Introduction to VTK

Jordi Campos i Miralles

July 13, 2007

Contents I

Contents II

Appendixes

Part I

Basic

Basic

Introduction

About these slides

This VTK part is strongly based on VTKbook (`http://www.bioimagesuite.org/vtkbook/`), but:

- using Python to show initial vtk examples
- using VTK 5.x (instead of VTK 4.x)

About VTK

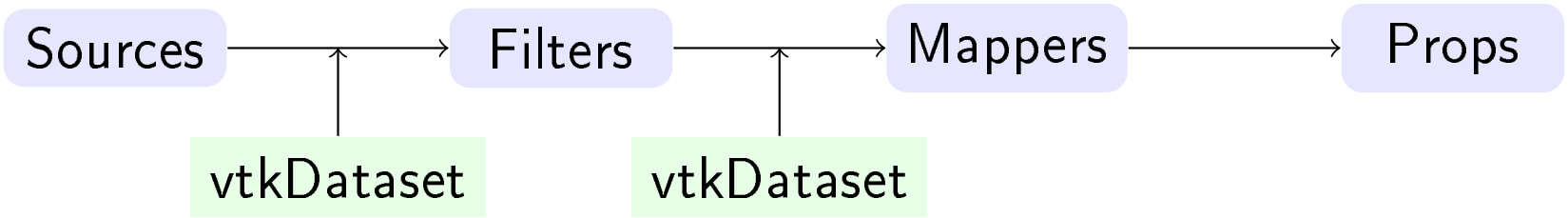
`http://vtk.org`:

- Open source, BSD style license
- High level library; Huge with over 900 classes
- 3D graphics, imaging and visualization
- Implemented in C++ for speed; Python, Tcl and Java wrappers
- Uses OpenGL for rendering
- Cross platform: *nix, Windows, and Mac OS X
- Very powerful with lots of features/functionality
- Pipeline architecture

(extracted from `https://svn.enthought.com/enthought/attachment/wiki/TVTK/tvtk-epc05.pdf`)

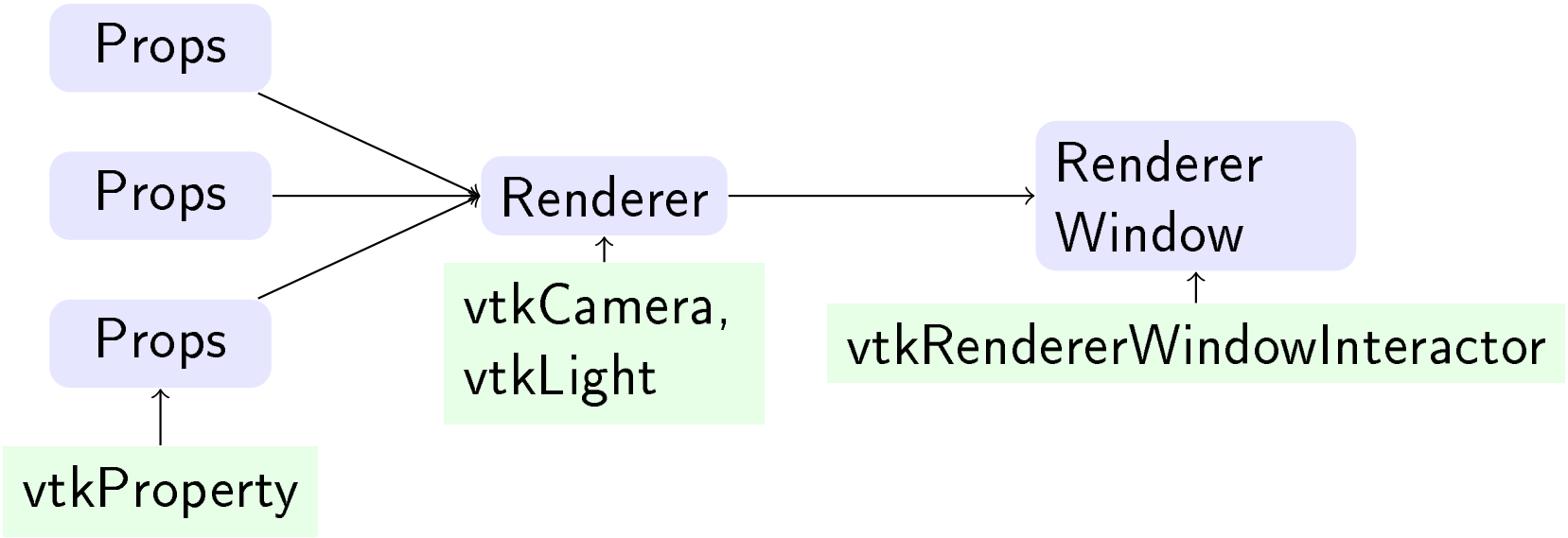
Pipeline

Pipeline: Source → Props



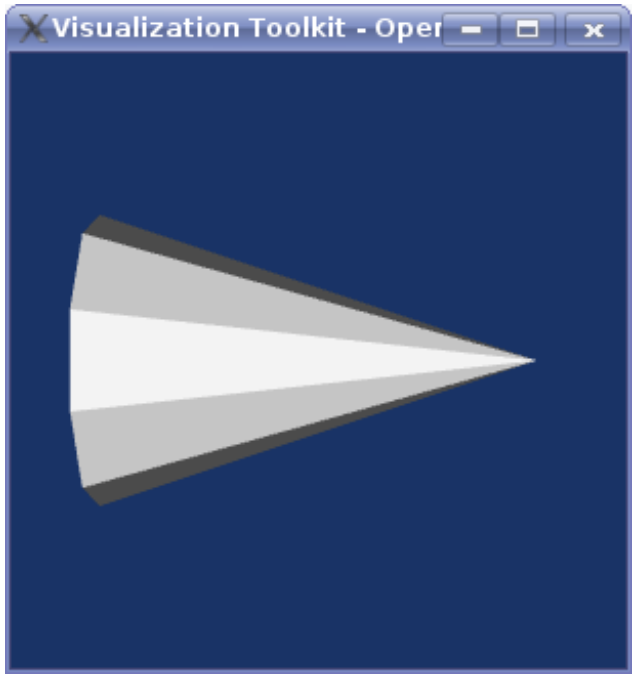
- **Source**: to produce data (i.e. `vtkJPEGReader`).
- **Filters**: to modify data (i.e. `vtkImageGaussianSmooth`).
- **Mappers**: interfaces data → graphic primitives (i.e. `vtkOpenGLPolyDataMapper`).

Pipeline: Props → Render



- **Prop**: visible representation of Mapper *Actor*; special VRender *Volume*.
- **Renderer**: 3D scene → 2D image (also coord. transf).
- **Renderer Window**: piece of screen.

Example: Pipeline - screenshot



Example: Pipeline I

```
#!/usr/bin/python
#                                                    01-Cone.py
import vtk
from    vtk      import *

cone = vtkConeSource()                  # Source
cone.SetHeight( 3.0 )
cone.SetRadius( 1.0 )
cone.SetResolution( 10 )

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

coneActor = vtkActor()                  # Actor
coneActor.SetMapper( coneMapper )

ren = vtkRenderer()                     # Renderer
ren.AddActor( coneActor )
ren.SetBackground( 0.1, 0.2, 0.4 )
ren.ResetCamera()

renWin = vtkRenderWindow()              # Window
renWin.AddRenderer( ren )
renWin.SetSize( 300, 300 )
```

Example: Pipeline II

```
iren = vtkRenderWindowInteractor() # Interactor
iren.SetRenderWindow(renWin)
style = vtkInteractorStyleTrackballCamera()
iren.SetInteractorStyle(style)

iren.Initialize()
iren.Start()
```

(use 'vtkpython 01-Cone.py' to execute the script)

Pipeline execution I

The pipeline does not **compute** anything until:

- a Write() method at the end of the pipeline is called
- a Render() method of a Window is called
- you interact with the Interactor

Remarks:

- a Filter cannot force changes further down the pipeline. It waits an **update request**, an then:
  - request an Update of its Input data
  - create new data (the Output) from the Input data

Pipeline execution II

- **Data flow**: (see subsection *Interrupting the pipeline*)
  - Each filter *may* has its **own copy of some data fragments**, created from its Input data.
  - It is possible to **interrupt the pipeline** and returning intermediate results. It is useful to "use" only a part of the pipeline.

Data structures

3 Data structures

- Data objects
- Data arrays
- Vectorial / Raster data
- Filters

Data Objects

`vtkDataObject`: combination of

- Topology/Geometry:
  - **Cells***(elements)*: specify **Topology** (set of properties invariant under certain geometric transformation
  - **Points***(nodes)*: specify **Geometry** (instantiation of the topology - location)
  - i.e.: 20x20 2D = 400 spaced points connected into square cells.
  - i.e.: `vtkImageData` (dimension, origin, spacing) $\rightarrow$ topology + geomety
- `vtkFieldData`: **data associated** to each point or cell
  - `vtkPointData`/`vtkCellData`: scalar, vector, normals... in `vtkDataArrays`

Data Arrays

`vtkDataArray`: 2D array

- **Component dimension** (column): "type" of data
- **Tuple dimension** (rows): data
- **actual data**: is stored in derived classes, such as `vtkFloatArray`, `vtkUnsignedCharArray` ...
- **Length**: fixed (hi-perf.) vs. dynamic (low-perf.)
- i.e.: 16x16 color image = 256 tuples with 3 components

	R	G	B
1			
...			
n			

Example: DataArray I

03-DataArrayFixed.py

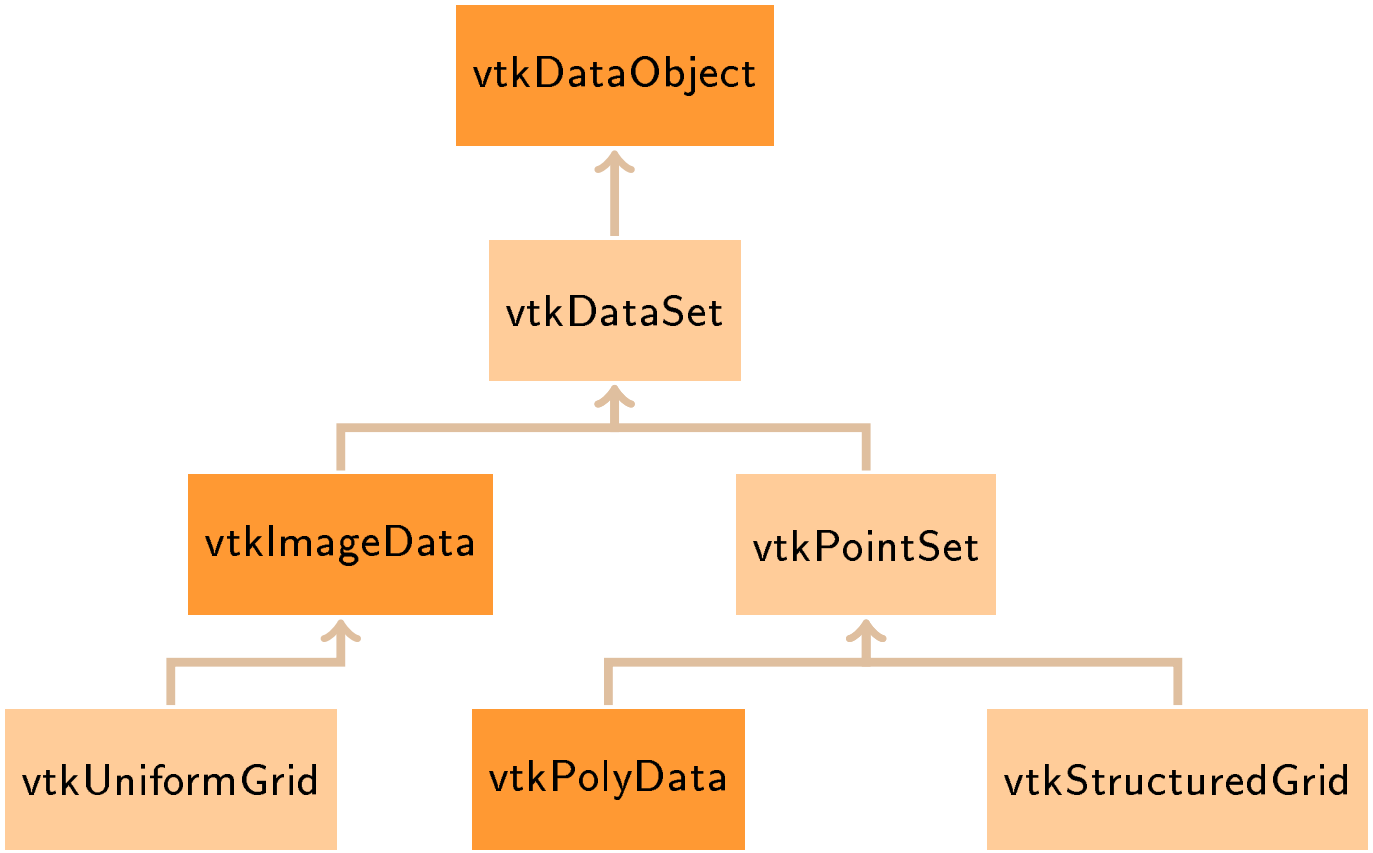
```
#!/usr/bin/python
#
import vtk
from    vtk      import *

arr = vtkFloatArray()
arr.SetNumberOfComponents(2)
arr.SetNumberOfTuples(12)
arr.FillComponent(0, 0.0)
arr.FillComponent(1, 10.0)

arr.SetComponent(10, 0, 3.0)
arr.SetTuple2(11, 9.0, 2.0)
arr.SetComponent(4, 4, -2.1)

for i in range(0, arr.GetNumberOfTuples()):
    print "Tuple %d : (" % i,
    for j in range(0, arr.GetNumberOfComponents()):
        print "\t %1.1f" % arr.GetComponent( i, j),
    print "\t)\n"
```

Example: DataArray II

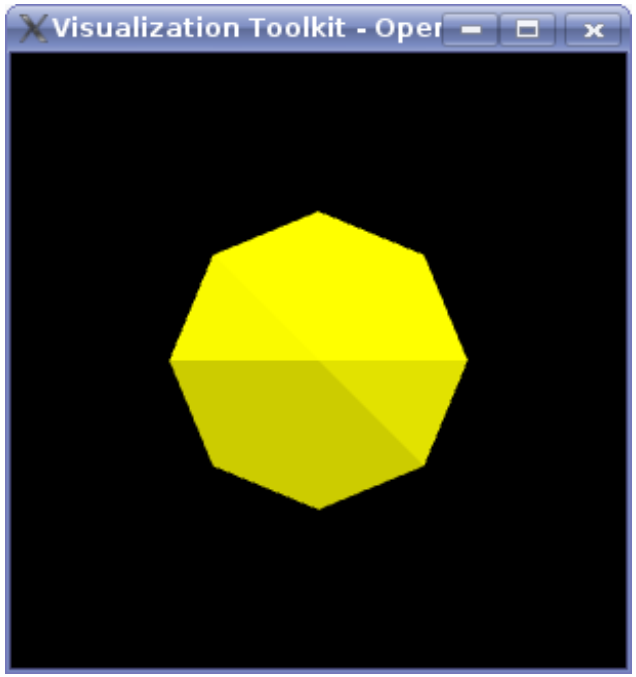
04-DataArrayDynamic.py

```
#!/usr/bin/python
#
import vtk
from    vtk      import *

arr = vtkFloatArray()
arr.SetNumberOfComponents(2)
arr.InsertNextTuple2( 5, 1)
arr.InsertNextTuple2(10, 2)
arr.InsertNextTuple2(20, 3)

for i in range(0, arr.GetNumberOfTuples()):
    print "Tuple %d : (" % i,
    for j in range(0, arr.GetNumberOfComponents()):
        print "\t %1.1f" % arr.GetComponent( i, j),
    print "\t)\n"
```

vtkDataObject descendants



Example: vtkPolyData cone - screenshot



Example: vtkPolyData cone I

```
#!/usr/bin/python
#                                                    02-PolyDataCone.py
import vtk
from    vtk      import *
import math
pts = vtkPoints()                    # Cone base (8 segments)
pts.SetNumberOfPoints(9)
for i in range(0, 8):
  rad =  2.0 * i * math.pi/8.0
  x   = 10.0 *     math.sin(rad) ; y = 10.0 * math.cos(rad)
  pts.SetPoint(i, x, y, 0.0)
pts.SetPoint(8, 0.0, 0.0, 20.0)

triangles = vtkCellArray()           # Triangles from base to apex
triangles.Allocate(10, 5)
for i in range(0, 8):
  p1 = i     ; p2 = (p1 + 1) % 8
  triangles.InsertNextCell(3)    ; triangles.InsertCellPoint(8)
  triangles.InsertCellPoint(p1) ; triangles.InsertCellPoint(p2)

surface = vtkPolyData()              # Source = Cone
surface.SetPoints(pts)
surface.SetPolys(triangles)

map = vtkPolyDataMapper()            # Mapper
```

Example: vtkPolyData cone II

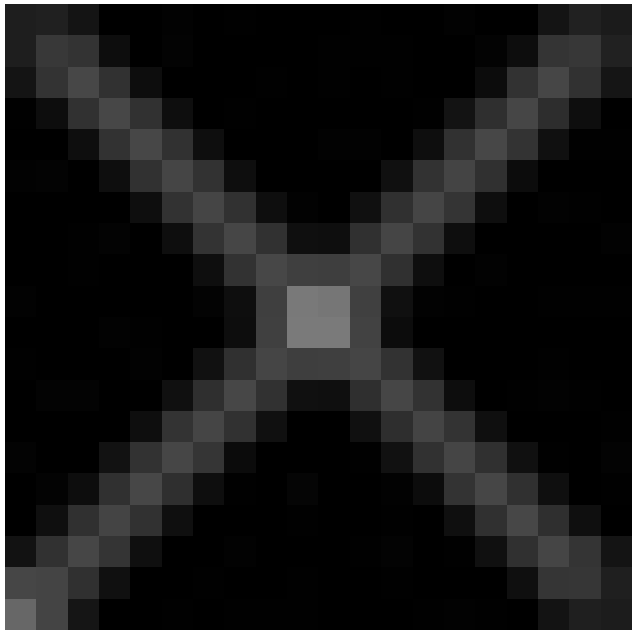
```
map.SetInput(surface)

actor = vtkActor()                    # Actor
actor.SetMapper(map)
property = actor.GetProperty()
property.SetColor(1, 1, 0)
property.SetAmbient(0.8) ; property.SetDiffuse(0.5) ; property.SetSpecular(0.5)

light = vtkLight()                    # Light
light.SetColor(0.5, 0.5, 0.5)
light.SetFocalPoint(0, 0, 0)
light.SetPosition(10, 40, 20)

ren = vtkRenderer()                   # Renderer
ren.AddActor(actor) ; ren.AddLight(light)
ren.ResetCamera()

renWin = vtkRenderWindow()            # Window
renWin.AddRenderer(ren)
renWin.SetSize(300, 300)

iren = vtkRenderWindowInteractor() # Interactor
iren.SetRenderWindow(renWin)
iren.Start()
```

Raster data

`vtkImageData`:

- **Topology/Geometry** derived from:
  - `int Dimensions[3]`: #voxels in x,y,z directions.
  - `double Origin[3]`: first voxel's centroid
  - `double Spacing[3]`: voxel's physical size
- **Data**:
  - Storage: `vtkImageData.vtkPointData.vtkDataArray`
  - Access:
    - Slow: `Get/SetScalarComponentAsDouble`
    - Fast: get specialized array and use `Get/SetComponent`
    - Direct: get pointer to real 1D array (c++ only)
- General comments:
  - implicit **interpolation**: interpolates in-between voxel values.
  - `vtkStructuredPoints` was used to store images before `vtkImageData`.

Example: vtkImageData I



Example: vtkImageData II

05-ImageDataCreateX.py

```
#!/usr/bin/python
#
import vtk

img = vtk.vtkImageData()
img.SetDimensions( 20,   20,    1 )
img.SetSpacing(   1.0,  1.0,  1.0 )
img.SetOrigin(    0.0,  0.0,  0.0 )
img.SetNumberOfScalarComponents( 1 )
img.SetScalarTypeToUnsignedChar()
img.AllocateScalars()

data = img.GetPointData().GetScalars()
data.FillComponent( 0, 0 )

for i in range(0, 19):
        for j in range(0, 19):
                if i == j or i+j == 19:
                        img.SetScalarComponentFromDouble(i, j, 0, 0, 200)

smooth = vtk.vtkImageGaussianSmooth()
smooth.SetInput( img )
smooth.SetStandardDeviations( 1.0, 1.0, 0.0 )
smooth.SetDimensionality( 2 )
smooth.Update()
```

Example: vtkImageData III

```
writer = vtk.vtkJPEGWriter()
writer.SetFileDimensionality( 2 )
writer.SetFileName("smoothedx.jpeg")
writer.SetInput( smooth.GetOutput() )
writer.Write()
```

Filters

All of them derived from `vtkImageToImageFilter`:

- **Smoothing**: `vtkImageGaussianSmooth`, `vtkImageMedian3D`
- **Computing Gradiens/Laplacians**: `vtkImageGradient`, `vtkImageLaplacian`
- **Fourier Operations** : `vtkImageFFT`, `vtkImageRFT`
- **Resampling/Reslicing**: `vtkImageResample`, `vtkImageReslice` (very useful!)
- **Flipping, Permutting**: `vtkImageFlip`, `vtkImagePermute`

Volume visualization

Techniques

- **2D**
  - **Direct slice**: `vtkImageViewer/vtkTkImageViewerWidget`; use bitmaps in actual-size $\longrightarrow$ no zoom!
  - **Texture mapping**: single texture (powers of two, so 257 = 512!)
- **3D**
  - **2D Texture mapping**: n textures with low opacity (pseudo-volume)
  - **3D Volume rendering**: Ray Casting

Example: 2D slices - screenshot

Example: 2D slices - vtkTkImageViewerWidget I

```
#!/usr/bin/python
#                                                    06-ImageViewer.py
import vtk
from    vtk      import *
import Tkinter
from    Tkinter import *
from    vtk.tk.vtkTkImageViewerWidget import vtkTkImageViewerWidget

class MyApp:
  def __init__(self, tkParent):
    self.guiTkParent = tkParent

    image = self.loadImage("brain.vt")

    self.guiCreate(tkParent, image)

  def loadImage(self, imageFileName):
    reader = vtkStructuredPointsReader()
    reader.SetFileName( imageFileName )
    reader.Update()

    image = reader.GetOutput()

    return image
```

Example: 2D slices - vtkTkImageViewerWidget II

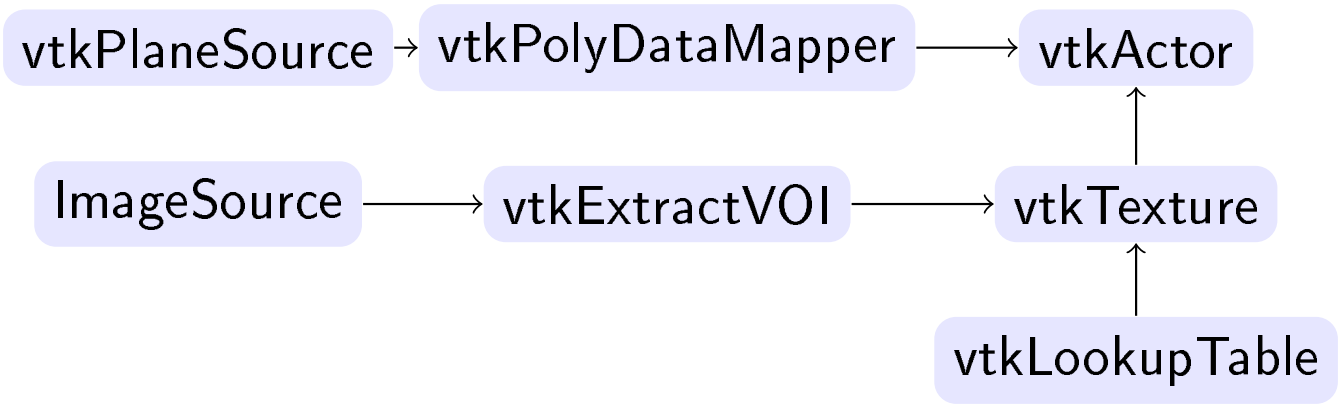
```
def guiCreate(self, tkParent, image):
  topFrame = Tkinter.Frame( tkParent )
  topFrame.pack( side=TOP, expand=TRUE , fill=BOTH)

  imgViewWg = vtkTkImageViewerWidget(topFrame, height=170, width=170)
  imgViewWg.pack(side=TOP, expand=TRUE, fill=BOTH, pady=2)

  imageViewer = imgViewWg.GetImageViewer()
  imageViewer.SetInput(image)
  imageViewer.SetZSlice(0)
  imageViewer.SetColorLevel(128)
  imageViewer.SetColorWindow(255)
  imageViewer.Render()

  bottomFrame = Tkinter.Frame( tkParent )
  bottomFrame.pack(side=TOP, anchor=N, fill=BOTH, expand=FALSE)

  slice = Tkinter.IntVar()
  slice.set( image.GetDimensions()[2] / 2.0 )
  slicescale = Tkinter.Scale( bottomFrame, from_=0, to=image.GetDimensions()[2],
                resolution=1, orient=HORIZONTAL, command=self.guiSetSlice,
                variable=slice, label="Smoothness:")
  slicescale.pack(side=LEFT, fill=X, expand=TRUE)

  quitButton = Tkinter.Button(bottomFrame, text="Exit!", command=self.guiQuit)
  quitButton.pack(side=RIGHT, expand=FALSE, fill=BOTH)
```

Example: 2D slices - vtkTkImageViewerWidget III

```
    self.guiImageViewer = imageViewer

  def guiSetSlice(self, sl):
    self.guiImageViewer.SetZSlice( int(sl) )
    self.guiImageViewer.Render()

  def guiQuit(self):
    self.guiTkParent.quit()

newWindow   = Tkinter.Tk()
myApp       = MyApp(newWindow)
newWindow.mainloop()
```

Slice 2D Texture mapping

Brief: first, extract an slice from the volume (VOI) and create a texture with it; then, the texture is mapped over a plane in the space.



Color maps

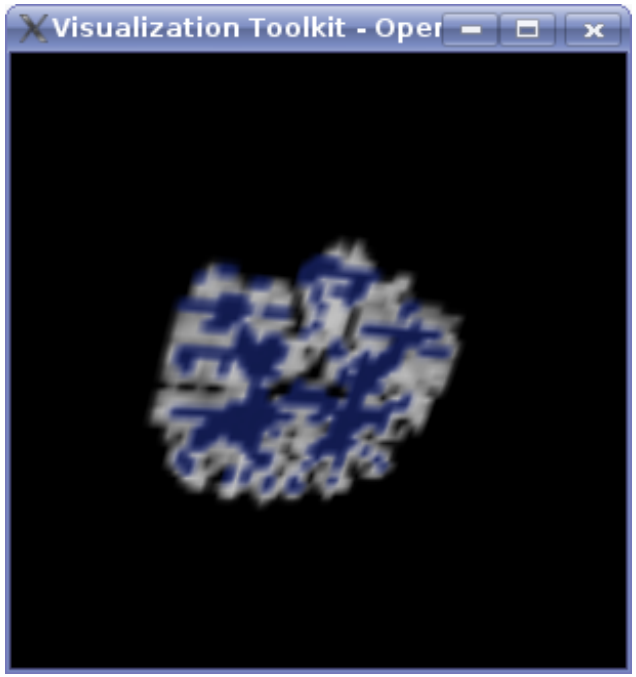
Color maps = Lookup Tables = Transfer Functions: to map image intensities to display colors

- **vtkLookupTable**: N rows, 4 columns (RGBA); L(low):H(hi)

$$i = \begin{cases} 0 & x \leq L \\ N-1 & x \geq H \\ \frac{x-L}{H-L}(N-1) & otherwise \end{cases}$$

- **vtkPiecewiseFunction**: piecewise linear function (controlled by knot points)
- **vtkColorTransferFunction**: 3 channel piecewise linear function

Example: Texture mapped slices - screenshot



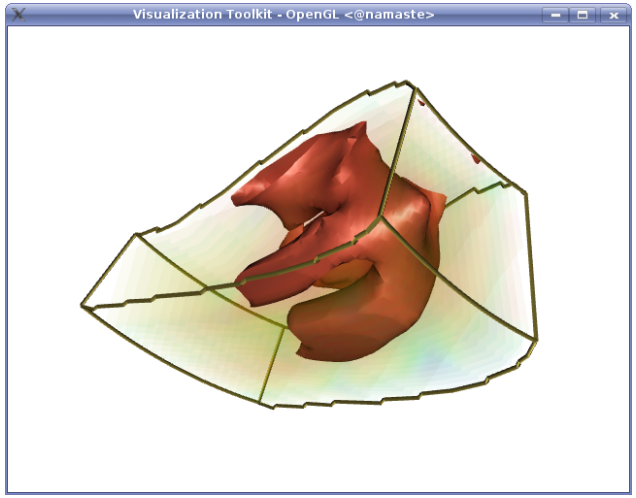
Example: Texture mapped slices I

```
#!/usr/bin/python
#                                                    07-TextureMappedSlice.py
import vtk
from    vtk      import *

reader = vtkStructuredPointsReader()  # Source (=reader)
reader.SetFileName('brain.vt')

voi = vtkExtractVOI()                 # Volume of Interest (=slice)
voi.SetInput(reader.GetOutput())
voi.SetVOI(0, 41, 0, 47, 30, 30)

colormap = vtkLookupTable()           # Color Lookup Table
colormap.SetNumberOfColors(256)
colormap.SetTableRange(0, 255)
for i in range(0, 255):
    v = i /255.0
    colormap.SetTableValue(i, v, v, v, 1.0)

texture = vtkTexture()                # Texture from the VOI
texture.SetInput(voi.GetOutput())
texture.InterpolateOn()
texture.SetLookupTable(colormap)

plane = vtkPlaneSource()              # Plane to project the texture
```

Example: Texture mapped slices II

```
plane.SetXResolution(1)
plane.SetYResolution(1)
plane.SetOrigin(-0.5, -0.5, 30)
plane.SetPoint1(40.5, -0.5, 30)
plane.SetPoint2(-0.5, 46.5, 30)

map = vtkPolyDataMapper()                # Mapper
map.SetInput(plane.GetOutput())

imactor = vtkActor()                     # Actor
imactor.SetMapper(map)
imactor.SetTexture(texture)

ren = vtkRenderer()                      # Renderer
ren.AddActor(imactor)
ren.ResetCamera()

renwin = vtkRenderWindow()               # Window
renwin.AddRenderer(ren)
renwin.SetSize(300, 300)

iren = vtkRenderWindowInteractor()       # Interactor
iren.SetRenderWindow(renwin)
iren.Start()
```

Example: 3D with 2D Textures - screenshot



Example: 3D with 2D Textures I

```
#!/usr/bin/python
#                                                    08-PseudoVolume.py
import vtk
from vtk.util.colors import *

pl3d = vtk.vtkPLOT3DReader()                         # Reader
pl3d.SetXYZFileName("combxyz.bin")
pl3d.SetQFileName("combq.bin")
pl3d.SetScalarFunctionNumber(100)
pl3d.SetVectorFunctionNumber(202)

extract = vtk.vtkExtractGrid()
extract.SetVOI(1, 55, -1000, 1000, -1000, 1000)
extract.SetInput(pl3d.GetOutput())

plane = vtk.vtkPlane()                               # Plane to perform cuts
plane.SetOrigin(0, 4, 2)
plane.SetNormal(0, 1, 0)

cutter = vtk.vtkCutter() # to process each contour value over all cells
cutter.SetInput(extract.GetOutput())
cutter.SetCutFunction(plane)
cutter.GenerateCutScalarsOff()
cutter.SetSortByToSortByCell()
```

Example: 3D with 2D Textures II

```
clut = vtk.vtkLookupTable()                                    # CLUT
clut.SetHueRange(0, .67)
clut.Build()

cutterMapper = vtk.vtkPolyDataMapper()
cutterMapper.SetInput(cutter.GetOutput())
cutterMapper.SetScalarRange(.18, .7)
cutterMapper.SetLookupTable(clut)

cut = vtk.vtkActor()
cut.SetMapper(cutterMapper)
cut.GetProperty().SetOpacity(1)

iso = vtk.vtkContourFilter() # Add in some surface geometry for interest.
iso.SetInput(pl3d.GetOutput())
iso.SetValue(0, .22)
normals = vtk.vtkPolyDataNormals()
normals.SetInput(iso.GetOutput())
normals.SetFeatureAngle(45)
isoMapper = vtk.vtkPolyDataMapper()
isoMapper.SetInput(normals.GetOutput())
isoMapper.ScalarVisibilityOff()
isoActor = vtk.vtkActor()
isoActor.SetMapper(isoMapper)
isoActor.GetProperty().SetDiffuseColor(tomato)
isoActor.GetProperty().SetSpecularColor(white)
```

Example: 3D with 2D Textures III

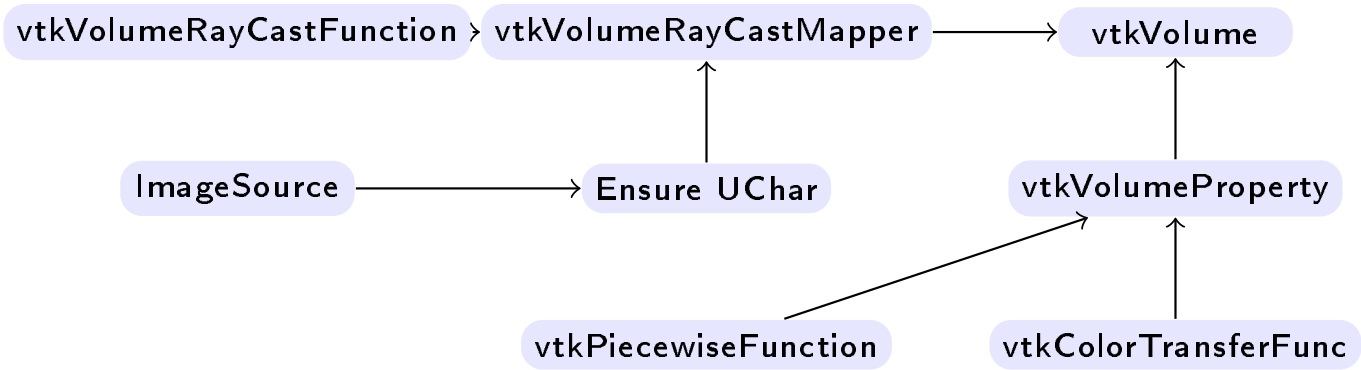
```
isoActor.GetProperty().SetDiffuse(.8)
isoActor.GetProperty().SetSpecular(.5)
isoActor.GetProperty().SetSpecularPower(30)
isoActor.VisibilityOn()

outline = vtk.vtkStructuredGridOutlineFilter()
outline.SetInput(pl3d.GetOutput())
outlineTubes = vtk.vtkTubeFilter()
outlineTubes.SetInput(outline.GetOutput())
outlineTubes.SetRadius(.1)

outlineMapper = vtk.vtkPolyDataMapper()
outlineMapper.SetInput(outlineTubes.GetOutput())
outlineActor = vtk.vtkActor()
outlineActor.SetMapper(outlineMapper)
outlineActor.GetProperty().SetColor(banana)

ren = vtk.vtkRenderer()                                    # Render
ren.AddActor(outlineActor) ; ren.AddActor(isoActor) ; ren.AddActor(cut)
ren.SetBackground(1, 1, 1)
ren.ResetCamera()

renWin = vtk.vtkRenderWindow()                             # Window
renWin.AddRenderer(ren)
renWin.SetSize(640, 480)
```

Example: 3D with 2D Textures IV

```
iren = vtk.vtkRenderWindowInteractor()                    # Interactor
iren.SetRenderWindow(renWin)

cam = ren.GetActiveCamera()
cam.SetClippingRange(3.95297, 50)
cam.SetFocalPoint(9.71821, 0.458166, 29.3999)
cam.SetPosition(2.7439, -37.3196, 38.7167)
cam.ComputeViewPlaneNormal()
cam.SetViewUp(-0.16123, 0.264271, 0.950876)

opacity = .05

def Cut(n): # Cut: generates n cut planes normal to camera's view plane
    global cam, opacity
    plane.SetNormal(cam.GetViewPlaneNormal())
    plane.SetOrigin(cam.GetFocalPoint())
    cutter.GenerateValues(n, -5, 5)
    clut.SetAlphaRange(opacity, opacity)
    renWin.Render()

Cut(20) # Generate 20 cut planes
iren.Start()
```

Volume Ray Casting

Brief: first, data is converted to uchar/short and the Transfer Functions are defined; then, the data is supplied to the mapper and the TF to the Prop; finally all is joined in vtkVolume (vtkActor).

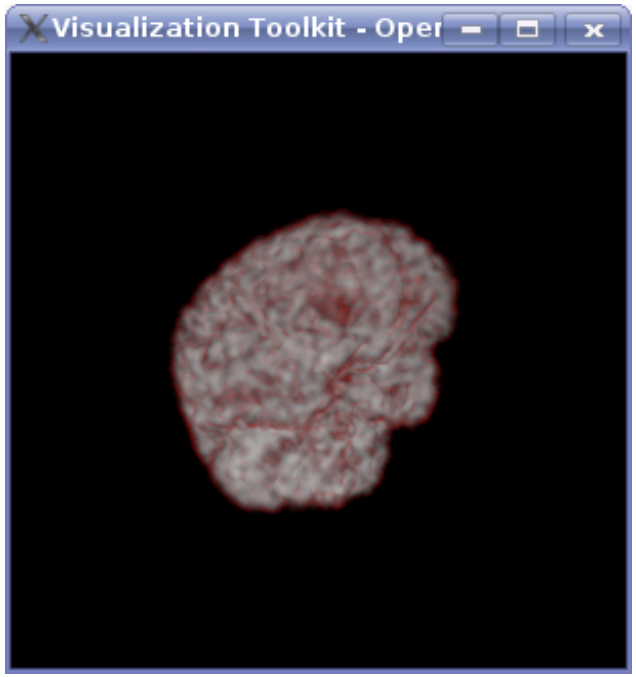


(`vtkVolumeRayCastFunction` controls the sampling rate of the ray-casting)

Volume Ray Casting

- **Software**: `vtkVolumeRayCastMapper`
  - `vtkVolumeRayCastCompositeFunction`:
  - `vtkVolumeRayCastIsosurfaceFunction`:
  - `vtkVolumeRayCastMIPFunction`:
- **Hardware**: `vtkVolumeTextureMapper2D`

Example: Ray Cast. SW - screenshot



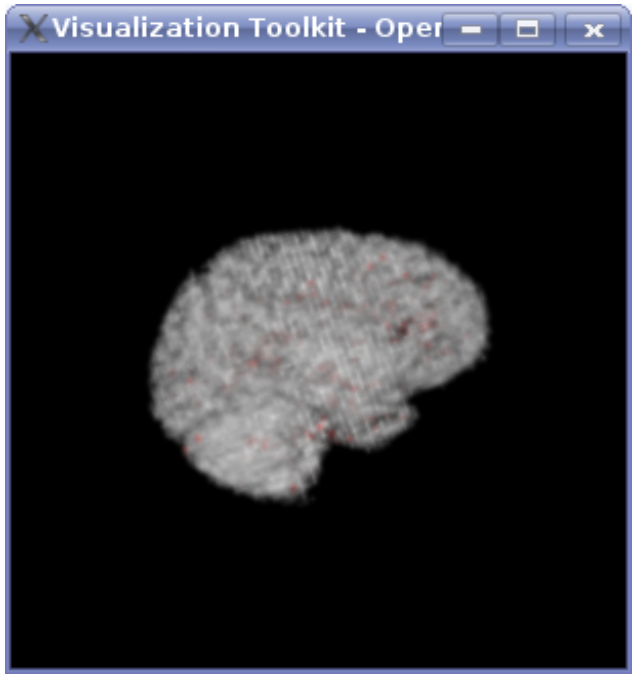
Example: Volume Ray Casting - Software I

```
#!/usr/bin/python
#                                                    09-VolumeRayCast.py
import vtk
from    vtk      import *

reader = vtkStructuredPointsReader()                    # Reader
reader.SetFileName('brain.vt')

alphaTF = vtkPiecewiseFunction()                        # Opacity (A-TF)
alphaTF.AddPoint(  0.0, 0.0)   ; alphaTF.AddPoint( 40.0, 0.0)
alphaTF.AddPoint( 64.0, 0.1)
alphaTF.AddPoint( 70.0, 0.05) ; alphaTF.AddPoint(255.0, 0.05)

colorTF = vtkColorTransferFunction()                    # Color (RGB-TF)
colorTF.AddRGBPoint(  0, 0,   0,   0)   ; colorTF.AddRGBPoint( 40, 0, 0, 0)
colorTF.AddRGBPoint( 64, 1,   0,   0)
colorTF.AddRGBPoint( 70, 0.1, 0.1, 0.1) ; colorTF.AddRGBPoint(255, 1, 1, 1)

volumeProperty = vtkVolumeProperty()                    # Property
volumeProperty.SetColor(colorTF)
volumeProperty.SetScalarOpacity(alphaTF)
volumeProperty.SetInterpolationTypeToLinear()

volumeMapper = vtkVolumeRayCastMapper()                 # Mapper
cast = vtkImageCast()                                   # - uchar conversion
```

Example: Volume Ray Casting - Software II

```
cast.SetInput(reader.GetOutput())                    #  (required by VRayCast)
cast.SetOutputScalarTypeToUnsignedChar()
volumeMapper.SetInput(cast.GetOutput())
compositeFunc = vtkVolumeRayCastCompositeFunction()  # - composite
volumeMapper.SetVolumeRayCastFunction(compositeFunc)

volume = vtkVolume()                                 # Volume=Mapper+Property
volume.SetMapper(volumeMapper)
volume.SetProperty(volumeProperty)

ren = vtkRenderer()                                  # Renderer
ren.AddVolume(volume) ; ren.ResetCamera()

renwin = vtkRenderWindow()                           # Window
renwin.AddRenderer(ren)
renwin.SetSize(300, 300)
renwin.SetDesiredUpdateRate(1.0)

iren = vtkRenderWindowInteractor()                   # Interactor
iren.SetRenderWindow(renwin)
iren.Start()
```

Example: Ray Cast. HD - screenshot



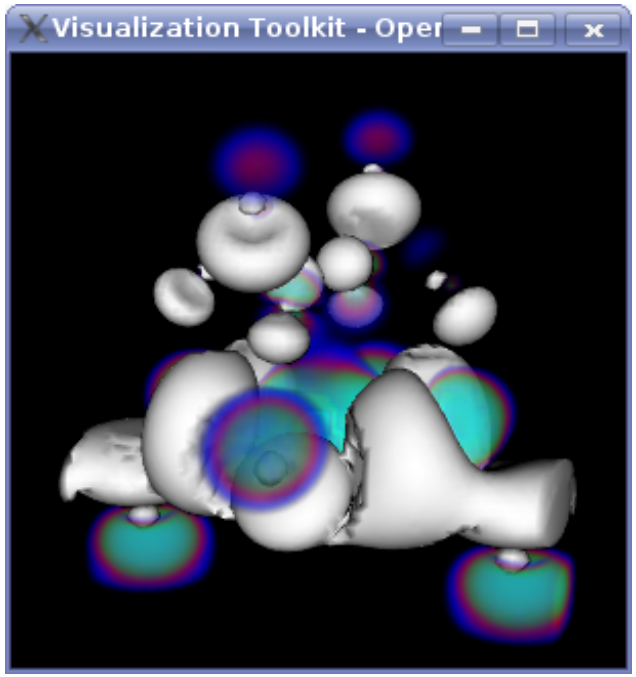
Example: Volume Ray Casting - Hardware I

```
#!/usr/bin/python
#                                                    10-VolumeRayCastHW.py
import vtk
from    vtk      import *

reader = vtkStructuredPointsReader()                    # Reader
reader.SetFileName('brain.vt')

cast = vtkImageCast()                                   # uchar conversion
cast.SetInput(reader.GetOutput())                       #  (required by VRayCast)
cast.SetOutputScalarTypeToUnsignedChar()

alphaTF = vtkPiecewiseFunction()                        # Opacity (A-TF)
alphaTF.AddPoint(  0.0, 0.0)   ; alphaTF.AddPoint( 40.0, 0.0)
alphaTF.AddPoint( 64.0, 0.1)
alphaTF.AddPoint( 70.0, 0.05) ; alphaTF.AddPoint(255.0, 0.05)

colorTF = vtkColorTransferFunction()                    # Color (RGB-TF)
colorTF.AddRGBPoint(  0, 0,   0,   0)    ; colorTF.AddRGBPoint( 40, 0, 0, 0)
colorTF.AddRGBPoint( 64, 1,   0,   0)
colorTF.AddRGBPoint( 70, 0.1, 0.1, 0.1) ; colorTF.AddRGBPoint(255, 1, 1, 1)

volumeProperty = vtkVolumeProperty()                    # Property
volumeProperty.SetColor(colorTF)
volumeProperty.SetScalarOpacity(alphaTF)
```

Example: Volume Ray Casting - Hardware II

```
volumeProperty.SetInterpolationTypeToLinear()

volumeMapper = vtkVolumeTextureMapper2D()                    # Mapper
volumeMapper.SetInput(cast.GetOutput())                      # (Hardware)

volume = vtkVolume()                                         # Volume=Mapper+Property
volume.SetMapper(volumeMapper)
volume.SetProperty(volumeProperty)

ren = vtkRenderer()                                          # Renderer
ren.AddVolume(volume) ; ren.ResetCamera()

renwin = vtkRenderWindow()                                   # Window
renwin.AddRenderer(ren)
renwin.SetSize(300, 300)
renwin.SetDesiredUpdateRate(1.0)

iren = vtkRenderWindowInteractor()                           # Interactor
iren.SetRenderWindow(renwin)
iren.Start()
```

Example: Unstrc. Grid RCast - screenshot



Example: Volume Ray Casting - Unstructured Grid I

```
#!/usr/bin/python
#                                                    10-UGridRayCast.py
import vtk
from    vtk import *

reader = vtkStructuredPointsReader()                 # 1- Source
reader.SetFileName("ironProt.vtk")

thresh = vtkThreshold()                              # 1- Filter
thresh.SetInputConnection( reader.GetOutputPort() )
thresh.ThresholdByUpper(80) ; thresh.AllScalarsOff()

trifilter = vtkDataSetTriangleFilter()
trifilter.SetInputConnection(thresh.GetOutputPort() )

alphaTF = vtkPiecewiseFunction()                     # 1- OpacityTF
alphaTF.AddPoint( 80, 0.0) ; alphaTF.AddPoint(120, 0.2)
alphaTF.AddPoint(255, 0.2)

colorTF = vtkColorTransferFunction()                 # 1- ColorTF
colorTF.AddRGBPoint( 80, 0, 0, 0) ; colorTF.AddRGBPoint(120, 0, 0, 1)
colorTF.AddRGBPoint(160, 1, 0, 0) ; colorTF.AddRGBPoint(200, 0, 1, 0)
colorTF.AddRGBPoint(255, 0, 1, 1)

vProp = vtkVolumeProperty()                          # 1- Property
```

Example: Volume Ray Casting - Unstructured Grid II

```
vProp.SetColor(colorTF) ; vProp.SetScalarOpacity(alphaTF)
vProp.ShadeOff()        ; vProp.SetInterpolationTypeToLinear()

vMapper = vtkUnstructuredGridVolumeRayCastMapper()    # 1- Mapper
vMapper.SetInputConnection(trifilter.GetOutputPort() )

volume = vtkVolume()                                  # 1- Volume
volume.SetMapper(vMapper) ; volume.SetProperty(vProp)

reader2 = vtkSLCReader()                              # 2- Source
reader2.SetFileName("neghip.slc")

contour = vtkContourFilter()                          # 2- Filter
contour.SetInputConnection( reader2.GetOutputPort() )
contour.SetValue(0, 80)

mapper = vtkPolyDataMapper()                          # 2- Mapper
mapper.SetInputConnection(contour.GetOutputPort() )
mapper.ScalarVisibilityOff()

actor = vtkActor()                                    # 2- Actor
actor.SetMapper(mapper)

ren = vtkRenderer()                                   # Renderer
ren.AddViewProp(actor) ; ren.AddVolume(volume)
ren.ResetCamera()      ; cam = ren.GetActiveCamera()
```

Example: Volume Ray Casting - Unstructured Grid III

```
cam.Azimuth(20.0)       ; cam.Elevation(10.0)          ; cam.Zoom(1.5)

renWin = vtkRenderWindow()                             # Window
renWin.AddRenderer(ren) ; renWin.SetSize(300, 300)

iren = vtkRenderWindowInteractor()                     # Interactor
iren.SetRenderWindow(renWin) ; iren.SetDesiredUpdateRate(3)
iren.Start()
```

Part II

Development

Development

Using VTK from C++

CMake

- **CMakeLists.txt**:
  - `PROJECT`: project name
  - `SET`: set of C++ files included in the library
  - `ADD_EXECUTABLE`: create executable from specified source file.
  - `FIND_LIBRARY`, `ADD_LIBRARY`: locate and add a library.
  - `INCLUDE`: include "sub-CMakeLists files"
  - `*_LINK_LIBRARIES`: add vtk modules (`vtkCommon`, `vtkIO`, `vtkFiltering...`)
- **ccmake**: "`ccmake .`"

$$CMakeLists.txt \rightarrow \begin{cases} GNU - Makefiles \\ Ms - VStudioPrj. \\ \dots \end{cases}$$

  - **c**: initial configuration
  - `CMAKE_BUILD_TYPE`: use `Debug` to include debug info.
  - **c**: apply configuration changes
  - **g**: generate and exit

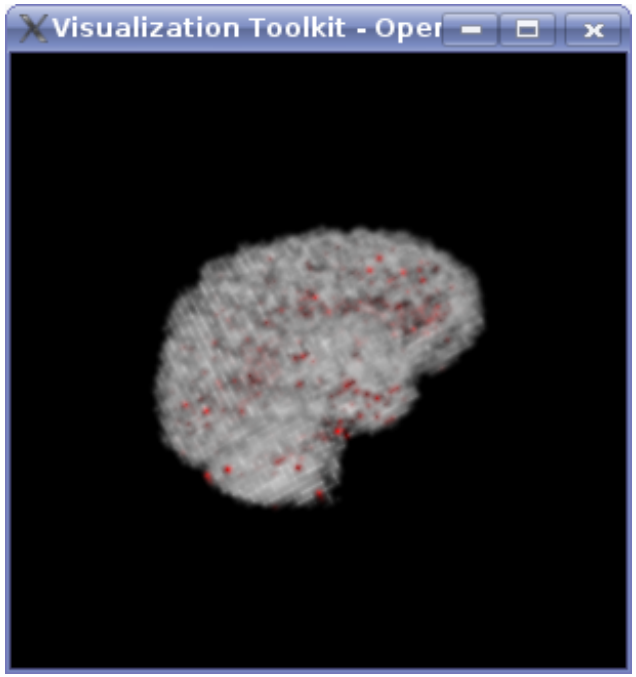
Example: CMakeLists.txt I

```
PROJECT(01-VolumeRayCastHW)

INCLUDE (${CMAKE_ROOT}/Modules/FindVTK.cmake)
FIND_PACKAGE(VTK)
IF (USE_VTK_FILE)
  INCLUDE(${USE_VTK_FILE})
ELSE(USE_VTK_FILE)
  MESSAGE(FATAL_ERROR,"Please specify VTK_DIR")
ENDIF (USE_VTK_FILE)

LINK_LIBRARIES(
vtkCommon
vtkGraphics
vtkIO
vtkRendering
vtkVolumeRendering
)

ADD_EXECUTABLE(01-vraycasthw 01-VolumeRayCastHW.cxx)
```

VTK from C++

- **Protected constructor**: use `<Class>::New()` to activate the reference counting.
- **No constructor parameters**: everything is specified subsequently.
- **Delete method**: use `<Class>::Delete()` to apply reference counting.
  - **Smart pointers**: alternativelly, you can use `vtkSmartPointers` and avoid Delete calls.

Example: RCast HW C++ - screenshot



Example: Volume RayCast with C++ I

01-VolumeRayCastHW.cxx

```
//
#include <vtkStructuredPointsReader.h>
#include <vtkImageCast.h>
#include <vtkPiecewiseFunction.h>
#include <vtkColorTransferFunction.h>
#include <vtkVolumeProperty.h>
#include <vtkVolumeTextureMapper2D.h>
#include <vtkVolume.h>
#include <vtkRenderer.h>
#include <vtkRenderWindow.h>
#include <vtkRenderWindowInteractor.h>
#include <vtkSmartPointer.h>

int main(int argc,char *argv[])
{
  // Reader
  vtkSmartPointer<vtkStructuredPointsReader>
  reader = vtkSmartPointer<vtkStructuredPointsReader>::New();
  reader->SetFileName("brain.vt");

  // uchar conversion  (required by VRayCast);
  vtkSmartPointer<vtkImageCast> cast = vtkSmartPointer<vtkImageCast>::New();
  cast->SetInput( (vtkDataObject *) reader->GetOutput() );
  cast->SetOutputScalarTypeToUnsignedChar();
```

Example: Volume RayCast with C++ II

```
// Opacity (A-TF);
vtkSmartPointer<vtkPiecewiseFunction>
alphaTF = vtkSmartPointer<vtkPiecewiseFunction>::New();
alphaTF->AddPoint(  0.0, 0.0);  alphaTF->AddPoint( 40.0, 0.0);
alphaTF->AddPoint( 64.0, 1.0);
alphaTF->AddPoint( 70.0, 0.05); alphaTF->AddPoint(255.0, 0.05);

// Color (RGB-TF);
vtkSmartPointer<vtkColorTransferFunction>
colorTF = vtkSmartPointer<vtkColorTransferFunction>::New();
colorTF->AddRGBPoint( 0., 0., 0., 0.); colorTF->AddRGBPoint( 40., 0., 0., 0.);
colorTF->AddRGBPoint(64., 1., 0., 0.);
colorTF->AddRGBPoint(70.,0.1,0.1,0.1); colorTF->AddRGBPoint(255., 1., 1., 1.);

// Property
vtkSmartPointer<vtkVolumeProperty>
vProp = vtkSmartPointer<vtkVolumeProperty>::New();
vProp->SetColor(colorTF);
vProp->SetScalarOpacity(alphaTF);
vProp->SetInterpolationTypeToLinear();

// Mapper (Hardware)
vtkSmartPointer<vtkVolumeTextureMapper2D>
vMapper = vtkSmartPointer<vtkVolumeTextureMapper2D>::New();
vMapper->SetInput(cast->GetOutput());
```

Example: Volume RayCast with C++ III

```
  // Volume=Mapper+Property
  vtkSmartPointer<vtkVolume> volume = vtkSmartPointer<vtkVolume>::New();
  volume->SetMapper(vMapper);
  volume->SetProperty(vProp);

  // Renderer
  vtkSmartPointer<vtkRenderer> ren = vtkSmartPointer<vtkRenderer>::New();
  ren->AddVolume(volume);

  // Window
  vtkSmartPointer<vtkRenderWindow>
  renwin = vtkSmartPointer<vtkRenderWindow>::New();
  renwin->AddRenderer(ren);
  renwin->SetSize(300, 300);
  renwin->SetDesiredUpdateRate(1.0);

  // Interactor
  vtkSmartPointer<vtkRenderWindowInteractor>
  iren = vtkSmartPointer<vtkRenderWindowInteractor>::New();
  iren->SetRenderWindow(renwin);
  iren->Start();

  return 0;
}
```

Extending VTK C++

Extending VTK with C++: basis I

Coding Standards: `http://www.vtk.org/Wiki/VTK_Coding_Standards`

- **Naming**: vtk prefix, starting uppercase, full words, uppercase word separation.
- **Style**: "indented brace" style, always use `this`.
- **Methods**:
  - **Constr./Destr.**: protected; constructor without parameters).
  - **Factory**: implement `New()` & `Delete()`.
  - **Macros**: always include vtkTypeRevisionMacro.
  - **PrintSelf**: always define it.
  - **Copy/Assign.**: copy constr./operator= must be private and not implemented.

Extending VTK with C++: basis II

- **Aggregation**: include pointers to other classes, no instances.
- **STL**: you can use the standard library for "internal" implementations.
  - **iostream**: do not use std:: or vtkstd:: prefixes (cerr, cout, ios...); do not include because it is already included.

Macros I

Use **vtkSetGet.h** macros to "automatically" create methods:

```
vtkTypeRevisionMacro(thisClass,superclass): GetClassNameInternal, IsTypeOf, IsA, SafeDownCast...
vtkCxxRevisionMacro(thisClass, revision): MyClass::CollectRevisions(ostream& sos)
vtkStandardNewMacro(thisClass): MyClass::New()
vtkCxxSetObjectMacro(class,name,type): SetAttribute(type *)
vtkSetMacro(name,type) : SetVisibility(int)
vtkSetStringMacro(name,type) : SetFilename(char *)
vtkSetClampMacro(name,type,min,max): SetRadius(int), GetRadiusMinValue(), GetRadiusMaxValue()
vtkBooleanMacro(name,type): DebugOn() DebugOff()
vtkSetVector3Macro(name,type):  SetColor(float,float,float) and SetColor(float* rgb[2]).
vtkSetVectorMacro(name,type,count): SetColor(c,3)
vtkDebugMacro(x): vtkDebugMacro(≪ "Debug msg" ≪ this->var)
vtkErrorMacro(x): vtkErrorMacro(≪ "Error msg" ≪ var)
vtkErrorWithObjectMacro(self, x): vtkErrorWithObjectMacro(self, ≪ "msg" ≪ var)
vtkWorldCoordinateMacro(name): SetPos(double x[3]), double *GetPos()...
vtkViewportCoordinateMacro(name): SetPos(double x[3]), double *GetPos()...
vtkImageScalarTypeNameMacro(type): returns string with the name of 'type'
```

Example: Filter - Extract Volume of Interest - .h I

```
//                                                    02-vtkLocalExample.h
#ifndef __vtkLocalExample_h
#define __vtkLocalExample_h

#include "vtkLocalConfigure.h" // Include configuration header.
#include "vtkImageAlgorithm.h"

class VTK_vtkLocal_EXPORT vtkLocalExample : public vtkImageAlgorithm
{
public:
  static vtkLocalExample* New();
  vtkTypeRevisionMacro(vtkLocalExample, vtkObject);
  void PrintSelf(ostream& os, vtkIndent indent);

  vtkSetVector6Macro(VOI,int); // Specify i-j-k (min,max) pairs to extract
  vtkGetVectorMacro (VOI,int,6);

  vtkSetVector3Macro(SampleRate, int); // Sampling rate in the i,j,k directions
  vtkGetVectorMacro (SampleRate, int, 3);

protected:
   vtkLocalExample();
  ~vtkLocalExample() {};

  virtual int RequestUpdateExtent(vtkInformation*, vtkInformationVector**,
```

Example: Filter - Extract Volume of Interest - .h II

```
                                                    vtkInformationVector*);
  virtual int RequestInformation (vtkInformation*, vtkInformationVector**,
                                                    vtkInformationVector*);
  virtual int RequestData         (vtkInformation*, vtkInformationVector**,
                                                    vtkInformationVector*);

  int VOI[6];
  int SampleRate[3];

private:
  vtkLocalExample(const vtkLocalExample&);  // Not implemented.
  void  operator=(const vtkLocalExample&);  // Not implemented.
};

#endif
```

Example: Filter - Extract Volume of Interest - .cxx I

02-vtkLocalExample.cxx

```
//
#include "vtkLocalExample.h"

#include "vtkCellData.h"
#include "vtkImageData.h"
#include "vtkInformation.h"
#include "vtkInformationVector.h"
#include "vtkObjectFactory.h"
#include "vtkStreamingDemandDrivenPipeline.h"
#include "vtkPointData.h"

vtkCxxRevisionMacro(vtkLocalExample, "$Revision: 1.2 $");
vtkStandardNewMacro(vtkLocalExample);

vtkLocalExample::vtkLocalExample()
{
  this->VOI[0] = this->VOI[2] = this->VOI[4] = -VTK_LARGE_INTEGER;
  this->VOI[1] = this->VOI[3] = this->VOI[5] = VTK_LARGE_INTEGER;
  this->SampleRate[0] = this->SampleRate[1] = this->SampleRate[2] = 1;
}

int vtkLocalExample::RequestUpdateExtent (
                    vtkInformation*          vtkNotUsed(request),
                    vtkInformationVector**   inputVector,
                    vtkInformationVector*    vtkNotUsed( outputVector ))
```

Example: Filter - Extract Volume of Interest - .cxx II

```
{
  vtkInformation *inInfo = inputVector[0]->GetInformationObject(0);

  // request all of the VOI
  int i;
  int inExt[6];
  inInfo->Get(vtkStreamingDemandDrivenPipeline::WHOLE_EXTENT(),inExt);

  for (i = 0; i < 3; ++i) // no need to go outside the VOI
    {
    if (inExt[i*2]   < this->VOI[i*2])   inExt[i*2]   = this->VOI[i*2];
    if (inExt[i*2+1] > this->VOI[i*2+1]) inExt[i*2+1] = this->VOI[i*2+1];
    }
  inInfo->Set(vtkStreamingDemandDrivenPipeline::UPDATE_EXTENT(),inExt,6);

  return 1;
}

int vtkLocalExample::RequestInformation (
                    vtkInformation*          vtkNotUsed(request),
                    vtkInformationVector**   inputVector,
                    vtkInformationVector*    outputVector)
{
  vtkInformation *outInfo = outputVector->GetInformationObject(0);
  vtkInformation *inInfo  = inputVector[0]->GetInformationObject(0);
```

Example: Filter - Extract Volume of Interest - .cxx III

```
int i, outDims[3], voi[6];
int rate[3];
int wholeExtent[6];
inInfo->Get(vtkStreamingDemandDrivenPipeline::WHOLE_EXTENT(), wholeExtent );

double ar[3], outAR[3];
inInfo->Get(vtkDataObject::SPACING(), ar );

double origin[3], outOrigin[3];
inInfo->Get(vtkDataObject::ORIGIN(), origin );

for ( i=0; i < 6; i++ ) voi[i] = this->VOI[i];

for ( i=0; i < 3; i++ )
  {
  if      ( voi[2*i+1] > wholeExtent[2*i+1] ) voi[2*i+1] = wholeExtent[2*i+1];
  else if ( voi[2*i+1] < wholeExtent[2*i] )   voi[2*i+1] = wholeExtent[2*i];

  if      ( voi[2*i]   < wholeExtent[2*i] )   voi[2*i] = wholeExtent[2*i];
  else if ( voi[2*i]   > wholeExtent[2*i+1] ) voi[2*i] = wholeExtent[2*i+1];

  if      ( voi[2*i]   > voi[2*i+1] )         voi[2*i] = voi[2*i+1];

  if ( (rate[i] = this->SampleRate[i]) < 1 ) rate[i] = 1;

  outDims[i] = (voi[2*i+1] - voi[2*i]) / rate[i] + 1;
```

Example: Filter - Extract Volume of Interest - .cxx IV

```
    if ( outDims[i] < 1 ) outDims[i] = 1;

    outAR[i] = ar[i] * this->SampleRate[i];
    wholeExtent[i*2]   = voi[i*2];
    wholeExtent[i*2+1] = voi[i*2] + outDims[i] - 1;
    outOrigin[i] = origin[i] + voi[2*i]*ar[i] - wholeExtent[2*i]*outAR[i];
    }

  outInfo->Set(vtkStreamingDemandDrivenPipeline::WHOLE_EXTENT(), wholeExtent ,6);
  outInfo->Set(vtkDataObject::SPACING(), outAR, 3);
  outInfo->Set(vtkDataObject::ORIGIN(), outOrigin, 3);

  return 1;
}


int vtkLocalExample::RequestData(
                    vtkInformation*          vtkNotUsed( request ),
                    vtkInformationVector**   inputVector,
                    vtkInformationVector*    outputVector)
{
  vtkInformation   *outInfo = outputVector->GetInformationObject(0);
  vtkImageData     *output  = vtkImageData::SafeDownCast(
    outInfo->Get(vtkDataObject::DATA_OBJECT()));

  vtkInformation   *inInfo   = inputVector[0]->GetInformationObject(0);
```

Example: Filter - Extract Volume of Interest - .cxx V

```
vtkImageData     *input    = vtkImageData::SafeDownCast(
  inInfo->Get(vtkDataObject::DATA_OBJECT()));

vtkPointData     *pd       = input->GetPointData();
vtkCellData      *cd       = input->GetCellData();
output->SetExtent(output->GetWholeExtent());
vtkPointData     *outPD    = output->GetPointData();
vtkCellData      *outCD    = output->GetCellData();
int i, j, k, dims[3], outDims[3], voi[6], dim, idx, newIdx;
int newCellId;
double origin[3], ar[3];
int sliceSize, outSize, jOffset, kOffset, rate[3];
int wholeExtent[6];
inInfo->Get(vtkStreamingDemandDrivenPipeline::WHOLE_EXTENT(),wholeExtent);
int *inExt = input->GetExtent();

vtkDebugMacro(<< "Extracting VOI");
input->GetDimensions(dims);
input->GetOrigin(origin);
input->GetSpacing(ar);

// Check VOI and clamp as necessary. Compute output parameters,
for ( i=0; i < 6; i++ ) voi[i] = this->VOI[i];

for ( outSize=1, dim=0, i=0; i < 3; i++ )
  {
```

Example: Filter - Extract Volume of Interest - .cxx VI

```
    if       ( voi[2*i+1] > wholeExtent[2*i+1] ) voi[2*i+1] = wholeExtent[2*i+1];
    else if  ( voi[2*i+1] < wholeExtent[2*i] )   voi[2*i+1] = wholeExtent[2*i];

    if       ( voi[2*i]   < wholeExtent[2*i] )   voi[2*i] = wholeExtent[2*i];
    else if  ( voi[2*i]   > wholeExtent[2*i+1] ) voi[2*i] = wholeExtent[2*i+1];

    if       ( voi[2*i]   > voi[2*i+1] )         voi[2*i] = voi[2*i+1];

    if       ( (voi[2*i+1]-voi[2*i]) > 0 ) dim++;

    if ( (rate[i] = this->SampleRate[i]) < 1 ) rate[i] = 1;

    outDims[i] = (voi[2*i+1] - voi[2*i]) / rate[i] + 1;
    if ( outDims[i] < 1 ) outDims[i] = 1;

    outSize *= outDims[i];
    }

  // If output same as input, just pass data through
  if ( outDims[0] == dims[0] && outDims[1] == dims[1] && outDims[2] == dims[2]
       && rate[0] == 1 && rate[1] == 1 && rate[2] == 1 )
    {
    output->GetPointData()->PassData(input->GetPointData());
    output->GetCellData()->PassData(input->GetCellData());
    vtkDebugMacro(<<"Passed data through because input and output are the same");
    return 1;
```

Example: Filter - Extract Volume of Interest - .cxx VII

```
  }

outPD->CopyAllocate(pd,outSize,outSize); // Allocate necessary objects
outCD->CopyAllocate(cd,outSize,outSize);
sliceSize = dims[0]*dims[1];

newIdx = 0; // Copy POINT ATTRIBUTES to output
for ( k=voi[4]; k <= voi[5]; k += rate[2] )
  {
  kOffset = (k-inExt[4]) * sliceSize;
  for ( j=voi[2]; j <= voi[3]; j += rate[1] )
    {
    jOffset = (j-inExt[2]) * dims[0];
    for ( i=voi[0]; i <= voi[1]; i += rate[0] )
      {
      idx = (i-inExt[0]) + jOffset + kOffset;
      outPD->CopyData(pd, idx, newIdx++);
      }
    }
  }

// Handle 2D, 1D and 0D degenerate data sets.
if (voi[5] == voi[4]) ++voi[5];
if (voi[3] == voi[2]) ++voi[3];
if (voi[1] == voi[0]) ++voi[1];
```

Example: Filter - Extract Volume of Interest - .cxx VIII

```
  newCellId = 0; // Copy CELL ATTRIBUTES to output
  sliceSize = (dims[0]-1)*(dims[1]-1);
  for ( k=voi[4]; k < voi[5]; k += rate[2] )
    {
    kOffset = (k-inExt[4]) * sliceSize;
    for ( j=voi[2]; j < voi[3]; j += rate[1] )
      {
      jOffset = (j-inExt[2]) * (dims[0] - 1);
      for ( i=voi[0]; i < voi[1]; i += rate[0] )
        {
        idx = (i-inExt[0]) + jOffset + kOffset;
        outCD->CopyData(cd, idx, newCellId++);
        }
      }
    }

  vtkDebugMacro(<<"Extracted " << newIdx << " point attributes on "
  << dim << "-D dataset\n\tDimensions are (" << outDims[0]
  << "," << outDims[1] << "," << outDims[2] <<")");

  return 1;
}


void vtkLocalExample::PrintSelf(ostream& os, vtkIndent indent)
{
```

Example: Filter - Extract Volume of Interest - .cxx IX

```
  this->Superclass::PrintSelf(os,indent);

  os << indent << "VOI:\n";
  os << indent << "Dim(" << this->VOI[0] << ".." << this->VOI[1];
  os << ", "             << this->VOI[2] << ".." << this->VOI[3];
  os << ", "             << this->VOI[4] << ".." << this->VOI[5] << ")\n";

  os << indent << "Rate("     << this->SampleRate[0];
  os << ", "                  << this->SampleRate[1];
  os << ", "                  << this->SampleRate[1] << ")\n";
}
```

Example: Filter - Extract Volume of Interest - .py I

```
#!/usr/bin/python
#                                                    02-vtkLocalTest.py

# execution:
# PYTHONPATH=$PYTHONPATH:<pathto-libvtkLocalPython.so> vtkpython vtkLocalTest.py

import vtk
from    vtk      import *

import os

if os.name == 'posix':
    from libvtkLocalPython import *
else:
    from vtkLocalPython import *

l = vtkLocalExample()
print l
```

Interrupting the pipeline I

Copying: `vtkPolyData`, `vtkImageData`...

- **CopyStructure**: copies geometric structure, but not contents (it does not allocate memory). Useful to prepare *output* data.

```
vtkImageData* out = vtkImageData::New();
out->CopyStructure(input);
out->SetScalarTypeToFloat(); // Modify as desired
out->AllocateScalars(); // Allocate memory
```

- **ShallowCopy**: like *CopyStructure* plus all array data is linked. Useful for preserving the results.

```
vtkContourFilter *ContourFilter = vtkContourFilter::New();
ContourFilter->SetInput(inputImageData);
ContourFilter->SetValue(1, 0.0);
ContourFilter->Update();
vtkPolyData* zerosurface = vtkPolyData::New();
zerosurface->ShallowCopy(ContourFilter->GetOutput());
ContourFilter->Delete();
return zerosurface;
```

- **DeepCopy**: creates a complete duplicate

Example: Library - Imaging - .h I

```
//                                                    03-vtkMyLibImaging.h
#ifndef __vtkMyLibImaging_h
#define __vtkMyLibImaging_h

#include "vtkMyLibConfigure.h" // Include configuration header.
#include <vtkObject.h>
#include <vtkImageData.h>
#include <vtkPolyData.h>

class VTK_vtkMyLib_EXPORT vtkMyLibImaging : public vtkObject
{
public:
  static vtkMyLibImaging *New();
  vtkTypeRevisionMacro(vtkMyLibImaging,vtkObject);
  void PrintSelf(ostream& os, vtkIndent indent);

  vtkPolyData*  ExtractContour(vtkImageData *img, double level);
  vtkImageData* SmoothImageAndComputeGradient(vtkImageData* i, double s, int d);

protected:
   vtkMyLibImaging() {};
  ~vtkMyLibImaging() {};
};
#endif
```

Example: Library - Imaging - .cxx I

03-vtkMyLibImaging.cxx

```
//
#include "vtkMyLibImaging.h"
#include <vtkObjectFactory.h>
#include <vtkImageData.h>
#include <vtkDataArray.h>
#include <vtkPointData.h>
#include <vtkPolyData.h>
#include <vtkContourFilter.h>
#include <vtkImageGradient.h>
#include <vtkImageGaussianSmooth.h>
#include <vtkSmartPointer.h>
vtkCxxRevisionMacro(vtkMyLibImaging, "$Revision: 1.2 $");
vtkStandardNewMacro(vtkMyLibImaging);
vtkPolyData* vtkMyLibImaging::ExtractContour(vtkImageData* img, double level)
{
  if (img == NULL) { vtkErrorMacro("Contour: no input\n"); return NULL; }
  vtkSmartPointer<vtkContourFilter>
  contourFilter = vtkSmartPointer<vtkContourFilter>::New();
  contourFilter->SetInput(img);
  contourFilter->SetValue(1, level); contourFilter->Update();

  vtkPolyData* zerosurface = vtkPolyData::New();
  zerosurface->ShallowCopy(contourFilter->GetOutput());
  return zerosurface;
}
```

Example: Library - Imaging - .cxx II

```
vtkImageData* vtkMyLibImaging::SmoothImageAndComputeGradient(vtkImageData* input,
                                                             double sigma,int dimensionality)
{
  if (input == NULL) { vtkErrorMacro("Smooth: no input\n"); return NULL; }
  vtkSmartPointer<vtkImageGaussianSmooth>
  sm = vtkSmartPointer<vtkImageGaussianSmooth>::New();
  sm->SetInput(input);
  sm->SetStandardDeviations(sigma,sigma,sigma);
  sm->SetDimensionality(dimensionality);

  vtkSmartPointer<vtkImageGradient>
  gradient = vtkSmartPointer<vtkImageGradient>::New();
  gradient->SetInput(sm->GetOutput());
  gradient->SetDimensionality(dimensionality); gradient->Update();

  vtkImageData* grad = vtkImageData::New();
  grad->ShallowCopy(gradient->GetOutput());
  return grad;
}
void vtkMyLibImaging::PrintSelf(ostream& os, vtkIndent indent)
{
  this->Superclass::PrintSelf(os,indent); os << indent << "vtkMyLibImaging\n";
}
```

Example: Library - Imaging - .py I

```
#!/usr/bin/python
#                                                    03-vtkMyLibTest.py

# execution:
# PYTHONPATH=$PYTHONPATH:<pathto-libvtkMyLibPython.so> vtkpython vtkMyLibTest.py

import vtk ; from    vtk      import *
import os
if os.name == 'posix': from libvtkMyLibPython import *
else:                  from    vtkMyLibPython import *

reader = vtkStructuredPointsReader();
reader.SetFileName("brain.vt");

util = vtkMyLibImaging() ;  print "Util: ", util

sur = util.ExtractContour( reader.GetOutput(), 50.0 )

writerPoly = vtkXMLPolyDataWriter()
writerPoly.SetInput( sur )
writerPoly.SetFileName("brainContour.vtp")
writerPoly.Write()

img = util.SmoothImageAndComputeGradient( reader.GetOutput(), 2.0, 3 )
```

Example: Library - Imaging - .py II

```
writer3dImg = vtkXMLImageDataWriter()
writer3dImg.SetInput( img )
writer3dImg.SetFileName("smoothedBrain.vti")
writer3dImg.Write()
```

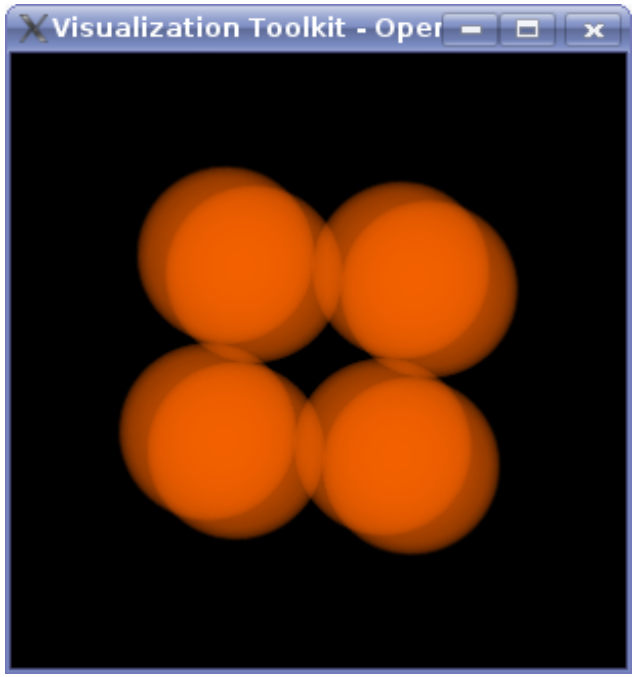
Programmable classes

Programmable classes

Using the *programmable* classes avoids the need for subclassing - and the function can be defined in an interpreter wrapper language:

- **vtkProgrammableSource**: generate source dataset via a user-specified function (any output type is possible).
- **vtkProgrammableFilter**: a user-programmable filter (it is your responsibility to set and get the correct input and output types). DIFFERENCE W AttributeDataFilter?
- **vtkProgrammableAttributeDataFilter**: manipulate attribute (cell and point) data via a user-specified function (i.e. to use a complex formula to generate, to convert vectors to normals, to compute scalar values as a function of vectors, texture coords...). It takes multiple inputs (you can combine several dataset point attributes).
- **vtkProgrammableDataObjectSource**:
- **vtkProgrammableGlyphFilter**:

Example: Prog.Source - screenshot



Example: Programmable source I

```
#!/usr/bin/python
#                                                    50-ProgrammableSource.py
import vtk
from    vtk import *

pointSource = vtkProgrammableSource()                    # Source
def genPoints():
    points = vtkPoints()

    output = pointSource.GetPolyDataOutput()
    output.SetPoints(points)

    # Left the points in a *.txt file, and read them from here
    for i in range(0,2):
      for j in range(0,2):
        for k in range(0,2):
          points.InsertNextPoint( i, j, k )

pointSource.SetExecuteMethod(genPoints)

popSplatter = vtkGaussianSplatter()                      # Splatter:
popSplatter.SetSampleDimensions(300, 300, 300)           # points -> gauss sphere
popSplatter.SetInputConnection( pointSource.GetOutputPort() )
popSplatter.SetRadius(0.5)              # f(x) = scale * exp( expF*((r/Radius)^2) )
popSplatter.SetExponentFactor(1.0)
```

Example: Programmable source II

```
popSplatter.SetScaleFactor(1.0)

cast = vtkImageCast()                                    # uchar conversion
cast.SetInput( popSplatter.GetOutput() )                 #  (required by VRayCast)
cast.SetOutputScalarTypeToUnsignedChar()

alphaTF = vtkPiecewiseFunction()                         # Opacity (A-TF)
alphaTF.AddPoint(0, 0) ; alphaTF.AddPoint(1, 0.01) ; alphaTF.AddPoint(255, 0.5)

colorTF = vtkColorTransferFunction()                     # Color (RGB-TF)
colorTF.AddRGBPoint(  0, 1, 0.4, 0) ;  colorTF.AddRGBPoint(255, 1, 0.0, 0)

volumeProperty = vtkVolumeProperty()                     # Property
volumeProperty.SetColor(colorTF) ; volumeProperty.SetScalarOpacity(alphaTF)
volumeProperty.SetInterpolationTypeToLinear()

compositeFunction = vtkVolumeRayCastCompositeFunction()# Mapper
volumeMapper = vtkVolumeRayCastMapper()                  # (Software)
volumeMapper.SetVolumeRayCastFunction(compositeFunction)
volumeMapper.SetInput( cast.GetOutput() )

volume = vtkVolume()                                     # Volume=Mapper+Property
volume.SetMapper(volumeMapper) ; volume.SetProperty(volumeProperty)

ren = vtkRenderer()                                      # Renderer
ren.AddVolume(volume) ; ren.ResetCamera()
```

Example: Programmable source III

```
ren.GetActiveCamera().ParallelProjectionOn();

renwin = vtkRenderWindow()                                    # Window
renwin.AddRenderer(ren)
renwin.SetSize(300, 300) ; renwin.SetDesiredUpdateRate(1.0)

iren = vtkRenderWindowInteractor()                            # Interactor
iren.SetRenderWindow(renwin)
iren.Start()
```

Part III

Appendixes

Appendixes

VTK Examples

Example locations I

Main locations and recomended examples:

- Included in this slides
- VTK/Examples/
  - `cd VTK/Examples/GUI/Python`
  - `vtkpython VolumeRenderWithBoxWidget.py`
  - `vtkpython ImagePlaneWidget.py`
- KWWidgets/Examples/
  - `cd KWWidgets/Examples/Python/WidgetsTour/`
  - `vtkpython KWWidgetsTourExample.py`

Executing environment

- **Python**:
  - **Testing**: use `vtkpython <script.py>`
  - **Generic**:

```
export LD_LIBRARY_PATH=<directory that contains the libvtk*.so>
export PYTHONPATH=$VTK_ROOT/Wrapping/Python:${LIBRARY_OUTPUT_PATH}
python <script.py>
  containing "from vtk import *"
```

  - **Requirements**: `aptitude install python-tk`
- **TCL**:

```
export TCLLIBPATH=/opt/local/VTK/Wrapping/Tcl
wish <script.tcl>
  containing "package require vtk"
```

vtkRenderWindowInteractor: mouse

Mouse bindings

- camera
  - **Button 1**: rotate
  - **Button 2**: pan
  - **Button 3**: zoom
  - **Ctrl-Button 1**: spin
- actor
  - **Button 1**: rotate
  - **Button 2**: pan
  - **Button 3**: uniform scale
  - **Ctrl-Button 1**: spin
  - **Ctrl-Button 2**: dolly.

vtkRenderWindowInteractor: keys

Keyboard bindings (upper or lower case)

- **j**(*joystick*) like mouse interactions
- **t**(*trackball*) like mouse interactions
- **o**(*object*)/actor interaction
- **c**(*camera*) interaction
- **r**(*reset*) camera view
- **w**(*wireframe*) all actors
- **s**(*surface*) all actors
- **u**(*user*) defined function execution
- **p**(*pick*) under mouse pointer
- **f**(*focus*) point to rotate around
- **3**(*stereo mode*) switch
- **e**(*exit*), **q**(*quit*)

Examples overview I

General

- Read VTK/Examples/README.txt

```
for f in *.py; do { echo $f; head $f; vtkpython $f; } done
```

VTK/Examples/**GUI**/Python/:

- **VolumeRenderWithBoxWidget**: head in VR
- **ImagePlaneWidget**: head with one 2D slice per axis
- **CustomInteraction**: define response to events
- 3D Widgets: press 'i' to show widget
  - **TransformWithBoxWidget**: expand/contract volume
  - **BoxWidget**: select a sub region with a box
  - **ImplicitPlaneWidget**: select a plane
  - **ProbeWithPointWidget**: point probe
  - **ProbingWithPlaneWidget**: plane probe

Examples overview II

- **SphereWidget**: to drag the light point
- **StreamlinesWithLineWidget**: to drag stream lines

Examples overview III

VTK/Examples/**VolumeRendering**/Python/:

- **PseudoVolumeRendering**: compositing translucent cut planes.
- **SimpleRayCast**: vtkVolumeRayCast mapper
- **SimpleTextureMap2D**: vtkVolumeTextureMapper2D mapper

Examples overview IV

VTK/Examples/**Rendering**/Python/:

- **rainbow**: lookup tables
- **TPlane**: basic texture mapping

Examples overview V

VTK/Examples/**VisualizationAlgorithms**/Python/:

- **ExtractGeometry**: extract a piece of a dataset using an implicit function (a boolean combination of two ellipsoids).
- **spikeF**: use of glyphs.
- **officeTube**: single streamline to streamtube.
- **streamSurface**: stream surfaces.
- **VisQuad**: volume of data samples from an implicit function.
- **DepthSort**: poor man's algorithm to sort polygons for proper transparent blending.
- **ExtractUGrid**: vtkConnectivityFilter is also used to extract connected components.
- **SubsampleGrid**: subsampling of a structured grid.

Examples overview VI

VTK/Examples/**Build**/

- **vtkLocal**: template to create a local project that adds classes to VTK and create wrappers.
- **vtkMy**: same that `vtkLocal` but with separate directory per module.

Examples overview VII

VTK/**bin**/

- **./VolumeRenderingCxxTests** `TestHAVSVolumeMapper -D /opt/local/fonts/VTKData/`
- **./VTKBenchMark**

An idea to build Volume Splatting

And idea to build Volume Splatting I

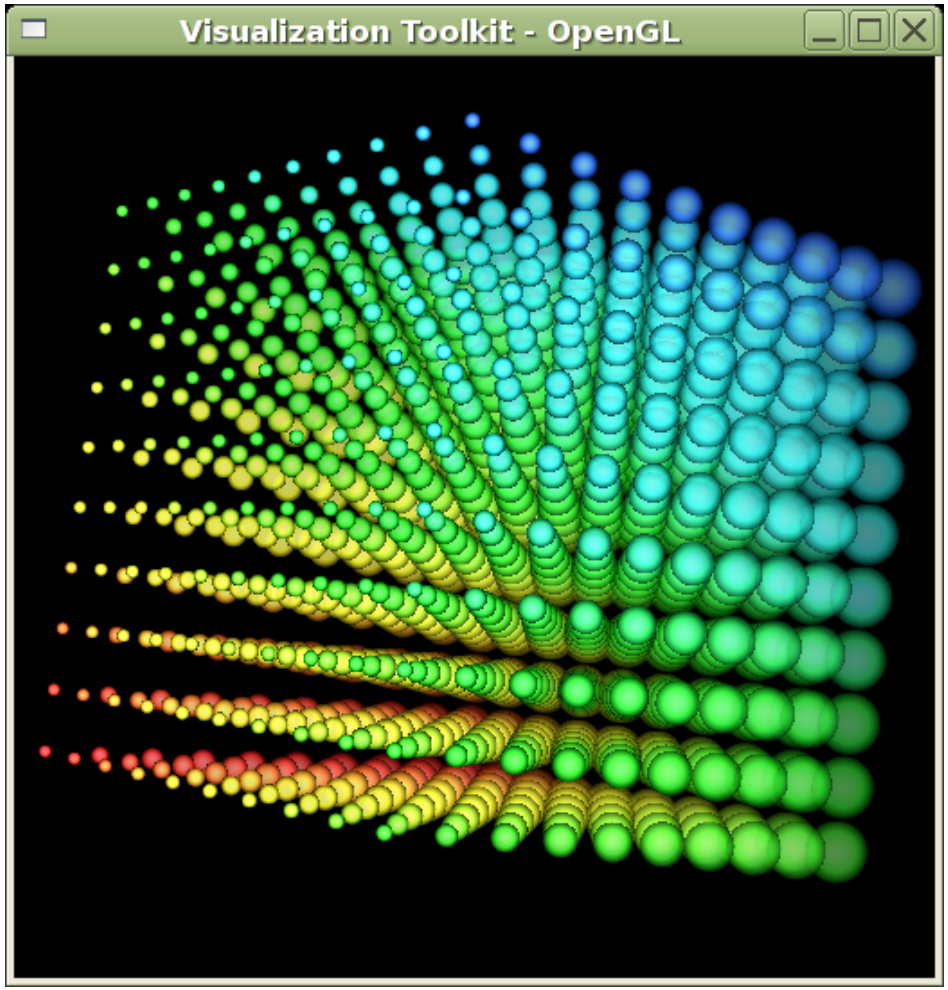
Idea: adapt the vtkPointSprites developed by CSCS `http://www.cscs.ch` to splat "sets of voxels" or ImageData.

(see Appendixes / Point Sprites installation)

WORK IN PROGRESS

- vtkPointSpritesMapper
  - Cleaning: John Biddiscombe is doing it.
  - Python wrapping: done
- Splatting sets of voxels
  - use ColorLUT: in progress
- Splatting imageData: to solve after the previous issue.
- Creating a Paraview plugin

Example: Using vtkPointSprites - Screenshot



Example: Using vtkPointSprites - from PolyData I

```
#!/usr/bin/python
#                                                    201-PointsToSprites.py
# execution:
# PYTHONPATH=$PYTHONPATH:<pathto-libvtkCSCSPointSpritesPython.so> python <script> <resou
# PYTHONPATH=$PYTHONPATH:vtkPointSprites/bin python 201-PointsToSprites.py vtkPointSprit

import vtk, os, sys
from    vtk      import *
if os.name == 'posix': from libvtkCSCSPointSpritesPython import *
else:                  from    vtkCSCSPointSpritesPython import *

SHADERS    = True  # alpha-size(shaders=T, sprites=F)
SPRITES    = False # alpha(shaders=F, sprites=T) squares(shaders=sprites=F)
ROWS       = 10
NUMVOXELS = ROWS*ROWS*ROWS

try: resources = sys.argv[1]                         # Resources dir
except Exception: sys.exit("\n<resources-dir> argument required\n");
os.environ["USER_MATERIALS_DIRS"]=resources

points = vtkPoints()                                 # Coordinates
points.SetNumberOfPoints(NUMVOXELS)

sizes = vtkFloatArray()                              # Sizes
sizes.SetName("PointSizes")
```

Example: Using vtkPointSprites - from PolyData II

```
sizes.SetNumberOfTuples(NUMVOXELS)
sizes.SetNumberOfComponents(1)

alpha = vtkFloatArray()                                    # Alphas
alpha.SetName("Alpha")
alpha.SetNumberOfTuples(NUMVOXELS)
alpha.SetNumberOfComponents(1)

for z in range(0, ROWS):                                   # Fill arrays
  for y in range(0, ROWS):
    for x in range(0, ROWS):
      id = z*ROWS*ROWS + y*ROWS + x    ; points.SetPoint(id, x, y, z)
      sizes.SetValue(id, .1 + .04 * z);  alpha.SetValue(id, 1. - .05 * z)

voxels = vtkPolyData()                                     # Voxels =
voxels.SetPoints(points);                                  #  coordinates
voxels.GetPointData().AddArray(sizes);                     #  + sizes
voxels.GetPointData().AddArray(alpha);                     #  + alphas

sColor = vtkElevationFilter()                              # ColorLUT
sColor.SetInput(voxels)
sColor.SetLowPoint ( 0.0, 0.0, 0.0)
sColor.SetHighPoint( ROWS, ROWS, 0.0)

mapper = vtkPointSpriteMapper()                            # Mapper
mapper.SetInputConnection(sColor.GetOutputPort())
```

Example: Using vtkPointSprites - from PolyData III

```
mapper.SetResourceDirectory(resources)
mapper.SetImmediateModeRendering(1)
mapper.SetRadiusChannelArray("PointSizes")
mapper.SetAlphaChannelArray("Alpha")
mapper.SelectColorArray("Elevation")
mapper.SetEnableDepthSort(1)
if    SHADERS:
  mapper.SetRenderModeToShadingLanguage()
  mapper.SetDefaultRadius(0.0001)
elif SPRITES:
  image = resources + "/sphere_64x64.png"
  mapper.SetParticleImageFileName(image)
  mapper.SetParticleImage(None)
  mapper.SetRenderModeToPointSprite()
  mapper.SetBlendModeToOcclude()
  mapper.SetDefaultPointSize(64)
  mapper.SetQuadraticPointDistanceAttenuation(1.0, 0.1, 0.0)

actor = vtkActor()                                    # Actor
actor.SetMapper(mapper)

ren = vtkRenderer()                                   # Renderer
ren.AddActor(actor)

renWin = vtkRenderWindow()                            # Window
renWin.SetSize(512, 512)
```

Example: Using vtkPointSprites - from PolyData IV

```
renWin.AddRenderer(ren)

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

vtkPointSprites on VTK I

CSCS: http://www.cscs.ch/a-display.php?id=1168

```
svn checkout https://svn.cscs.ch/vtkContrib/trunk/vtkCSCS/vtkPointSprites
cd vtkPointSprites

vi vtkPointSpriteMapper.cpp        # Comment glu stuff
  //  gluLookAt(
  //    eye[0],  eye[1],  eye[2],
  //    view[0], view[1], view[2],
  //    this->SliceUp[0], this->SliceUp[1], this->SliceUp[2]
  //  );
vi glErrorUtil.cpp                 # Comment glu stuff
  //    errstr = reinterpret_cast<const char *>(gluErrorString(errnum));
  errstr = NULL;

ccmake .
  CMAKE_BUILD_TYPE                 Debug
  CSCS_PARAVIEW_MODULES            OFF    (ON=PV, OFF=VTK)
  USE_VTK_OUTPUT_PATHS             ON

make

<path-vtk-bin>/vtkCSCSPointSpritesCxxTests -R TestPointSpriteTransparency -res <path-sprites>/bin/resource
```

Python TVTK

TVTK

(extracted from `https://svn.enthought.com/enthought/wiki/TVTK`)

- tvtk is **free software** with a BSD style license (from Enthought framework, related to scipy!)
- All VTK classes are wrapped in a **Pythonic feel**.
- Classes are **generated at install time** on the installed platform.
- Support for **traits** (default value, strongly typed, **GUI**...)
- Elementary **pickle support** (object serialization)
- Handles **Numeric/numarray/scipy arrays** & **Python lists** transparently.
- Support for a **pipeline browser**, ivtk and a high-level mlab like module.
- **Envisage plugins** for a tvtk scene and the pipeline browser.

Example: basic Cone in TVTK I

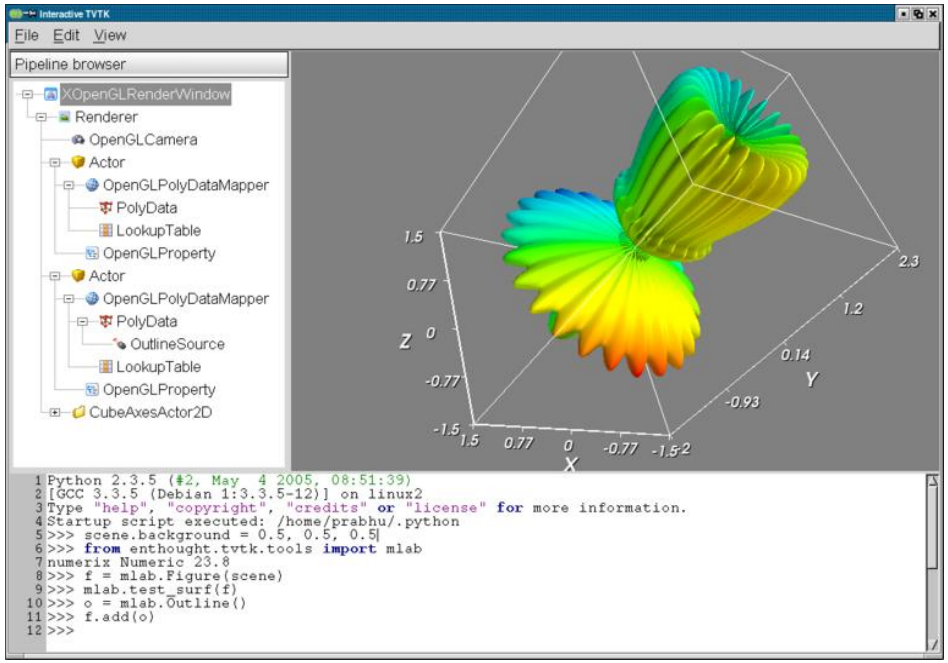
TODO: put this in a file, test, and insert here with syntax highlighting

```
from enthought.tvtk.api import tvtk
source = tvtk.ConeSource(resolution=36)
mapper = tvtk.PolyDataMapper(input=source.output)
prop   = tvtk.Property(representation='w')
actor  = tvtk.Actor(mapper=mapper, property=prop)
render = tvtk.Renderer()
render.add_actor(actor)
window = tvtk.RenderWindow(size=(400,400))
window.add_renderer(renderer)
inter  = tvtk.RenderWindowInteractor(render_window=window)
```

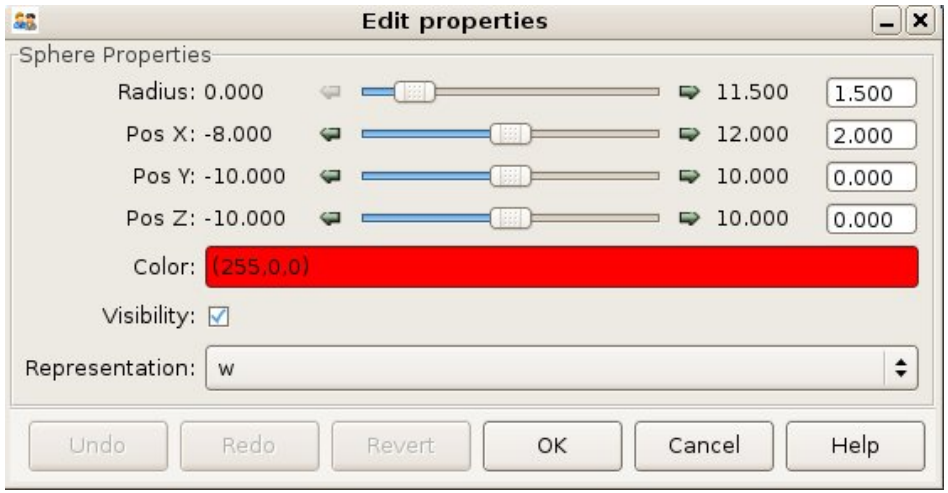
Example: basic Cone in TVTK II

```
inter.initialize()
inter.start()
```

Example: itvk - pipeline browser



Example: TVTK - Traits - GUI



References

References

VTK

- **VTK User's Guide**: `http://www.kitware.com/products/vtkguide.html`
- **VTKbook**: `http://www.bioimagesuite.org/vtkbook/`
- **KW Newsletter**: `http://www.kitware.com/products/newsletter.html`
- **Original paper**: `http://vtk.org/pdf/dioot.pdf`
- **Python TVTK**: `https://svn.enthought.com/enthought/wiki/TVTK`