow(renderWindow)

# Create a nice view
#######################
renderer.ResetCamera()
renderer.GetActiveCamera().Azimuth(180)
renderer.GetActiveCamera().Elevation(-20)
renderer.ResetCameraClippingRange()

### Finally render the Pyradim object
#####################################
renderWindow.Render()
renderWindowInteractor.Start()
```

A Simple VTK Program

$> Python Pyramid.py

Demo