VTK Introduction

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

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, and Linux.

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

It has users and developers all over the world.
System Architecture

Interpreted Wrapper (Tcl, Java, Python)

• Tcl/Tk shell
• Java interpreter
• Python interpreter

C++ core

• Tcl/Tk source
• Java JDK
• Python source

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

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

Source code Installation: If you want to extend vtk

All class source code (could take hours to compile)
In VTK, visualizations are created via a pipeline as shown to the right.
VTK pipeline

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

The **source** imports (from file) or creates (e.g., function) the data
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.
One or more filters process the data (from source) to create geometric objects (lines or surfaces).
The **mapper** converts geometry to graphical primitives (points, line segments, triangles, ...)

```
''' 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()
```

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

---

VTK pipeline

---

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

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

---

### 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()
```

**Create an actor**
VTK pipeline

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

```py
''' 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)
```

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, ...)

```
''' 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)

Set a constant color for these graphical primitives
The **render** controls the camera and lighting.

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

```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()
```
The **render** controls the camera and lighting

Add the above graphical objects into the scene. Multiple set of graphical objects can be added.
The **render window** displays the result on the screen and sets the resolution.

---

```
''' 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()
```

Create a window on the screen.
VTK pipeline

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

```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)

Render our scene in that window
VTK pipeline

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

Set the resolution of the window.

---

```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)
```
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()
```

Add some user interaction (via mouse) to the render window
'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_reader.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()  # Nothing will happen until Render() is called.
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()  

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 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()
window.SetWindowName('COSC 6344 Visualization')
VTK pipeline

'The VTK pipeline consists of the following steps:

1. **Source**
   - Read a vtk data file
     ```python
vtk_reader = vtk.vtkDataSetReader()
vtk_reader.SetFileName(input_file_name)
```

2. **Filter**
   - Get geometry using a filter
     ```python
     vtk_geometry = vtk.vtkExtractEdges()
     vtk_geometry.SetInputData(vtk_reader.GetPolyDataOutput())
     vtk_geometry.SetInputConnection(vtk_reader.GetOutputPort())
     vtk_geometry.Update()
     ```

3. **Mapper**
   - Use a mapper to get the geometry primitives
     ```python
     vtk_poly_mapper = vtk.vtkPolyDataMapper()
     vtk_poly_mapper.SetInputConnection(vtk_geometry.GetOutputPort())
     vtk_poly_mapper.ScalarVisibilityOff() # Turn this on when showing scalar field
     ```

4. **Actor**
   - Create an actor and set the appearance for the mapper
     ```python
     vtk_actor = vtk.vtkActor()
     vtk_actor.SetMapper(vtk_poly_mapper)
     vtk_actor.GetProperty().SetColor(1, 1, 0)
     ```

5. **Renderer**
   - Create a renderer to set camera, lighting
     ```python
     render = vtk.vtkRenderer()
     render.AddActor(vtk_actor)
     ```

6. **Render window**
   - Set the render window to show the result
     ```python
     window = vtk.vtkRenderWindow()
     window.AddRenderer(render)
     window.SetSize(600, 600)
     ```

7. **Render window**
   - Add user interaction to the render window
     ```python
     window_interactor = vtk.vtkRenderWindowInteractor()
     window_interactor.SetRenderWindow(window)
     window_interactor.Initialize()
     ```

   - Launch the window
     ```python
     window.Render()
     window.SetWindowName('COSC_6344_Visualization')
     window_interactor.Start()
     ```
Show demo
VTK Cell Types

- Vertex
- Polyvertex
- Line
- Polyline
- Triangle
- Triangle strip
- Quadrilateral
- Pixel
- Tetrahedron
- Hexahedron
- Voxel
- Wedge
- Pyramid
VTK Dataset Types

(a) Image Data (vtkImageData)
(b) Rectilinear Grid (vtkRectilinearGrid)
(c) Structured Grid (vtkStructuredGrid)
(d) Unstructured Points (use vtkPolyData)
(e) Polyhedral Data (vtkPolyData)
(f) Unstructured Grid (vtkUnstructuredGrid)
VTK Dataset Hierarchy
VTK Dataset Types

(b) Rectilinear Grid
(vtkRectilinearGrid)

(c) Structured Grid
(vtkStructuredGrid)

(d) Unstructured Points
(use vtkPolyData)

(e) Polyhedral Data
(vtkPolyData)

(f) Unstructured Grid
(vtkUnstructuredGrid)
VTK Dataset Types

(a) Image Data (vtkImageData)
(b) Structured Grid (vtkStructuredGrid)
(c) Unstructured Grid (vtkUnstructuredGrid)

Diagram showing various VTK dataset types and readers.
Attribute Types

Scalar: single data value
Vector: 3D direction and magnitude
Normal: 3D direction

2D: (u,v)
3D: (u,v,w)

Texture coordinate: n-dimensional index into texture map
Tensor: nxn matrix

Array 0 Array 1 • • • Array n-1

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
-1.00000 0.00000
-0.95918 0.00000
-0.91836 0.00000
-0.87755 0.00000
-0.83573 0.00000
0.00000 0.00000
0.95918 0.00000
0.91836 0.00000
0.87755 0.00000
0.83573 0.00000
0.00000 0.00000
0.75918 0.00000

.......

0.95918 1.00000 0.00000
1.00000 1.00000 0.00000
POINT_DATA 2500
SCALARS s float 1
LOOKUP_TABLE s
70.599884
71.457848
71.131317
69.636490
67.046814
63.487019
59.123165
54.440830

.......

# 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
0.716014 0.020922 0.785714

.......

0.857760 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
3 144 133 77

.......

3 234 343 381
POINT_DATA 382
VECTORS velocity float
-0.001876 0.001649 0.000000
-0.001929 0.001638 0.000000
-0.001772 0.001426 0.000000

.......

-0.002146 -0.001512 0.000000
SCALARS s float 1
LOOKUP_TABLE s
0.00000
0.00000
0.00000
0.00000
0.00000

.......

# 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 -0.841815 -0.458954
-1.729041 -0.187040 2.916374

.......

0.355445 1.133200 1.959400
0.333941 1.167505 -2.848074

CELLS 996 3984
3 270 374 393
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
5
5
5
5
5

.......

...
Example of getting value range of a scalar field

```python
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

```python
all_scalars = vtk_geometry.GetOutput().GetPointData().GetScalars('s')
```

name of the scalar field

Example of getting all vector values

```python
all_vectors = vtk_reader.GetOutput().GetPointData().GetVectors('velocity')
```

name of the vector field
Additional References

- VTK Wiki [http://www.vtk.org/Wiki/VTK](http://www.vtk.org/Wiki/VTK)

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

- Books:
    (https://www.kitware.com/products/books/VTKUsersGuide.pdf)
    Online version [https://lorensen.github.io/VTKExamples/site/VTKBook/](https://lorensen.github.io/VTKExamples/site/VTKBook/)