Here are some more examples of simple ParaView 3 python filters.

From KitwarePublic

Contents

Examples of Filters Programmed using the Python Programmable Filter

It would be nice, if you have written a possibly useful pp-filter, if you would add the code to this page. Here are some simple examples.

Note: These scripts are meant to be used at values for the **Script** paramaters for **Programmable Filter** or **Programmable Source** as identified by the title. These will not directly work from the python shell or pvpython.

CSV Reader (Source)

```
# This source reads a text file and produces a data set
# The file is of the form (where [denotes optional and repeatable]:
# X,Y,Z[,ARRAYNAME]
# x0,y0,z0[,value]
# x1,y1,z1[,value]
# ...
from paraview import vtk
import os
pts = vtk.vtkPoints()
firstline = True
filename = os.path.normcase("c:/datafile.txt")
f = open(filename)
pdo = self.GetOutput()
for line in f:
  if firstline:
    #skip first line, presumed to be a header
    firstline = False
    for pos,word in enumerate(line.split(",")):
       if pos > 2:
         newArray = vtk.vtkDoubleArray()
         newArray.SetName(word)
         newArray.SetNumberOfComponents(1)
         pdo.GetPointData().AddArray(newArray)
  else:
    for pos,word in enumerate(line.split(",")):
       print word
       if pos == 0:
         x = float(word)
       if pos == 1:
         y = float(word)
       if pos == 2:
         z = float(word)
       if pos > 2:
         array = pdo.GetPointData().GetArray(pos-3)
         array.InsertNextValue(float(word))
    pts.InsertNextPoint(x,y,z)
pdo.SetPoints(pts)
```

Tetrahedra Volume (Filter)

```
# This filter computes the volume of the tetrahedra in an unstructured mesh:
pdi = self.GetInput()
pdo = self.GetOutput()
newData = vtk.vtkDoubleArray()
newData.SetName("Volume")
numTets = pdi.GetNumberOfCells()
for i in range(numTets):
       cell = pdi.GetCell(i)
       p1 = pdi.GetPoint(cell.GetPointId(0))
       p2 = pdi.GetPoint(cell.GetPointId(1))
       p3 = pdi.GetPoint(cell.GetPointId(2))
       p4 = pdi.GetPoint(cell.GetPointId(3))
       volume = vtk.vtkTetra.ComputeVolume(p1,p2,p3,p4)
       newData.InsertNextValue(volume)
pdo.GetCellData().AddArray(newData)
```

Tetrahedra Radius (Filter)

```
# This filter computes the radius h of the tetrahedra in an unstructured mesh:
# Adapted by Johan Jansson (jjan@csc.kth.se)
from math import *
pdi = self.GetInput()
pdo = self.GetOutput()
newData = vtk.vtkDoubleArray()
newData.SetName("h")
numTets = pdi.GetNumberOfCells()
for i in range(numTets):
    cell = pdi.GetCell(i)
    p1 = pdi.GetPoint(cell.GetPointId(0))
    p2 = pdi.GetPoint(cell.GetPointId(1))
    p3 = pdi.GetPoint(cell.GetPointId(2))
    p4 = pdi.GetPoint(cell.GetPointId(3))
    c = [0.0, 0.0, 0.0]
    h = vtk.vtkTetra.Circumsphere(p1,p2,p3,p4,c)
    # VTK actually computes the square
    h = sqrt(h)
    newData.InsertNextValue(h)
pdo.GetCellData().AddArray(newData)
```

Flip Tetrahedra (Filter)

```
# This filter flips the tetrahedra (useful, if you have different convention of
# tet orientation than VTK, and wish to use the vtkMeshQuality filter).
pdi = self.GetInput()
pdo = self.GetOutput()
numTets = pdi.GetNumberOfCells()
newcells = vtk.vtkCellArray()
for i in range(numTets):
       cell = pdi.GetCell(i)
       i1 = cell.GetPointId(0)
       i2 = cell.GetPointId(1)
       i3 = cell.GetPointId(2)
       i4 = cell.GetPointId(3)
       newcells.InsertNextCell(4)
       newcells.InsertCellPoint(i1)
       newcells.InsertCellPoint(i2)
       newcells.InsertCellPoint(i4)
       newcells.InsertCellPoint(i3)
pdo.SetCells( 10, newcells )
```

Helix (Source)

This is intended as the Python script for a 'Programmable Source'. It generates a helix with two sets of scalar data defined along the helix. The scalar data can be visualized using filters such as the 'tube' filter where the scalar value is represented as a color. The scalar called 'First Property' or the scalar called 'Second Property' can be chosen from the 'Display' tab of the 'tube' filter under the 'Color by' option. This lists the available scalars that are defined at points along the helix.

```
#This script generates a helix curve.
#This is intended as the script of a 'Programmable Source'
import math

numPts = 80.0 # Points along Helix
```

```
numPts = 80.0 # Points along Helix
length = 8.0 # Length of Helix
rounds = 3.0 # Number of times around

#Get a vtk.PolyData object for the output
pdo = self.GetPolyDataOutput()

#This will store the points for the Helix
newPts = vtk.vtkPoints()

#Associate scalar values with the points.
#Also set the name for the scalar that will be
#visible in Paraview dropdown lists for setting
#color along the helix.
vals1 = vtk.vtkDoubleArray()
vals1.SetName('First Property')
vals2 = vtk.vtkDoubleArray()
vals2.SetName('Second Property')

for i in range(numPts):
   #Generate the Points along the Helix
   x = i*length/numPts
   y = math.sin(i*rounds*2*math.pi/numPts)
   z = math.cos(i*rounds*2*math.pi/numPts)
   #Insert the Points into the vtkPoints object
   #The first parameter indicates the reference.
   #value for the point. Here we add the sequentially.
   #Note that the first point is at index 0 (not 1).
   newPts.InsertPoint(i, x,y,z)

   #Order of Scalars should match order Pts were added
   vals1.InsertNextValue(i)
   vals2.InsertNextValue(math.sin(i*2.0*math.pi/numPts))

#Add the points to the vtkPolyData object
#Right now the points are not associated with a line -
#it is just a set of unconnected points. We need to
#create a 'cell' object that ties points together
#to make, in this case, a curve. This is done below.
#A 'cell' is just an object that tell how points are
#connected to make a 1D, 2D, or 3D object.
pdo.SetPoints(newPts)

# Add the scalar point data
pdo.GetPointData().SetScalars(vals1) #'First Property' scalar
pdo.GetPointData().AddArray(vals2) #'Second Property' scalar

#Make a vtkPolyLine which holds the info necessary
#to create a curve composed of line segments. This
#really just hold constructor data that will be passed
#to vtkPolyData to add a new line.
aPolyLine = vtk.vtkPolyLine()

#Indicate the number of points along the line
aPolyLine.GetPointIds().SetNumberOfIds(numPts)
for i in range(numPts):
   #Add the points to the line. The first value indicates
   #the order of the point on the line. The second value
   #is a reference to a point in a vtkPoints object. Depends
   #on the order that Points were added to vtkPoints object.
   #Note that this will not be associated with actual points
   #until it is added to a vtkPolyData object which holds a
```

```
    #vtkPoints object.
    aPolyLine.GetPointIds().SetId(i, i)

#Allocate the number of 'cells' that will be added. We are just
#adding one vtkPolyLine 'cell' to the vtkPolyData object.
pdo.Allocate(1, 1)

#Add the poly line 'cell' to the vtkPolyData object.
pdo.InsertNextCell(aPolyLine.GetCellType(), aPolyLine.GetPointIds())

#The Helix is ready to plot! Click 'Apply'.
```

Producing Data with Timesteps (Source)

This example demonstrates how to write a source which produces multiple timesteps. There's one caveat. Currently ParaView client does not notice the timesteps produced by the source and hence the GUI will not update the animation time ranges automatically, as is the case for standard reader.

Script(RequestInformation)

```
def SetOutputTimesteps(algorithm, timesteps):
    executive = algorithm.GetExecutive()
    outInfo = executive.GetOutputInformation(0)
    outInfo.Remove(executive.TIME_STEPS())
    for timestep in timesteps:
      outInfo.Append(executive.TIME_STEPS(), timestep)
    outInfo.Remove(executive.TIME_RANGE())
    outInfo.Append(executive.TIME_RANGE(), timesteps[0])
    outInfo.Append(executive.TIME_RANGE(), timesteps[-1])
SetOutputTimesteps(self, (0, 10, 20, 30))
```

Script (RequestData)

```
def GetUpdateTimesteps(algorithm):
    executive = algorithm.GetExecutive()
    outInfo = executive.GetOutputInformation(0)
    if not outInfo.Has(executive.UPDATE_TIME_STEPS()):
        return []
    count = outInfo.Length(executive.UPDATE_TIME_STEPS())
    timesteps = [outInfo.Get(executive.UPDATE_TIME_STEPS(), cc)
                 for cc in range(count)]
    return timesteps

# This is the requested time-step. This may not be exactly equal to the
# timesteps published in RequestInformation(). Your code must handle that
# correctly
req_timesteps = GetUpdateTimesteps(self)

output = self.GetOutput()

# TODO: Generate the data as you want.

# Now mark the timestep produced.
output.GetInformation().Remove(output.DATA_TIME_STEPS())
output.GetInformation().Append(output.DATA_TIME_STEPS(), req_timesteps[0])
```

Producing Image Data (Source)

This example demonstrates how to produce image data using the python programmable source. After creating the programmable source, you should set the *Output Data Set Type* to **vtkImageData**. In RequestInformation, the whole extent, spacing, and scalar type is set. In RequestData, image data is produced for the region defined by the UPDATE_EXTENT.

Script(RequestInformation)

```
executive = self.GetExecutive()
outInfo = executive.GetOutputInformation(0)
executive.SetExtentTranslator(outInfo, vtk.vtkExtentTranslator())
outInfo.Set(executive.WHOLE_EXTENT(), 0, 9, 0, 9, 0, 9)
outInfo.Set(vtk.vtkDataObject.SPACING(), 1, 1, 1)
dataType = 10 # VTK_FLOAT
numberOfComponents = 1
vtk.vtkDataObject.SetPointDataActiveScalarInfo(outInfo, dataType, numberOfComponents)
```

Script (RequestData)

```
executive = self.GetExecutive()
outInfo = executive.GetOutputInformation(0)
updateExtent = [executive.UPDATE_EXTENT().Get(outInfo, i) for i in xrange(6)]

imageData = self.GetOutput()
imageData.SetExtent(updateExtent)
imageData.AllocateScalars()

pointData = imageData.GetPointData().GetScalars()
pointData.SetName("my data")

dimensions = imageData.GetDimensions()

for i in xrange(dimensions[0]):
  for j in xrange(dimensions[1]):
    for k in xrange(dimensions[2]):
      pointId = vtk.vtkStructuredData.ComputePointId(dimensions, (i, j, k))
      pointData.SetValue(pointId, pointId)
```

Producing An Unstructured Grid (Source)

This example demonstrates how to produce an unstructured grid using the python programmable source. After creating the programmable source, you should set the *Output Data Set Type* to **vtkUnstructuredGrid**. In RequestData, an unstructured grid is created with both point and cell data. An example input file is File:HexMesh.txt.

Script (RequestData)

```
from paraview import vtk
import os
filename = os.path.normcase("HexMesh.txt")
f = open(filename)
output = self.GetOutput()

numPoints = int(f.readline())
pts = vtk.vtkPoints()
for point in range(numPoints):
  x = float()
  y = float()
  z = float()
  for pos,word in enumerate(f.readline().split(",")):
    if pos == 0:
      x = float(word)
    if pos == 1:
      y = float(word)
    if pos == 2:
      z = float(word)
  pts.InsertNextPoint(x,y,z)

output.SetPoints(pts)

numCells = int(f.readline())
output.Allocate(numCells, 1000)
for cell in range(numCells):
  data = f.readline().split(",")
  numCellPoints = int(data[0])
  pointIds = vtk.vtkIdList()
  for pointId in range(numCellPoints):
    pointIds.InsertId(pointId, int(data[pointId+1]))
  # cell type number is listed in vtkCellType.h
  # 12 is for hexahedron
  output.InsertNextCell(12, pointIds)

# load in point data
def addfielddata(numTuples, dataSetAttributes):
  numArrays = int(f.readline())
  for array in range(numArrays):
    data = f.readline().split(",")
    numberOfComponents = int(data[0])
    dataArray = vtk.vtkDoubleArray()
    dataSetAttributes.AddArray(dataArray)
    dataArray.SetNumberOfComponents(numberOfComponents)
    dataArray.SetName(data[1].strip())
    for t in range(numTuples):
      data = f.readline().split(",")
      for value in data:
        dataArray.InsertNextValue(float(value))

addfielddata(numPoints, output.GetPointData())
addfielddata(numCells, output.GetCellData())
```

Labeling common points between two datasets (Filter)

In this example, the programmable filter to takes two input datasets, A and B, and outputs dataset B with a new scalar array that labels the points in B that are also in A. You should select two datasets in the pipeline browser and then apply the programmable filter.

Script (RequestData)

```
# Get the two inputs
A = self.GetInputDataObject(0, 0)
B = self.GetInputDataObject(0, 1)

# Construct a point locator containing A's points
locator = vtk.vtkPointLocator()
locator.InitPointInsertion(vtk.vtkPoints(), A.GetBounds())
for i in xrange(A.GetNumberOfPoints()):
    locator.InsertNextPoint(A.GetPoints().GetPoint(i))

# Create an array to store the labels
labels = vtk.vtkUnsignedCharArray()
labels.SetName("common points")
labels.SetNumberOfTuples(B.GetNumberOfPoints())

# Label the points in B that are also in A
for i in xrange(B.GetNumberOfPoints()):
    point = B.GetPoints().GetPoint(i)
    if locator.IsInsertedPoint(point) != -1:
        labels.SetValue(i, 1)
    else:
        labels.SetValue(i, 0)

# Initialize the output and add the labels array
output = self.GetOutput()
output.ShallowCopy(B)
output.GetPointData().AddArray(labels)
```
