VTK Introduction
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.

• Object--oriented design with different interpreted language wrappers.

https://vtk.org/gallery/
The core of VTK is written in C++ 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 all over the world.
# System Architecture

## Interpreted Wrapper (Tcl, Java, Python)

<table>
<thead>
<tr>
<th>C++ Core</th>
<th>Libraries and includes (dll and .h files) Or (.a and .h files)</th>
<th>All class source code (could take hours to compile)</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th></th>
<th>• Tcl/Tk shell</th>
<th>• Tcl/Tk source</th>
<th>• Java interpreter</th>
<th>• Java JDK</th>
<th>• Python interpreter</th>
<th>• Python source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary Installation: if you will use the classes to build your application</td>
<td>Source code Installation: If you want to extend vtk</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
In VTK, visualizations are created via a pipeline as shown to the right.
The **source** imports (from file) or creates (e.g., function) the data.
VTK pipeline

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

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

Source

Filter

Mapper

Actor

Renderer

Render window

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

VTK pipeline connection

1. `Receiver.SetInputConnection(Supplier.GetOutputPort())`
2. `Receiver.SetInputData(Supplier.GetOutput())`
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,...)
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**!*

```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
```
VTK pipeline

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

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

---

**VTK pipeline**

---

***Step 1: Read a vtk data file***

```python
_reader = vtk.vtkDataSetReader()
_reader.SetFileName(input_file_name)
```

***Step 2: 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()
```

***Step 3: 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
```

***Step 4: create an actor and set the appearance for the mapper***

```python
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, ...)

---

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

# Step 2: Get geometry using a filter
geometry = vtk.vtkExtractEdges()
#geometry.SetInputData(reader.GetOutput())
geometry.SetInputConnection(reader.GetOutputPort())
geometry.Update()

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

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

Set a constant color for these graphical primitives
```
VTK pipeline

The `render` controls the camera and lighting

```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()
Create the scene render and set the camera and lighting. Here we will use the default setting.
VTK pipeline

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.

---

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()

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

**VTK pipeline**

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

2. **Filter**
   ```python
   # 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()
   ```

3. **Mapper**
   ```python
   # 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
   ```

4. **Actor**
   ```python
   # 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)
   ```

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

6. **Render window**
   ```python
   # Step 6: set the render window to show the result
   window = vtk.vtkRenderWindow()
   window.AddRenderer(render)
   ```

**Render our scene in that window**
The render window displays the result on the screen and sets the resolution.

Set the resolution of the window.
VTK pipeline

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

2. Step 2: Get geometry using a filter
   ```
   _geometry = vtk.vtkExtractEdges()
   #_geometry.SetInputData(_reader.GetPolyDataOutput())
   _geometry.SetInputConnection(_reader.GetOutputPort())
   _geometry.Update()
   ```

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

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

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

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

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

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

Source ➔ Filter ➔ Mapper ➔ Actor ➔ Renderer ➔ Render window

Nothing will happen until Render() is called.
VTK pipeline

Source
Filter
Mapper
Actor
Renderer
Render window

---

Once Render() is called.
VTK pipeline

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

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

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

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

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

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

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

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

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

2. **Filter**: Get geometry using a filter
   ```python
   geometry = vtk.vtkExtractEdges()
   geometry.SetInputData(reader.GetOutput())
   geometry.Update()
   ```

3. **Mapper**: Use a mapper to get the geometry primitives
   ```python
   poly_mapper = vtk.vtkPolyDataMapper()
   poly_mapper.SetInputConnection(geometry.GetOutputPort())
   ```

4. **Actor**: Create an actor and set the appearance for the mapper
   ```python
   actor = vtk.vtkActor()
   actor.SetMapper(poly_mapper)
   ```

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

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

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

8. **Launch the window**
   ```python
   render()  
   window.SetWindowTitle('COSC 6344 Visualization')
   window_interactor.Start()
   ```

**Start user interaction**
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

- Rectilinear Grid (vtkRectilinearGrid)
- Structured Grid (vtkStructuredGrid)
- Unstructured Points (use vtkPolyData)
- Polyhedral Data (vtkPolyData)
- Unstructured Grid (vtkUnstructuredGrid)

Diagram showing various vtkDataReader classes and their relationships with different dataset types.
VTK Dataset Types

(a) Image Data (vtkImageData)

(c) Structured Grid (vtkStructuredGrid)

(e) Polygonal Data (vtkPolyData)

(f) Unstructured Grid (vtkUnstructuredGrid)
Attribute Types

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

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)

Example of getting value range of a scalar field

tk_geometry.GetOutput().GetPointData().GetArray(scalar_field).GetRange()
# vtk DataFile Version 3.0
DAT Converted Data
ASCII

DATASET STRUCTURED_GRID
DIMENSIONS 50 50 1
POINTS 2500 float
-1.00000  -1.00000  0.00000
-0.959184  -1.00000  0.00000
-0.918367  -1.00000  0.00000
-0.877551  -1.00000  0.00000
-0.836735  -1.00000  0.00000
-0.8  -1.00000  0.00000
0.795918  1.00000  0.00000
0.8  1.00000  0.00000
0.836735  1.00000  0.00000
0.877551  1.00000  0.00000
0.918367  1.00000  0.00000
0.959184  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
68.048614
63.487019
59.123165
58.140820


# 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
0.814780  0.370024  0.785714
0.157770  0.158756  0.785714
0.716014  0.020922  0.785714

POLYGONS 702 2808
3  141  140  44
3  88  81  67
3  57  109  79
3  173  43  140
3  140  43  44

POINT_DATA 382
SCALARS velocity float
-0.001876  0.001649  0.000000
-0.001929  0.001638  0.000000
-0.001777  0.001562  0.000000
-0.002146  0.001512  0.000000

SCALARS s float 1
LOOKUP_TABLE s
0.000000
0.000000
0.000000
0.000000

# vtk DataFile Version 1.0
rbc_001.vtk  3D Unstructured Grid of Triangles
ASCII

DATASET UNSTRUCTURED_GRID
POINTS 500 float
-3.424999 -0.855454  2.257396
-1.484919  0.665606 -3.151304
1.636841 -0.848154 -0.458954
3  730341 0 187065 1 319734

CELLS 996 3984
3  270  374  303
3  104  55  232
3  339  225  45
3  410  374  315
3  104  232  416

CELL_TYPES 996
5
5
5
5
5
5
5
5

3  0  225  339
3  0  339  410
3  374  410  339
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

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

• Books:
    Online version https://lorensen.github.io/VTKExamples/site/VTKBook/