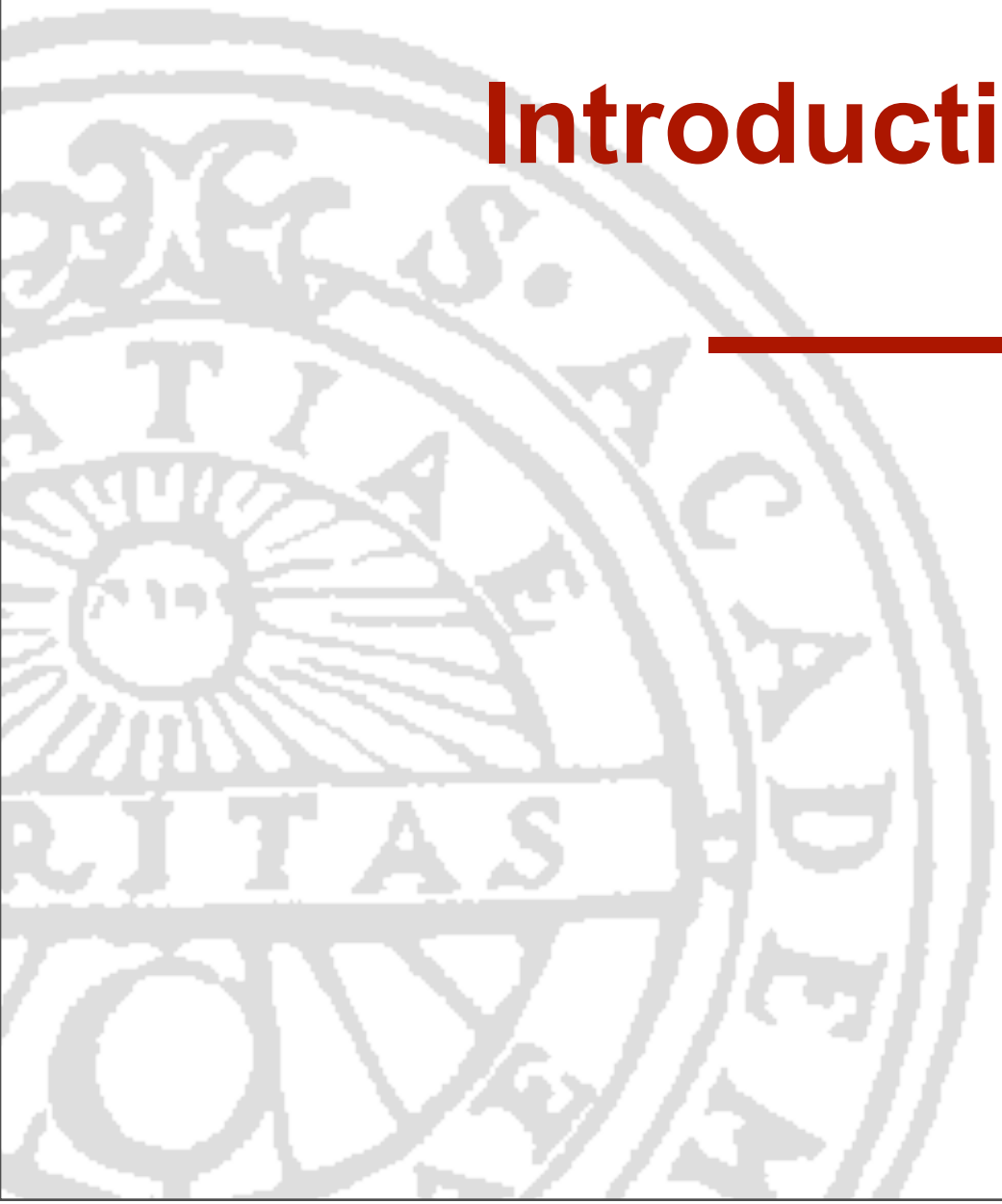


Introduction to Python and VTK

Anders Hast

Informationsteknologi

Key Features

Key Features

- Very clear, readable syntax

Key Features

- Very clear, readable syntax
- Intuitive object orientation

Key Features

- Very clear, readable syntax
- Intuitive object orientation
- Full modularity, supporting hierarchical packages

python™

Key Features

- Very clear, readable syntax
- Intuitive object orientation
- Full modularity, supporting hierarchical packages
- Exception-based error handling

python™

Key Features

- Very clear, readable syntax
- Intuitive object orientation
- Full modularity, supporting hierarchical packages
- Exception-based error handling
- Very high level dynamic data types

python™

Key Features

- Very clear, readable syntax
- Intuitive object orientation
- Full modularity, supporting hierarchical packages
- Exception-based error handling
- Very high level dynamic data types
- Extensive standard libraries and third party modules for virtually every task

Key Features

- Very clear, readable syntax
- Intuitive object orientation
- Full modularity, supporting hierarchical packages
- Exception-based error handling
- Very high level dynamic data types
- Extensive standard libraries and third party modules for virtually every task
- Extensions and modules easily written in C, C++

Why Python?

- Clear syntax
- No compiling
- We just make calls to the VTK API
- Easy to incorporate other API's such as
  - OpenCV
  - SciPy
  - NumPy
  - MatPlotLib
  - ITK

Example Code

- The following examples shows what you can do with Python, but you can do a lot more!
  - Input
  - Lists
  - Tuples
  - Flow control
    - While
    - If
  - Dictionaries
  - Exceptions

Hello World

```
# Hello World
print 'Hello World'
```

```
/home/user/Python/Python> python HelloWorld.py
Hello World
/home/user/Python/Python>
```

Input

```
# Input
word = raw_input("Write a word in italian: ")
print "Your italian word was", word
```

Write a word in Italian: Ciao
Your italian word was Ciao

Math

```
# Math
a=10
b=8.2
print a/3, a/3.0
print b/2, b/2.0
print "(b*a)/3.0=" , (b*a)/3.0, "."
```

```
3 3.33333333333
4.1 4.1
(b*a)/3.0= 27.3333333333 .
```

Flow Control

```
# Control
i=1
while i < 7:
    if i == 3:
        print "3"
    elif i==4:
        print "4"
    else:
        print "x"
    i=i+1

print "End"
```

```
x
x
3
4
x
x
End
```

Lists

```
# Lists
cars=['volvo', 'saab', 'fiat', 'skoda']
print cars[2]
cars.append('audi')
print cars
```

```
fiat
['volvo', 'saab', 'fiat', 'skoda', 'audi']
```

Tuples

```
scooters= 'vespa', 'lambretta'
# scooter.append() does not work on tuples!
print scooters, scooters[0]
```

```
('vespa', 'lambretta') vespa
```

Lists

```
vehicle=list()
vehicle.append(scooters)
vehicle.append(cars)
print vehicle
print vehicle[1]
```

```
[('vespa', 'lambretta'), ['volvo', 'saab', 'fiat',
'skoda', 'audi']]
['volvo', 'saab', 'fiat', 'skoda', 'audi']
```

Looping with for

```
for car in cars:
    print car
```

```
volvo
saab
fiat
skoda
audi
```

Dictionaries

```
# Dictionaries
EngIta={'all':'tutto', 'begin': 'cominciare',
'dark': ['buio', 'scuro'], 'find':
'trovare'}

print EngIta
print EngIta['begin']
print EngIta['dark']
print
for word in EngIta:
    print word+" =", EngIta[word]
```

```
{'dark': ['buio', 'scuro'], 'all': 'tutto', 'begin': 'cominciare',
'find': 'trovare'}
cominciare
['buio', 'scuro']

dark = ['buio', 'scuro']
all = tutto
begin = cominciare
find = trovare
```

Exceptions

```
try:
    word = raw_input("Write a word in English: ")
    print EngIta[word]
except KeyError:
    print "Word not in dictionary!"
```

```
Write a word in English: dark
['buio', 'scuro']

Write a word in English: light
Word not in dictionary!
```

Running programs

- python progname.py
- or
  - ./progname.py
- Then the first line in the program must be
  - #!/usr/bin/env python

Conclusions

- Python is rather easy to learn
- The syntax is similar to C/C++ and Java
- Use indention and not `{}`
- Python also have features that is different from C, or even are unique for Python
  - Lists
  - Tuples
  - Dictionaries
  - the use of the `for` statements

VTK

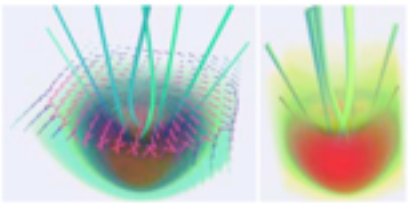
- Open source, freely available software for
  - 3D computer graphics
  - image processing
  - visualization
- Managed by Kitware, Inc.
- Object-oriented design (C++)
- High-level of abstraction
- Use C++, Tcl/Tk, Python, Java

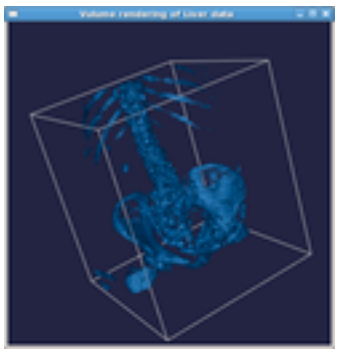
UPPSALA UNIVERSITET

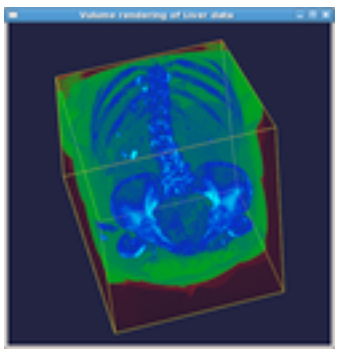
Informationsteknologi

Some examples of what you can do with VTK

- Visualization techniques for visualizing
  - scalar, vector and tensor fields
  - volume data, etc
- Mesh and polygon processing
- Image processing
- Isosurface extraction
- Your own algorithms







And more

- Surface rendering
- Volume rendering
  - Ray casting
  - Texture mapping (2D)
- Lights and cameras
- Textures
- Save render window to .png, .jpg, ... (useful for movie creation)

Hello World

```
# This program demonstrates how VTK can be used to render a text
# The user can also interact with the text by using the mouse

# load VTK
from vtk import *

# Create a Text source and set the text
text = vtkTextSource()
text.SetText("UPPMAX")
#text.SetForegroundColor(0.6,0.2,0.2)

# Create a mapper and set the Text source as input
textMapper = vtkPolyDataMapper()
textMapper.SetInputConnection(text.GetOutputPort())

# Create an actor and set the mapper as input
textActor = vtkActor()
textActor.SetMapper(textMapper)

# Create a renderer
ren = vtkRenderer()

# Assign the actor to the renderer
ren.AddActor(textActor)

# Create a rendering window
renWin = vtkRenderWindow()

# Add the renderer to the window
renWin.AddRenderer(ren)

# Set the name of the window (this is optional)
renWin.SetWindowName("Hello World!")

# Make sure that we can interact with the application
iren = vtkRenderWindowInteractor()
iren.SetRenderWindow(renWin)

# Initialze and start the application
iren.Initialize()
iren.Start()
```

UPPSALA UNIVERSITET

Informationsteknologi

Hello World

```
from vtk import *                                    # load VTK
text = vtkTextSource()
text.SetText("UPPMAX")
textMapper = vtkPolyDataMapper()
textMapper.SetInputConnection(text.GetOutputPort())
textActor = vtkActor()
textActor.SetMapper(textMapper)
ren = vtkRenderer()
ren.AddActor(textActor)
renWin = vtkRenderWindow()
renWin.AddRenderer(ren)
renWin.SetWindowName("Hello World!")
iren = vtkRenderWindowInteractor()
iren.SetRenderWindow(renWin)
iren.Initialize()
iren.Start()
```

UPPSALA UNIVERSITET

Informationsteknologi

Hello World

```
from vtk import *
text = vtkTextSource()                          # Create a Text source
text.SetText("UPPMAX")                          and set the text
textMapper = vtkPolyDataMapper()
textMapper.SetInputConnection(text.GetOutputPort())
textActor = vtkActor()
textActor.SetMapper(textMapper)
ren = vtkRenderer()
ren.AddActor(textActor)
renWin = vtkRenderWindow()
renWin.AddRenderer(ren)
renWin.SetWindowName("Hello World!")
iren = vtkRenderWindowInteractor()
iren.SetRenderWindow(renWin)
iren.Initialize()
iren.Start()
```

Hello World

```
from vtk import *
text = vtkTextSource()
text.SetText("UPPMAX")
textMapper = vtkPolyDataMapper()
textMapper.SetInputConnection(text.GetOutputPort())
textActor = vtkActor()
textActor.SetMapper(textMapper)
ren = vtkRenderer()
ren.AddActor(textActor)
renWin = vtkRenderWindow()
renWin.AddRenderer(ren)
renWin.SetWindowName("Hello World!")
iren = vtkRenderWindowInteractor()
iren.SetRenderWindow(renWin)
iren.Initialize()
iren.Start()
```

Create a mapper and set the Text source as input

UPPSALA UNIVERSITET

Informationsteknologi

Hello World

```
from vtk import *
text = vtkTextSource()
text.SetText("UPPMAX")
textMapper = vtkPolyDataMapper()
textMapper.SetInputConnection(text.GetOutputPort())
textActor = vtkActor()                          # Create an actor and
textActor.SetMapper(textMapper)                  set the mapper as input
ren = vtkRenderer()
ren.AddActor(textActor)
renWin = vtkRenderWindow()
renWin.AddRenderer(ren)
renWin.SetWindowName("Hello World!")
iren = vtkRenderWindowInteractor()
iren.SetRenderWindow(renWin)
iren.Initialize()
iren.Start()
```

UPPSALA UNIVERSITET

Informationsteknologi

Hello World

```
from vtk import *
text = vtkTextSource()
text.SetText("UPPMAX")
textMapper = vtkPolyDataMapper()
textMapper.SetInputConnection(text.GetOutputPort())
textActor = vtkActor()
textActor.SetMapper(textMapper)
ren = vtkRenderer()                          # Create a renderer
ren.AddActor(textActor)                      # Assign the actor to the renderer
renWin = vtkRenderWindow()
renWin.AddRenderer(ren)
renWin.SetWindowName("Hello World!")
iren = vtkRenderWindowInteractor()
iren.SetRenderWindow(renWin)
iren.Initialize()
iren.Start()
```

UPPSALA UNIVERSITET

Informationsteknologi

Hello World

```
from vtk import *
text = vtkTextSource()
text.SetText("UPPMAX")
textMapper = vtkPolyDataMapper()
textMapper.SetInputConnection(text.GetOutputPort())
textActor = vtkActor()
textActor.SetMapper(textMapper)
ren = vtkRenderer()
ren.AddActor(textActor)
renWin = vtkRenderWindow()                    # Create a rendering window
renWin.AddRenderer(ren)                       # Add the renderer to the window
renWin.SetWindowName("Hello World!")          # Set the name of the window
iren = vtkRenderWindowInteractor()
iren.SetRenderWindow(renWin)
iren.Initialize()
iren.Start()
```

UPPSALA UNIVERSITET

Informationsteknologi

Hello World

```
from vtk import *
text = vtkTextSource()
text.SetText("UPPMAX")
textMapper = vtkPolyDataMapper()
textMapper.SetInputConnection(text.GetOutputPort())
textActor = vtkActor()
textActor.SetMapper(textMapper)
ren = vtkRenderer()
ren.AddActor(textActor)
renWin = vtkRenderWindow()
renWin.AddRenderer(ren)
renWin.SetWindowName("Hello World!")
iren = vtkRenderWindowInteractor()
iren.SetRenderWindow(renWin)
iren.Initialize()
iren.Start()
```

```
# Make sure that we can
interact with the application
# Initialze and
start the application
```

Hungarian Notation

```
from vtk import *
sText = vtkTextSource()
sText.SetText("UPPMAX")

mText = vtkPolyDataMapper()
mText.SetInputConnection(sText.GetOutputPort())
aText = vtkActor()
aText.SetMapper(mText)

rMain = vtkRenderer()
rMain.AddActor(aText)
wMain = vtkRenderWindow()
wMain.AddRenderer(rMain)
wMain.SetWindowName("Hello World!")

iMain = vtkRenderWindowInteractor()
iMain.SetRenderWindow(wMain)
iMain.Initialize()
iMain.Start()
```

Color the text

```
sText.SetForegroundColor(0.6,1.0,0.2)
```

UPPSALA UNIVERSITET

Informationsteknologi

Result

UPPSALA UNIVERSITET

Informationsteknologi

Mappers

- vtkMapper is an abstract class to specify interface between data and graphics primitives.
- Subclasses of vtkMapper map data through a lookuptable and control the creation of rendering primitives that interface to the graphics library.
- The mapping can be controlled by supplying a lookup table and specifying a scalar range to map data through.
- vtkPolyDataMapper is a class that maps polygonal data (i.e., vtkPolyData) to graphics primitives.
- vtkPolyDataMapper serves as a superclass for device-specific poly data mappers, that actually do the mapping to the rendering/graphics hardware/software.

Actors

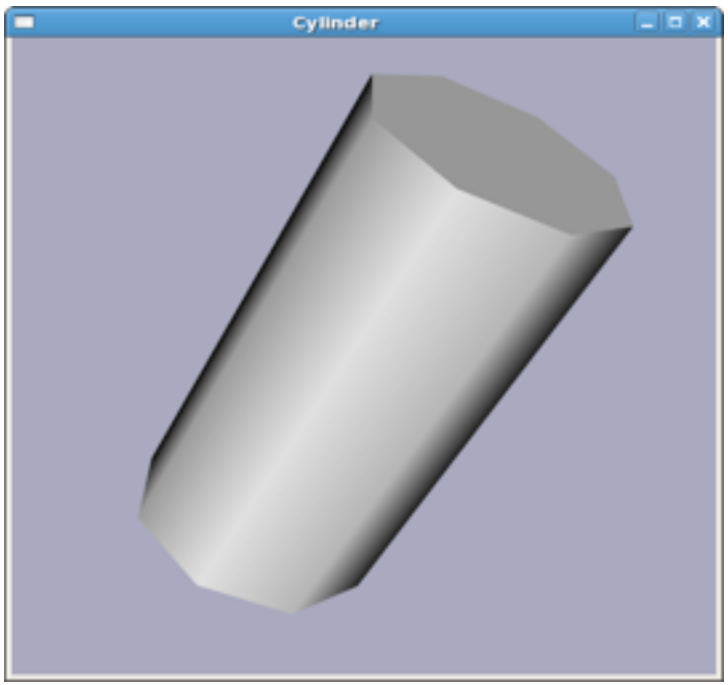
- vtkActor is used to represent an entity in a rendering scene.
- It inherits functions related to the actors position, and orientation from vtkProp.
- The actor also has scaling and maintains a reference to the defining geometry (i.e., the mapper), rendering properties, and possibly a texture map.

UPPSALA UNIVERSITET

Informationsteknologi

A Cylinder

- Render and interact with a Cylinder
- Set the size of the window
- Change background color



```
from vtk import *                                    # From Vtk Import *

cylinder = vtkCylinderSource()
cylinder.SetResolution(8)
cylinder.SetHeight(12)
cylinder.SetRadius(3)

cylinderMapper = vtkPolyDataMapper()
cylinderMapper.SetInputConnection(cylinder.GetOutputPort())

cylinderActor = vtkActor()
cylinderActor.SetMapper(cylinderMapper)

ren = vtkRenderer()
ren.AddActor(cylinderActor)
ren.SetBackground(0.6, 0.6, 0.7)

renWin = vtkRenderWindow()
renWin.AddRenderer(ren)
renWin.SetWindowName("Cylinder")
renWin.SetSize(500,500)

iren = vtkRenderWindowInteractor()
iren.SetRenderWindow(renWin)
iren.Initialize()
iren.Start()
```

```
from vtk import *

cylinder = vtkCylinderSource()                    # Create a Cylinder
cylinder.SetResolution(8)
cylinder.SetHeight(12)
cylinder.SetRadius(3)

cylinderMapper = vtkPolyDataMapper()
cylinderMapper.SetInputConnection(cylinder.GetOutputPort())

cylinderActor = vtkActor()
cylinderActor.SetMapper(cylinderMapper)

ren = vtkRenderer()
ren.AddActor(cylinderActor)
ren.SetBackground(0.6, 0.6, 0.7)

renWin = vtkRenderWindow()
renWin.AddRenderer(ren)
renWin.SetWindowName("Cylinder")
renWin.SetSize(500,500)

iren = vtkRenderWindowInteractor()
iren.SetRenderWindow(renWin)
iren.Initialize()
iren.Start()
```

```
from vtk import *

cylinder = vtkCylinderSource()
cylinder.SetResolution(8)
cylinder.SetHeight(12)
cylinder.SetRadius(3)

cylinderMapper = vtkPolyDataMapper()                    # Create a Mapper
cylinderMapper.SetInputConnection(cylinder.GetOutputPort())

cylinderActor = vtkActor()
cylinderActor.SetMapper(cylinderMapper)

ren = vtkRenderer()
ren.AddActor(cylinderActor)
ren.SetBackground(0.6, 0.6, 0.7)

renWin = vtkRenderWindow()
renWin.AddRenderer(ren)
renWin.SetWindowName("Cylinder")
renWin.SetSize(500,500)

iren = vtkRenderWindowInteractor()
iren.SetRenderWindow(renWin)
iren.Initialize()
iren.Start()
```

```
from vtk import *

cylinder = vtkCylinderSource()
cylinder.SetResolution(8)
cylinder.SetHeight(12)
cylinder.SetRadius(3)

cylinderMapper = vtkPolyDataMapper()
cylinderMapper.SetInputConnection(cylinder.GetOutputPort())

cylinderActor = vtkActor()                                  # Create an Actor
cylinderActor.SetMapper(cylinderMapper)

ren = vtkRenderer()
ren.AddActor(cylinderActor)
ren.SetBackground(0.6, 0.6, 0.7)

renWin = vtkRenderWindow()
renWin.AddRenderer(ren)
renWin.SetWindowName("Cylinder")
renWin.SetSize(500,500)

iren = vtkRenderWindowInteractor()
iren.SetRenderWindow(renWin)
iren.Initialize()
iren.Start()
```

```
from vtk import *

cylinder = vtkCylinderSource()
cylinder.SetResolution(8)
cylinder.SetHeight(12)
cylinder.SetRadius(3)

cylinderMapper = vtkPolyDataMapper()
cylinderMapper.SetInputConnection(cylinder.GetOutputPort())

cylinderActor = vtkActor()
cylinderActor.SetMapper(cylinderMapper)

ren = vtkRenderer()                          # Create a Renderer
ren.AddActor(cylinderActor)                  and add Cylinder
ren.SetBackground(0.6, 0.6, 0.7)             # Set background Color

renWin = vtkRenderWindow()
renWin.AddRenderer(ren)
renWin.SetWindowName("Cylinder")
renWin.SetSize(500,500)

iren = vtkRenderWindowInteractor()
iren.SetRenderWindow(renWin)
iren.Initialize()
iren.Start()
```

```
from vtk import *

cylinder = vtkCylinderSource()
cylinder.SetResolution(8)
cylinder.SetHeight(12)
cylinder.SetRadius(3)

cylinderMapper = vtkPolyDataMapper()
cylinderMapper.SetInputConnection(cylinder.GetOutputPort())

cylinderActor = vtkActor()
cylinderActor.SetMapper(cylinderMapper)

ren = vtkRenderer()
ren.AddActor(cylinderActor)
ren.SetBackground(0.6, 0.6, 0.7)

renWin = vtkRenderWindow()                          # Create a rendering window
renWin.AddRenderer(ren)                             # Add the renderer to the window
renWin.SetWindowName("Cylinder")                    # Set the name of the window
renWin.SetSize(500,500)

iren = vtkRenderWindowInteractor()
iren.SetRenderWindow(renWin)
iren.Initialize()
iren.Start()
```

```
from vtk import *

cylinder = vtkCylinderSource()
cylinder.SetResolution(8)
cylinder.SetHeight(12)
cylinder.SetRadius(3)

cylinderMapper = vtkPolyDataMapper()
cylinderMapper.SetInputConnection(cylinder.GetOutputPort())

cylinderActor = vtkActor()
cylinderActor.SetMapper(cylinderMapper)

ren = vtkRenderer()
ren.AddActor(cylinderActor)
ren.SetBackground(0.6, 0.6, 0.7)

renWin = vtkRenderWindow()
renWin.AddRenderer(ren)
renWin.SetWindowName("Cylinder")
renWin.SetSize(500,500)
```

```
iren = vtkRenderWindowInteractor()
iren.SetRenderWindow(renWin)
iren.Initialize()
iren.Start()
```

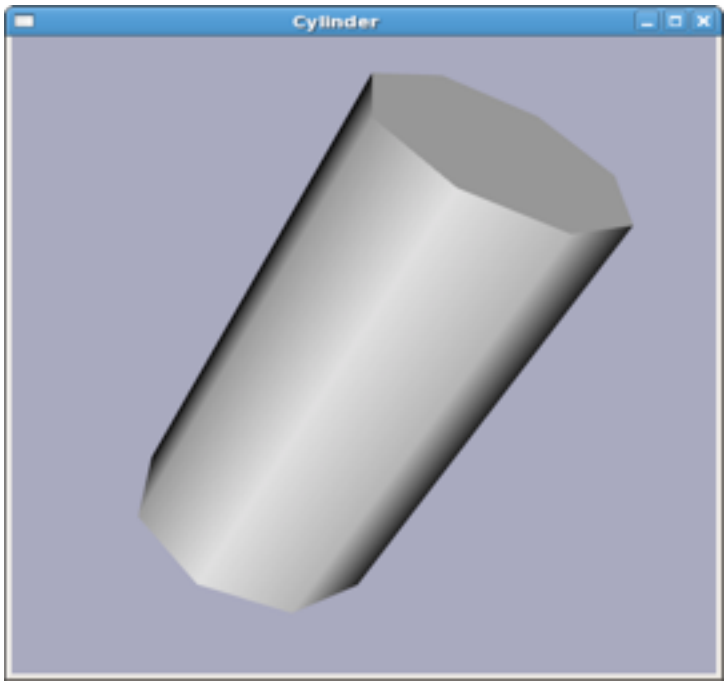
Make sure that we can interact with the application

Initialze and start the application

UPPSALA UNIVERSITET

Informationsteknologi

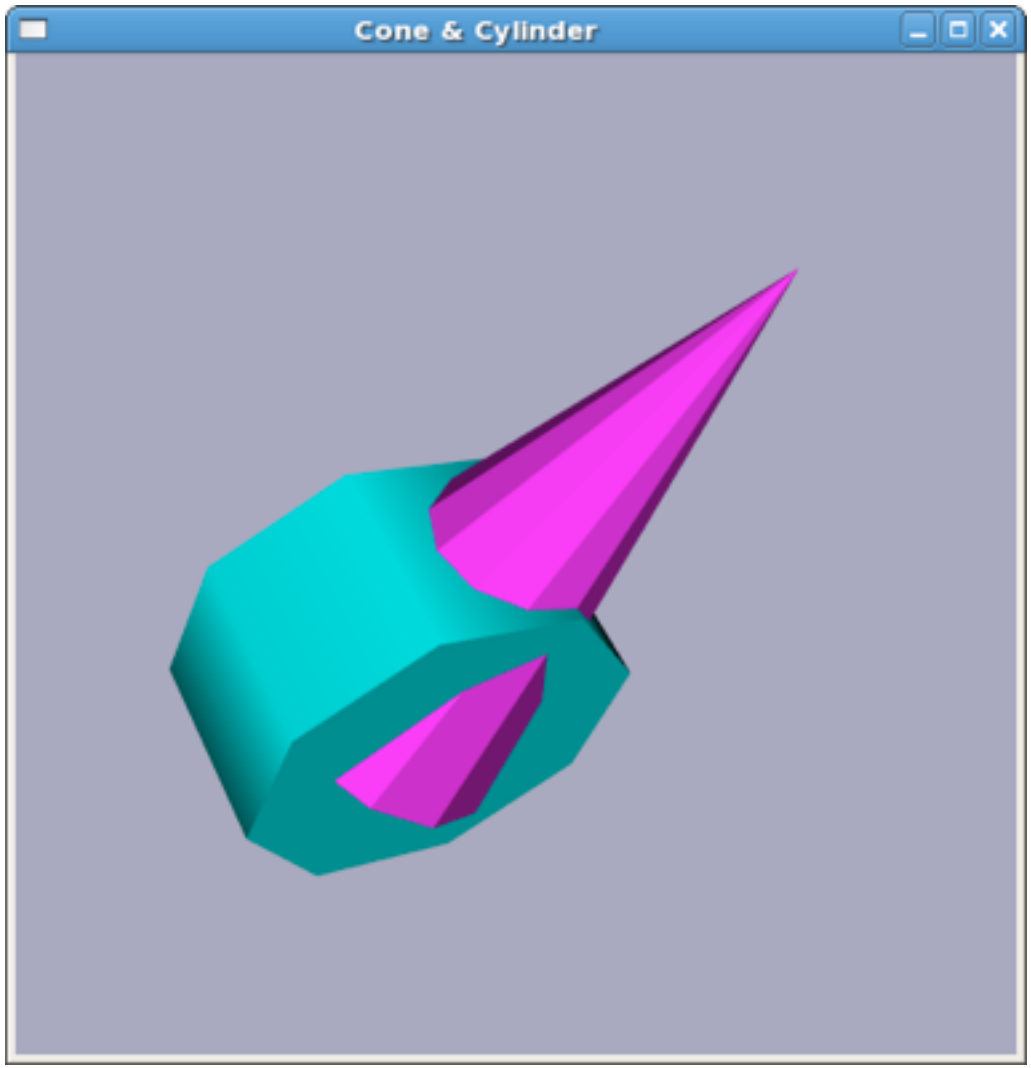
Result



UPPSALA UNIVERSITET

Informationsteknologi

Playing with Actors and Colours



```
from vtk import *

# Create a Cylinder, giving size, resolution and color
cylinder = vtkCylinderSource()
cylinder.SetResolution(8)
cylinder.SetHeight(4)
cylinder.SetRadius(4)

cylinderMapper = vtkPolyDataMapper()
cylinderMapper.SetInput(cylinder.GetOutput())

cylinderActor = vtkActor()
cylinderActor.SetMapper(cylinderMapper)

cylinderActor.GetProperty().SetColor(0.0,1.0,1.0)          #Set the color!
```

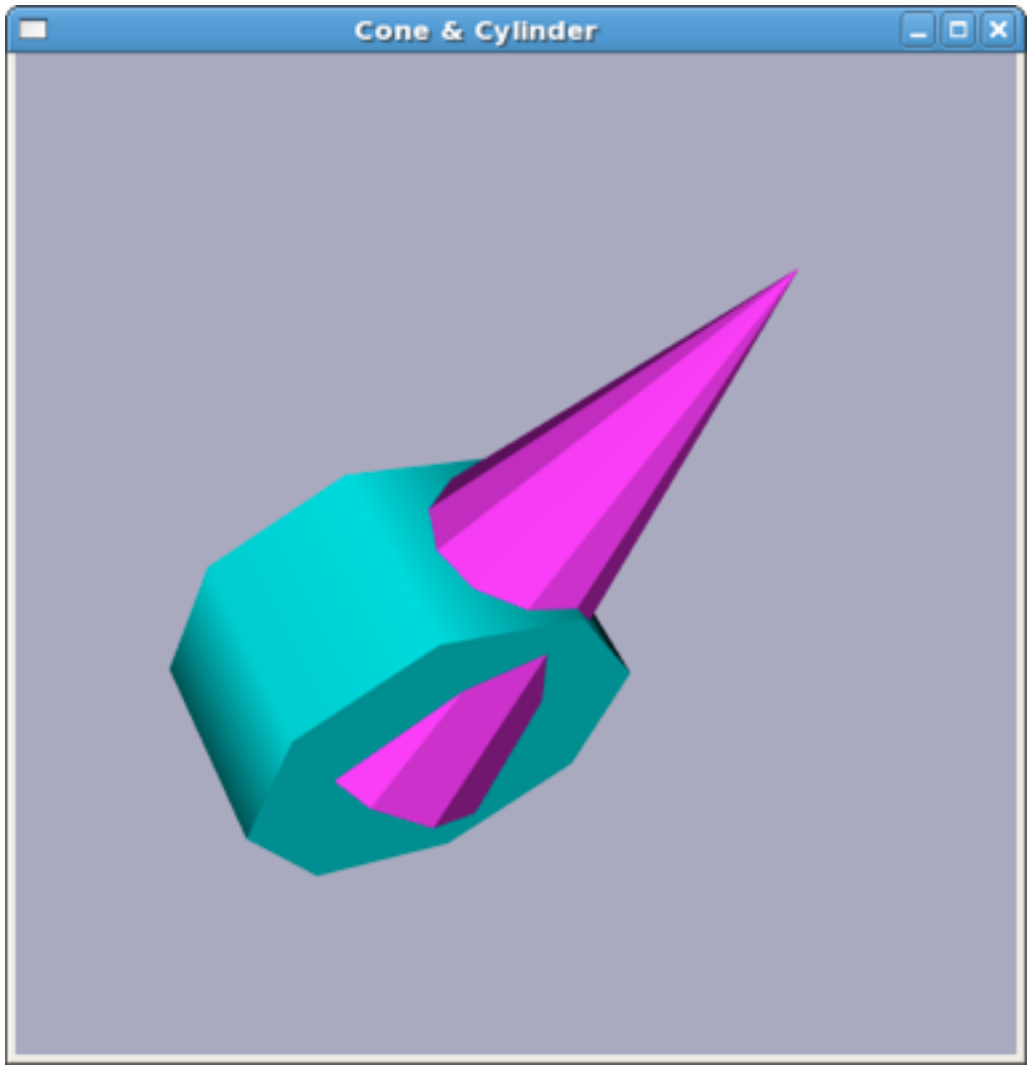
```
# Create a Cone, giving size, resolution, position and color
cone = vtkConeSource()
cone.SetResolution(12)
cone.SetHeight(12)
cone.SetRadius(3)
cone.SetCenter(5,0,0)

coneMapper = vtkPolyDataMapper()
coneMapper.SetInputConnection(cone.GetOutputPort())

coneActor = vtkActor()
coneActor.SetMapper(coneMapper)
coneActor.GetProperty().SetColor(1.0,0.0,1.0)
```

```
# Create a renderer and assign the actors to the renderer
ren = vtkRenderer()

ren.AddActor(cylinderActor)
ren.AddActor(coneActor)

ren.SetBackground(0.6, 0.6, 0.7)

# Create the window and set the name and size of the window
renWin = vtkRenderWindow()
renWin.AddRenderer(ren)
renWin.SetWindowName("Cone & Cylinder")
renWin.SetSize(500,500)

# Make sure that we can interact with the application
iren = vtkRenderWindowInteractor()
iren.SetRenderWindow(renWin)

# Initialze and start the application
iren.Initialize()
iren.Start()
```

UPPSALA UNIVERSITET

Informationsteknologi

Result



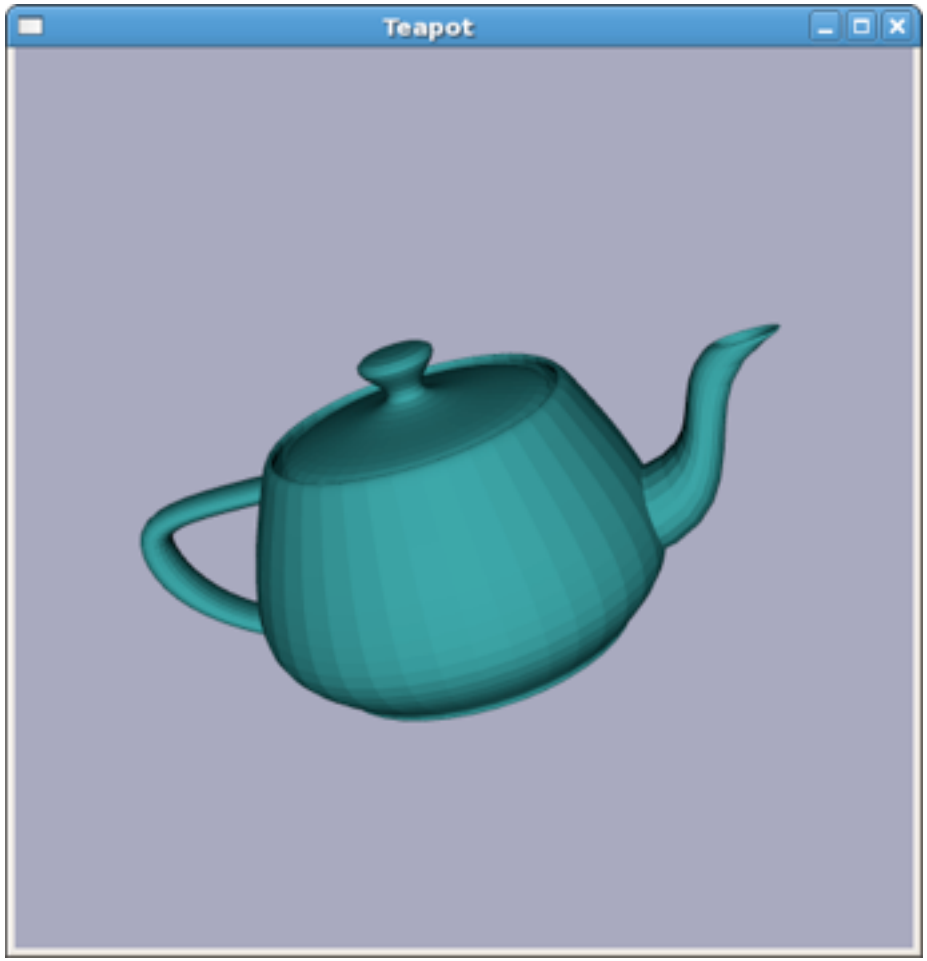
Reading Files

- VTK file format
  - .vtk
- Graphics objects
  - .obj
- Your own data files, use python!
  - Text files
  - Binary files

The Utah Teapot

- teapot.obj
- Ascii

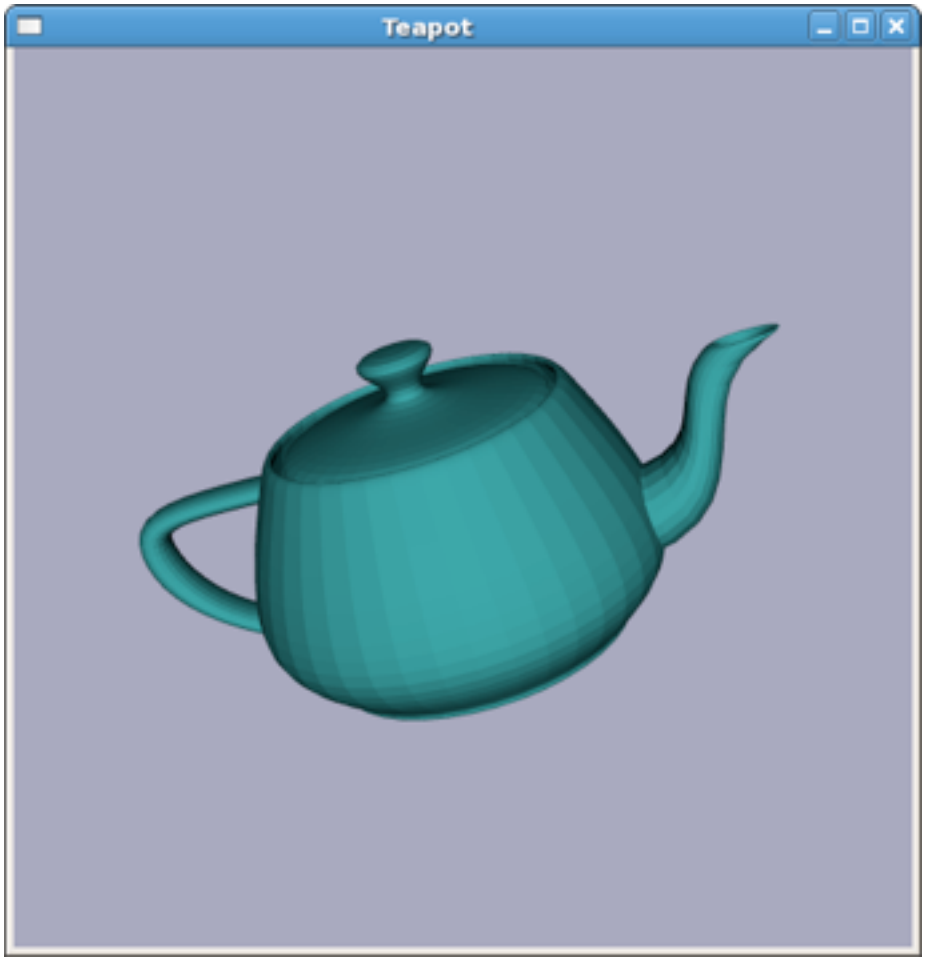
```
v -3.000000 1.800000 0.000000
v -2.991600 1.800000 -0.081000
v -2.991600 1.800000 0.081000
v -2.989450 1.666162 0.000000
v -2.985000 1.921950 0.000000
v -2.985000 1.921950 0.000000
v -2.981175 1.667844 -0.081000
v -2.981175 1.667844 0.081000
v -2.976687 1.920243 -0.081000
v -2.976687 1.920243 0.081000
v -2.968800 1.800000 -0.144000
v -2.968800 1.800000 0.144000
```



```
from vtk import *

# Read the teapot from file
object = vtkOBJReader()
object.SetFileName('teapot.obj')

objectMapper = vtkPolyDataMapper()
objectMapper.SetInputConnection(object.GetOutputPort())


objectActor=vtkActor()
objectActor.SetMapper(objectMapper)
objectActor.GetProperty().SetColor(0.2,0.6,0.6)

ren = vtkRenderer()
ren.AddActor(objectActor)
ren.SetBackground(0.6, 0.6, 0.7)

renWin = vtkRenderWindow()
renWin.AddRenderer(ren)
renWin.SetWindowName("Teapot")
renWin.SetSize(500,500)

iren = vtkRenderWindowInteractor()
iren.SetRenderWindow(renWin)
iren.Initialize()
iren.Start()
```

The usual stuff!

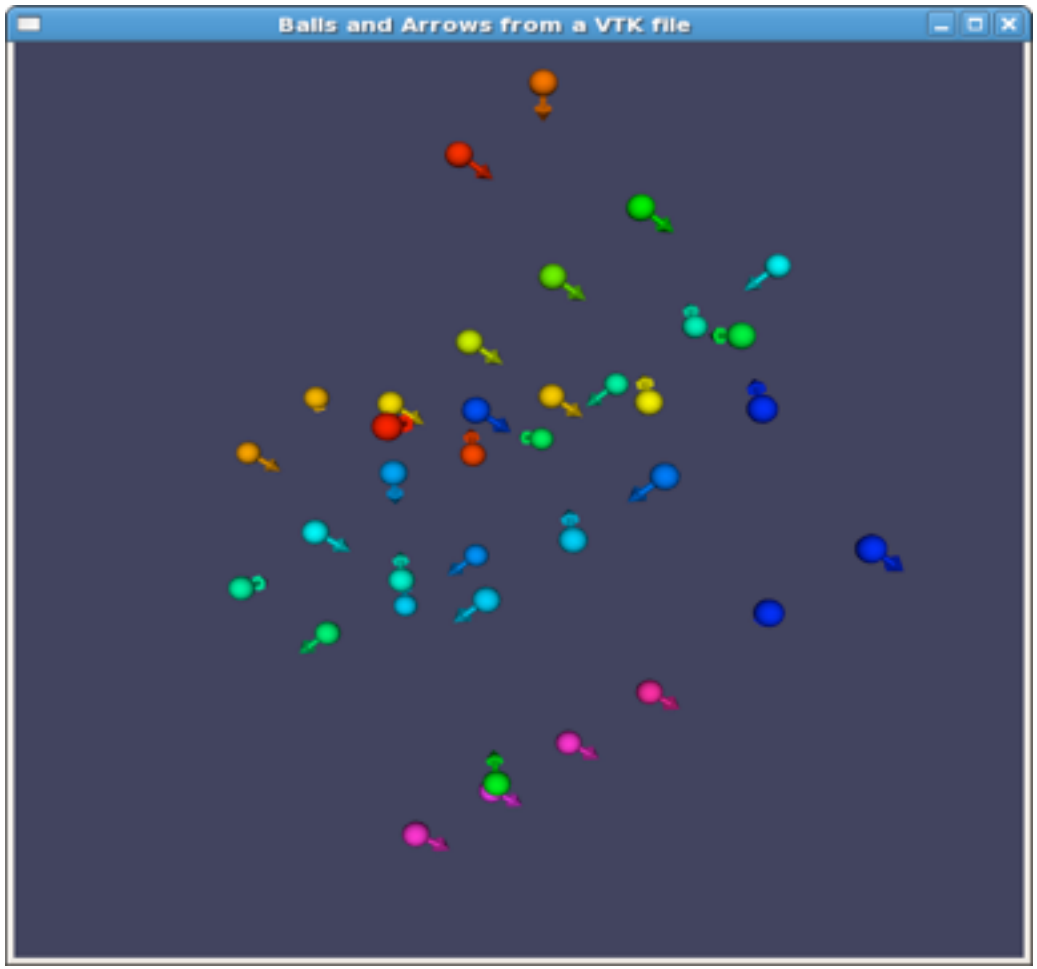
Result



UPPSALA UNIVERSITET

Informationsteknologi

VTK files and Colour Transfer Functions



```
import sys
from vtk import *

# Use the VTK reader to read the vtk file
reader = vtkUnstructuredGridReader()

# Don't forget to give the file name as an argument: "python Vectors.py data.vtk"
reader.SetFileName(sys.argv[1])

# Put spheres at each point in the dataset
ball = vtkSphereSource()
ball.SetRadius(0.12)
ball.SetThetaResolution(12)
ball.SetPhiResolution(12)

ballGlyph = vtkGlyph3D()
ballGlyph.SetSourceConnection(ball.GetOutputPort())
ballGlyph.SetInputConnection(reader.GetOutputPort())

# We do not want the Ball to have the size depending on the Scalar
ballGlyph.SetScaleModeToDataScalingOff()

ballMapper = vtkPolyDataMapper()
ballMapper.SetInputConnection(ballGlyph.GetOutputPort())
```

Colour Transfer Function

```
# Create a color transfer function to be used for both the balls and arrows.
colorTransferFunction = vtkColorTransferFunction()
colorTransferFunction.AddRGBPoint(5.0 , 0.0, 0.0, 1.0)
colorTransferFunction.AddRGBPoint(10.0, 0.0, 1.0, 1.0)
colorTransferFunction.AddRGBPoint(15.0, 0.0, 1.0, 0.0)
colorTransferFunction.AddRGBPoint(20.0, 1.0, 1.0, 0.0)
colorTransferFunction.AddRGBPoint(25.0, 1.0, 0.0, 0.0)
colorTransferFunction.AddRGBPoint(30.0, 1.0, 0.0, 1.0)

# Set colors depending on the color transfer functions
ballMapper.SetLookupTable(colorTransferFunction)
```

```
ballActor = vtkActor()
ballActor.SetMapper(ballMapper)
```

```
#Put an arrow (vector) at each ball
arrow = vtkArrowSource()
arrow.SetTipRadius(0.2)
arrow.SetShaftRadius(0.075)

arrowGlyph = vtkGlyph3D()
arrowGlyph.SetInputConnection(reader.GetOutputPort())
arrowGlyph.SetSourceConnection(arrow.GetOutputPort())
arrowGlyph.SetScaleFactor(0.4)

# We do not want the Arrow's size to depend on the Scalar
arrowGlyph.SetScaleModeToDataScalingOff()

arrowMapper = vtkPolyDataMapper()
arrowMapper.SetInputConnection(arrowGlyph.GetOutputPort())

# Set colors depending on the color transfer functions
arrowMapper.SetLookupTable(colorTransferFunction)

arrowActor = vtkActor()
arrowActor.SetMapper(arrowMapper)
```

UPPSALA UNIVERSITET

Informationsteknologi

As usual...

```
# Create the RenderWindow,Renderer and Interator
ren = vtkRenderer()
ren.AddActor(ballActor)
ren.AddActor(arrowActor)
ren.SetBackground(0.2, 0.2, 0.3)

renWin = vtkRenderWindow()
renWin.AddRenderer(ren)
renWin.SetWindowName("Balls and Arrows from a VTK file")
renWin.SetSize(600,600)

iren = vtkRenderWindowInteractor()
iren.SetRenderWindow(renWin)
iren.Initialize()
iren.Start()
```

```
# vtk DataFile Version 1.0
Unstructured Grid Example
ASCII

DATASET UNSTRUCTURED_GRID
POINTS 36 float
0.0 2.0 3.0
1.0 2.0 3.0
2.0 2.0 3.0
3.0 2.0 3.0
4.0 3.0 3.0
5.0 3.0 3.0
0.0 3.0 3.0
1.0 3.0 3.0
2.0 4.0 3.0
3.0 4.0 3.0
4.0 4.0 4.0
5.0 4.0 4.0
0.0 5.0 4.0
1.0 5.0 4.0
2.0 5.0 4.0
3.0 5.0 4.0
4.0 1.0 4.0
5.0 1.0 4.0
0.0 1.0 4.0
1.0 1.0 4.0
2.0 3.0 2.0
3.0 3.0 2.0
4.0 3.0 2.0
5.0 3.0 2.0
0.0 4.0 2.0
1.0 4.0 2.0
2.0 4.0 2.0
3.0 4.0 2.0
4.0 5.0 2.0
5.0 5.0 2.0
0.0 5.0 1.0
1.0 5.0 1.0
2.0 2.0 1.0
3.0 2.0 5.0
4.0 2.0 5.0
5.0 2.0 5.0
```

Position

```
POINT_DATA 36
SCALARS scalars float
LOOKUP_TABLE default
5.0
7.0
9.0
9.0
11.0
13.0
14.0
20.0
21.0
24.0
7.5
9.0
10.0
11.5
12.0
13.5
14.5
29.0
3.0
4.0
6.0
8.0
10.0
12.0
15.0
17.0
19.0
20.5
21.5
22.0
23.0
24.5
25.0
28.0
29.0
30.0
```

Colors are taken from here

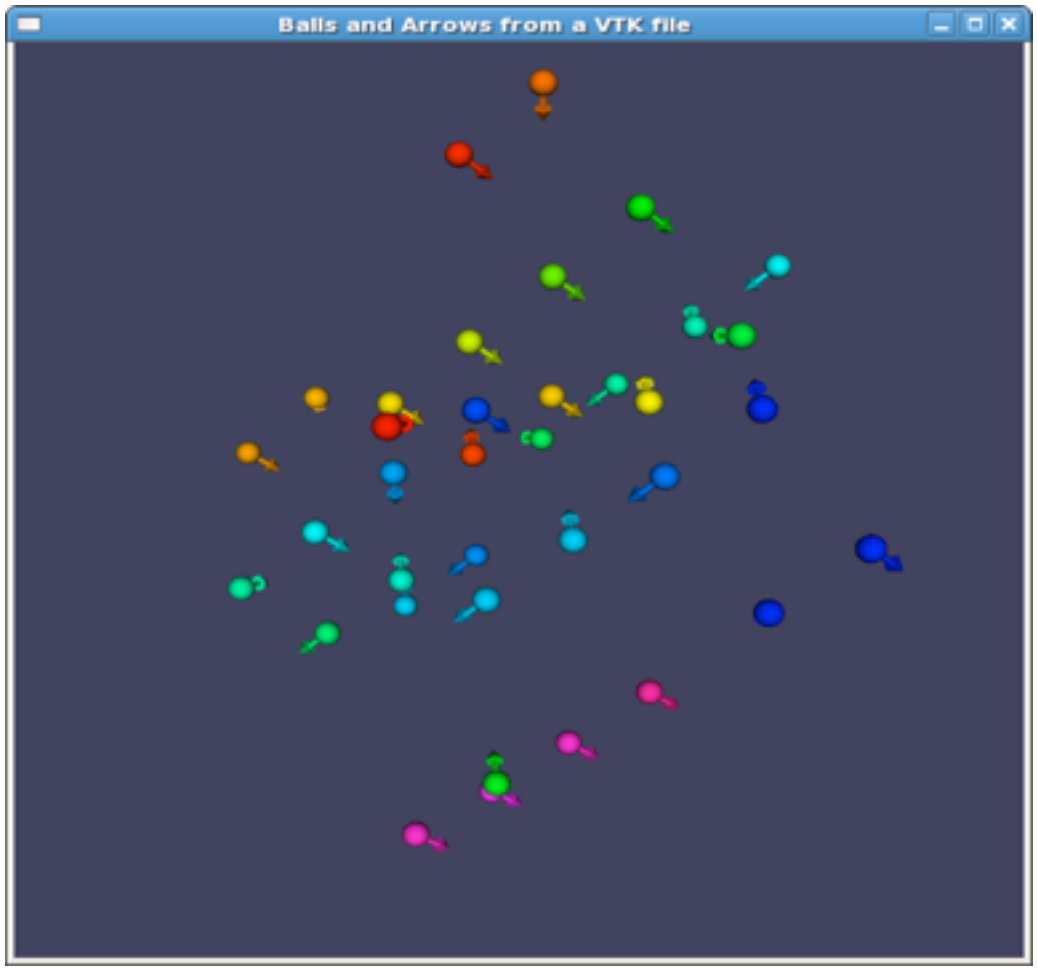
```
VECTORS vectors float
0 1 0
1 0 0
0 1 0
1 0 0
0 1 0
1 0 0
1 1 0
0 1 0
0 0 1
0 1 0
1 0 0
0 1 0
1 0 0
0 1 0
1 0 0
1 1 0
0 1 0
0 0 1
0 0 1
1 1 1
0 0 1
1 0 1
0 0 1
0 1 1
0 0 1
0 0 1
0 0 1
0 0 1
1 1 1
0 0 1
1 0 1
0 0 1
0 1 1
0 0 1
0 0 1
0 0 1
```

Vectors

UPPSALA UNIVERSITET

Informationsteknologi

Result



Glyphs

- vtkGlyph3D is a filter that copies a geometric representation (called a glyph) to every point in the input dataset. The glyph is defined with polygonal data from a source filter input.
- The glyph may be oriented along the input vectors or normals, and it may be scaled according to scalar data or vector magnitude.

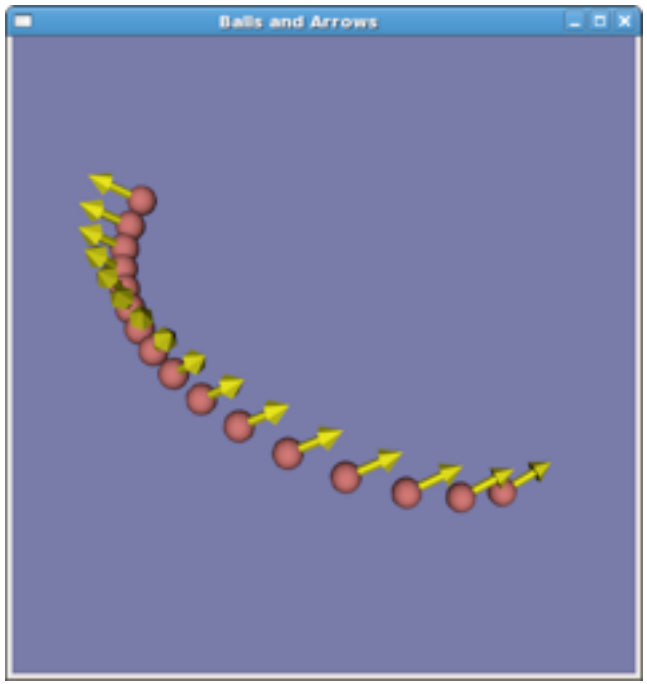
Your own Files

data1.txt

```
# nb =
   0.6744    0.1113    0.7300
   0.5362    0.2407    0.8091
   0.3671    0.3744    0.8515
   0.1866    0.4948    0.8487
   0.0175    0.5904    0.8069
  -0.1259    0.6597    0.7409
  -0.2399    0.7078    0.6644
  -0.3273    0.7412    0.5860
  -0.3929    0.7654    0.5096
  -0.4409    0.7843    0.4364
  -0.4746    0.8005    0.3661
  -0.4958    0.8159    0.2973
  -0.5053    0.8321    0.2287
  -0.5023    0.8501    0.1581
  -0.4845    0.8708    0.0830
  -0.4472    0.8944    0.0000
```

data2.txt

```
# c =
   0.8798    0.0573   -0.4718
   0.9105    0.1226   -0.3950
   0.9299    0.1968   -0.3107
   0.9345    0.2806   -0.2189
   0.9194    0.3746   -0.1200
   0.8781    0.4783   -0.0154
   0.8029    0.5890    0.0916
   0.6864    0.7007    0.1944
   0.5248    0.8028    0.2831
   0.3230    0.8813    0.3449
   0.0976    0.9243    0.3690
  -0.1271    0.9275    0.3515
  -0.3287    0.8966    0.2968
  -0.4943    0.8425    0.2141
  -0.6198    0.7766    0.1128
  -0.7071    0.7071   -0.0000
```



```
import sys
from vtk import *

# This import style makes it possible to write just readPoints
from ReadPoints import *

# Read the data into a vtkPolyData using the functions in ReadPoints.py
data=vtkUnstructuredGrid()

# Read arguments
data.SetPoints(readPoints(sys.argv[1]))
data.GetPointData().SetVectors(readVectors(sys.argv[2]))

# Put spheres at each point in the dataset.
ball = vtkSphereSource()
ball.SetRadius(0.05)
ball.SetThetaResolution(12)
ball.SetPhiResolution(12)

ballGlyph = vtkGlyph3D()
ballGlyph.SetInput(data)
ballGlyph.SetSourceConnection(ball.GetOutputPort())
```

```
ballMapper = vtkPolyDataMapper()
ballMapper.SetInputConnection(ballGlyph.GetOutputPort())

ballActor = vtkActor()
ballActor.SetMapper(ballMapper)
ballActor.GetProperty().SetColor(0.8,0.4,0.4)

arrow = vtkArrowSource()
arrow.SetTipRadius(0.2)
arrow.SetShaftRadius(0.075)

arrowGlyph = vtkGlyph3D()
arrowGlyph.SetInput(data)
arrowGlyph.SetSourceConenction(arrow.GetOutputPort())
arrowGlyph.SetScaleFactor(0.2)

arrowMapper = vtkPolyDataMapper()
arrowMapper.SetInputConnection(arrowGlyph.GetOutputPort()

arrowActor = vtkActor()
arrowActor.SetMapper(arrowMapper)
arrowActor.GetProperty().SetColor(0.9,0.9,0.1)
```

```
ren = vtkRenderer()
ren.AddActor(ballActor)
ren.AddActor(arrowActor)
ren.SetBackground(0.4, 0.4, 0.6)

renWin = vtkRenderWindow()
renWin.AddRenderer(ren)
renWin.SetWindowName("Balls and Arrows")
renWin.SetSize(500,500)

iren = vtkRenderWindowInteractor()
iren.SetRenderWindow(renWin)
iren.Initialize()
iren.Start()
```

```
import string
#Read Points
def readPoints(file):
    # Create an array of Points
    points = vtkPoints()

    #Open the file
    file = open(file)
    # Read one line
    line = file.readline()
    # Loop through lines
    while line:
        # Split the line into data
        data = string.split(line)

        # Skip the commented lines
        if data and data[0] != '#':
            # Convert data into floats
            x, y, z = float(data[0]), float(data[1]), float(data[2])

            # Insert floats into the point array
            points.InsertNextPoint(x, y, z)

        # read next line
        line = file.readline()

    return points;
```

UPPSALA UNIVERSITET

Informationsteknologi

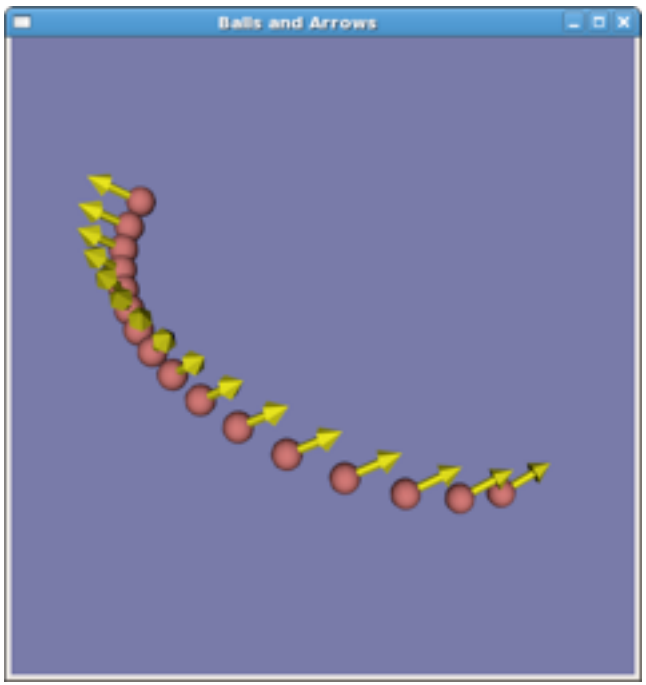
Read Vectors

```
# Read Vectors.
# This method works in the same way as readPoints but
returns a different type of array
def readVectors(file):
    # Create a Double array which represents the vectors
    vectors = vtkDoubleArray()

    # Define number of elements
    vectors.SetNumberOfComponents(3)

    file = open(file)
    line = file.readline()
    while line:
        data = string.split(line)
        if data and data[0] != '#':
            x, y, z = float(data[0]), float(data[1]), float(data[2])
            vectors.InsertNextTuple3(x, y, z)
        line = file.readline()
    return vectors
```

UPPSALA UNIVERSITET

Informationsteknologi

Result



Conclusions

- VTK Contains thousands of Classes
  - However, one can make powerful visualisations with just a few of them
- The pipeline is often
  - source -> filter -> mapper -> actor -> renderer -> renderwindow -> interactor
- Use other programs as templates when writing new programs!