Formation VTK 2007

Listing et Annexe

Contact:
Artenum SARL
Technopole paris Cyber Village
101-103 boulevard Mac Donald
75019 PARIS, France

tel : (+00 33) 01 44 89 45 15
fax: (+00 33) 01 44 89 45 17

E-mail : contact@artenum.com
Site : http://www.artenum.com/services.php

Sommaire

Scripts Python

Gestion de données

Création d'une ImageData

```
# import des different composants necessaires
from vtk import *

dataset = vtkImageData()
dataset.SetDimensions(26, 26, 26)
dataset.SetOrigin(-0.5, -0.5, -0.5)
gridStep = 1.0/25.0
dataset.SetSpacing(gridStep, gridStep, gridStep)

# creation des donnees, i.e un tableau de flottants
scalarData = vtkFloatArray()

for k in range(26):
    z = -0.5 + k*gridStep
    kOffset = k * 26 *26
    for j in range(26):
        y = -0.5 + j*gridStep
        jOffset = j * 26
        for i in range(26):
            x = -0.5 + i*gridStep
            value = x*x + y*y + z*z - (0.4*0.4)
            offset = i + jOffset + kOffset
            # ajout de la donnee au vtkFloatArray
            scalarData.InsertTuple1(offset, value)

dataset.GetPointData().SetScalars(scalarData)
```

Création d'une StructuredGrid

```
# import des different composants necessaires
import vtk
from vtk import *
import math
from math import *

x = [0,0,0]
rMin = 0.5
rMax = 1.0
dims = [13, 11, 11]

# Create the structured grid.
dataset = vtkStructuredGrid()
dataset.SetDimensions(dims);

# We also create the points and datas. The points
# form a hemi-cylinder of data.

scalarData = vtkFloatArray()
points = vtkPoints()

deltaZ = 2.0 / (dims[2]-1)
deltaRad = (rMax-rMin) / (dims[1]-1)
for k in range(dims[2]):
    x[2] = -1.0 + k*deltaZ
    kOffset = k * dims[0] * dims[1]
    for j in range(dims[1]):
      radius = rMin + j*deltaRad
      jOffset = j * dims[0]
      for i in range(dims[0]):
        theta = i * 15.0 * math.pi/180.0  #vtkMath.DegreesToRadians()
        x[0] = radius * cos(theta)
        x[1] = radius * sin(theta)
        offset = i + jOffset + kOffset
        points.InsertPoint(offset,x)
        valeur = offset
        scalarData.InsertTuple1(offset,valeur)

dataset.SetPoints(points)
dataset.GetPointData().SetScalars(scalarData)
```

Création d'une UnstructuredGrid

```
# import des different composants necessaires
from vtk import *

# ------------------------------------------------
# creation de structure de donnee
# ------------------------------------------------

nbpoints = 8

# Definition des noeuds
pointCoord   = range(nbpoints)
pointCoord[0]=(0.0, 0.0, 0.0)
pointCoord[1]=(0.3, 0.1, 1.0)
pointCoord[2]=(0.7, 0.1, -0.2)
pointCoord[3]=(0.5, 1.2, 1.1)
pointCoord[4]=(0.9, 2.0, -0.3)
pointCoord[5]=(1.5, 2.5, 1.4)
pointCoord[6]=(0.2, 3.4, -1.2)
pointCoord[7]=(2, 4, -1.2)

# Definition des valeures correspondantes
ptData    = range(nbpoints)
ptData[0] = 1.0
ptData[1] = 1.2
ptData[2] = 1.6
ptData[3] = 2.0
ptData[4] = 2.5
ptData[5] = 3.4
ptData[6] = 5.7
ptData[7] = 7.0


# Constantes definissant le type de cellule
# NB: Deja definies dans VTK, redefinies ici que pour
# la lisibite de l'exemple
VTK_TETRA = 10
VTK_TRIANGLE = 5
VTK_LINE = 3
VTK_VERTEX = 1

dataset =  vtkUnstructuredGrid()
points = vtkPoints()

for i in range(nbpoints):
    points.InsertNextPoint(pointCoord[i])
```

```
data = vtkFloatArray()
data.SetName("My data name")
data.SetNumberOfTuples(1)
data.SetNumberOfValues(nbpoints)

# 1er tetrahedre
idList = vtkIdList()
idList.InsertNextId(0)
idList.InsertNextId(1)
idList.InsertNextId(2)
idList.InsertNextId(3)
dataset.InsertNextCell( VTK_TETRA, idList) #VTK_TETRA = 10

# 2eme tetrahedre
idList = vtkIdList()
idList.InsertNextId(0)
idList.InsertNextId(2)
idList.InsertNextId(3)
idList.InsertNextId(4)
dataset.InsertNextCell( VTK_TETRA, idList) #VTK_TETRA = 10

# le triangle
idList = vtkIdList()
idList.InsertNextId(3)
idList.InsertNextId(4)
idList.InsertNextId(5)
dataset.InsertNextCell( VTK_TRIANGLE, idList) #VTK_TRIANGLE = 5

# l'arrete
idList = vtkIdList()
idList.InsertNextId(5)
idList.InsertNextId(6)
dataset.InsertNextCell( VTK_LINE, idList) #VTK_LINE = 3

# le dernier noeud en tant que simple vertex
idList = vtkIdList()
idList.InsertNextId(7)
dataset.InsertNextCell( VTK_VERTEX, idList) #VTK_VERTEX = 1

# attribution des valeurs aux noeuds
for i in range(nbpoints):
        data.InsertTuple1(i, ptData[i])


dataset.SetPoints(points)
dataset.GetPointData().SetScalars(data)
```

Gestion de la lecture / écriture de fichiers

```
from vtk import *

def readVTK(fileName):
   reader = vtkDataSetReader()
   reader.SetFileName(fileName)
   reader.Update()
   reader.CloseVTKFile()
   return reader.GetOutput()

def readUnstructuredXML(fileName):
   reader = vtkXMLUnstructuredGridReader()
   reader.SetFileName(fileName)
   reader.Update()
   return reader.GetOutput()

def convertLegacy(legacy,xml):
    writer = vtkXMLDataSetWriter()
    writer.SetFileName(xml)
    writer.SetInput( readeVTK(legacy) )
    writer.Update();

def writeASCII(dataset, fileName):
    writer = vtkDataSetWriter()
    writer.SetFileName(fileName)
    writer.SetFileTypeToASCII()
    writer.SetInput(dataset)
    writer.Update()
```

Gestion du pipeline

Réalisation d'un IsoContour

```
from vtk import *

nbContours = 4
shift = 0.5
iso = vtkContourFilter()
iso.SetInput(dataset)
iso.SetNumberOfContours(nbContours)
dataRange = dataset.GetScalarRange()

for i in range(4):
    level = (i+1)*shift*(dataRange[1]-dataRange[0])/nbContours+dataRange[0]
    iso.SetValue(i, level)

dataset = iso.GetOutput()
```

Réalisation d'un plan de coupe

```
# definition de la fonction de coupe
plane = vtkPlane()
plane.SetOrigin(0.0, 0.0, 0.0)
plane.SetNormal(0, 0, 1.0)

#creation du filtre
planeCut = vtkCutter()
planeCut.SetCutFunction(plane)
planeCut.SetInput(dataset)

dataset = planeCut.GetOutput()
```

Réalisation d'une déformation

```
warp = vtkWarpScalar()
warp.SetInput(dataset)
warp.SetNormal(0.0, 0.0, 1.0)
warp.SetScaleFactor(1.0)

dataset = warp.GetOutput()
```

Visualisation d'une donnée dans Cassandra

```
from vtk import *

nom_du_fichier = "test.vtk"

# Lecture du dataset
reader = vtkDataSetReader()
reader.SetFileName(nom_du_fichier)
reader.Update()
reader.CloseVTKFile()
dataset = reader.GetOutput()

# creation du pipeline de visualisation par defaut
mapper = vtkDataSetMapper()
mapper.SetInput(dataset)

actor = vtkActor()
actor.SetMapper(mapper)

# Integration de l'acteur dans la vue 3D
cassandra.getPipeLineManager().addDataSet( dataset,"Data")
cassandra.getPipeLineManager().addMapper(mapper, "")
actorCassandra = cassandra.getPipeLineManager().addActor(actor, "DataView")
cassandra.getPipeLineManager().setActorVisible(actorCassandra,1)
cassandra.getPipeLineManager().validateViewAndWait()
```

Visualisation d'une donnée dans la fenêtre standard VTK

```
from vtk import *

nom_du_fichier = "test.vtk"

# Lecture du dataset
reader = vtkDataSetReader()
reader.SetFileName(nom_du_fichier)
reader.Update()
reader.CloseVTKFile()
dataset = reader.GetOutput()

# creation du pipeline de visualisation par defaut
mapper = vtkDataSetMapper()
mapper.SetInput(dataset)

actor = vtkActor()
actor.SetMapper(mapper)

# Integration de l'acteur dans la vue 3D
renderer = vtkRenderer()
renWin = vtkRenderWindow()
renWin.AddRenderer(renderer)

iren = vtkRenderWindowInteractor()
iren.SetRenderWindow(renWin)

renderer.AddActor(actor)
renderer.SetBackground(1,1,1)
renderer.ResetCamera()

renderer.GetActiveCamera().Elevation(60.0)
renderer.GetActiveCamera().Azimuth(30.0)
renderer.GetActiveCamera().Zoom(1.25)
renWin.SetSize(300,300)

# interact with data
renWin.Render()
iren.Start()
```

Gestion de la scène 3D

Ajout d'une LookupTable

```
from vtk import *

nom_du_fichier = "test.vtk"

# Lecture du dataset
reader = vtkDataSetReader()
reader.SetFileName(nom_du_fichier)
reader.Update()
reader.CloseVTKFile()
dataset = reader.GetOutput()

# creation du pipeline de visualisation par defaut
mapper = vtkDataSetMapper()
mapper.SetInput(dataset)

actor = vtkActor()
actor.SetMapper(mapper)

# Creation d'une lookuptable et connection au mapper
lookupTable = vtkLookupTable()
lookupTable.SetHueRange(0.66667, 0)
#lookupTable.SetScalarRange(dataset.GetScalarRange())
lookupTable.Build()

# connection de la lookuptable avec le mapper
mapper.SetLookupTable(lookupTable)
mapper.SetScalarRange(dataset.GetScalarRange())

# Definition de la bar de couleur
scalBar = vtkScalarBarActor()
scalBar.SetLookupTable(lookupTable)
```

Manipulations d'un acteur 3D

```
from vtk import *

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

mapper = vtkDataSetMapper()
mapper.SetInput(cone.GetOutput())

actor = vtkActor()
actor.SetMapper(mapper)
actor.GetProperty().SetColor(1.0, 0.0, 0.3)

actor.GetProperty().SetRepresentationToWireframe()
actor.GetProperty().SetRepresentationToPoints()
actor.GetProperty().SetPointSize(10.0)

# gestion des interpolations (si donnees mappees)
# actor.GetProperty().SetInterpolationToFlat()
# actor.GetProperty().SetInterpolationToGouraud()
# actor.GetProperty().SetRepresentationToWireframe()
```

Application d'une texture sur un plan

```
from vtk import *

reader = vtkJPEGReader()
reader.SetFileName("c:/texture.jpg")

texture = vtkTexture()
texture.SetInput(reader.GetOutput())

plane = vtkPlaneSource()

mapper = vtkPolyDataMapper()
mapper.SetInput(plane.GetOutput())

actor = vtkActor()
actor.SetMapper(mapper)
actor.SetTexture(texture)
```

Multiplication des renderers

```
#!/usr/bin/env python
#
# This example demonstrates how to use multiple renderers within a
# render window. It is a variation of the Cone.py example. Please
# refer to that example for additional documentation.
#

import vtk
import time

#
# Next we create an instance of vtkConeSource and set some of its
# properties. The instance of vtkConeSource "cone" is part of a visualization
# pipeline (it is a source process object); it produces data (output type is
# vtkPolyData) which other filters may process.
#
cone = vtk.vtkConeSource()
cone.SetHeight( 3.0 )
cone.SetRadius( 1.0 )
cone.SetResolution( 10 )

#
# In this example we terminate the pipeline with a mapper process object.
# (Intermediate filters such as vtkShrinkPolyData could be inserted in
# between the source and the mapper.)  We create an instance of
# vtkPolyDataMapper to map the polygonal data into graphics primitives. We
# connect the output of the cone souece to the input of this mapper.
#
coneMapper = vtk.vtkPolyDataMapper()
coneMapper.SetInput(cone.GetOutput())

#
# Create an actor to represent the cone. The actor orchestrates rendering of
# the mapper's graphics primitives. An actor also refers to properties via a
# vtkProperty instance, and includes an internal transformation matrix. We
# set this actor's mapper to be coneMapper which we created above.
#
coneActor = vtk.vtkActor()
coneActor.SetMapper(coneMapper)

#
# Create two renderers and assign actors to them. A renderer renders into a
# viewport within the vtkRenderWindow. It is part or all of a window on the
# screen and it is responsible for drawing the actors it has.  We also set
# the background color here. In this example we are adding the same actor
# to two different renderers; it is okay to add different actors to
# different renderers as well.
#
ren1 = vtk.vtkRenderer()
ren1.AddActor(coneActor)
ren1.SetBackground(0.0, 0.7, 0.0)
ren1.SetViewport(0.0, 0.0, 0.5, 1.0)
```

```
ren2 = vtk.vtkRenderer()
ren2.AddActor(coneActor)
ren2.SetBackground(0.7, 0.0, 0.0)
ren2.SetViewport(0.5, 0.0, 1.0, 1.0)

#
# Finally we create the render window which will show up on the screen.
# We add our two renderers into the render window using AddRenderer. We also
# set the size to be 600 pixels by 300.
#
renWin = vtk.vtkRenderWindow()
renWin.AddRenderer( ren1 )
renWin.AddRenderer( ren2 )
renWin.SetSize(600, 300)

#
# Make one camera view 90 degrees from other.
#
ren1.ResetCamera()
ren1.GetActiveCamera().Azimuth(90)

#
# Now we loop over 360 degreeees and render the cone each time.
#
for i in range(0,360):
    time.sleep(0.03)

    renWin.Render()
    ren1.GetActiveCamera().Azimuth( 1 )
    ren2.GetActiveCamera().Azimuth( 1 )
```

Exemple Java

```
package simple;
import java.awt.BorderLayout;
import javax.swing.JFrame;

import vtk.vtkActor;
import vtk.vtkConeSource;
import vtk.vtkPanel;
import vtk.vtkPolyDataMapper;

public class ExempleBasic {
    /*
      static {
      System.loadLibrary("vtkCommonJava");
      System.loadLibrary("vtkFilteringJava");
      System.loadLibrary("vtkIOJava");
      System.loadLibrary("vtkImagingJava");
      System.loadLibrary("vtkGraphicsJava");
      System.loadLibrary("vtkRenderingJava");
      System.loadLibrary("vtkHybridJava");
      }
      */

    public static void main(String[] args) {
        // Creation de la vue 3D VTK
        vtkPanel panel3D = new vtkPanel();

        // Creation des composants du pipeline
        vtkConeSource cone = new vtkConeSource();
        cone.SetHeight(3);
        cone.SetRadius(1);
        cone.SetResolution(10);

        vtkPolyDataMapper mapper = new vtkPolyDataMapper();
        vtkActor actor = new vtkActor();

        // Definition du pipeline VTK
        mapper.SetInput(cone.GetOutput());
        actor.SetMapper(mapper);
        panel3D.GetRenderer().AddActor(actor);

        // fenetre java
        JFrame fenetre = new JFrame("Exemple VTK de base");
        // Insert la vue 3D
        fenetre.getContentPane().add(panel3D, BorderLayout.CENTER);
        // finir l'initialisation de la fenetre
        fenetre.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        fenetre.setSize(200, 200);
        fenetre.setVisible(true);
    }
}
```