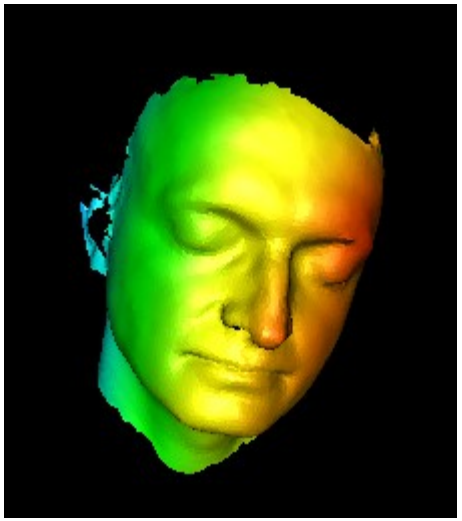
Cours Vtk



Contenu

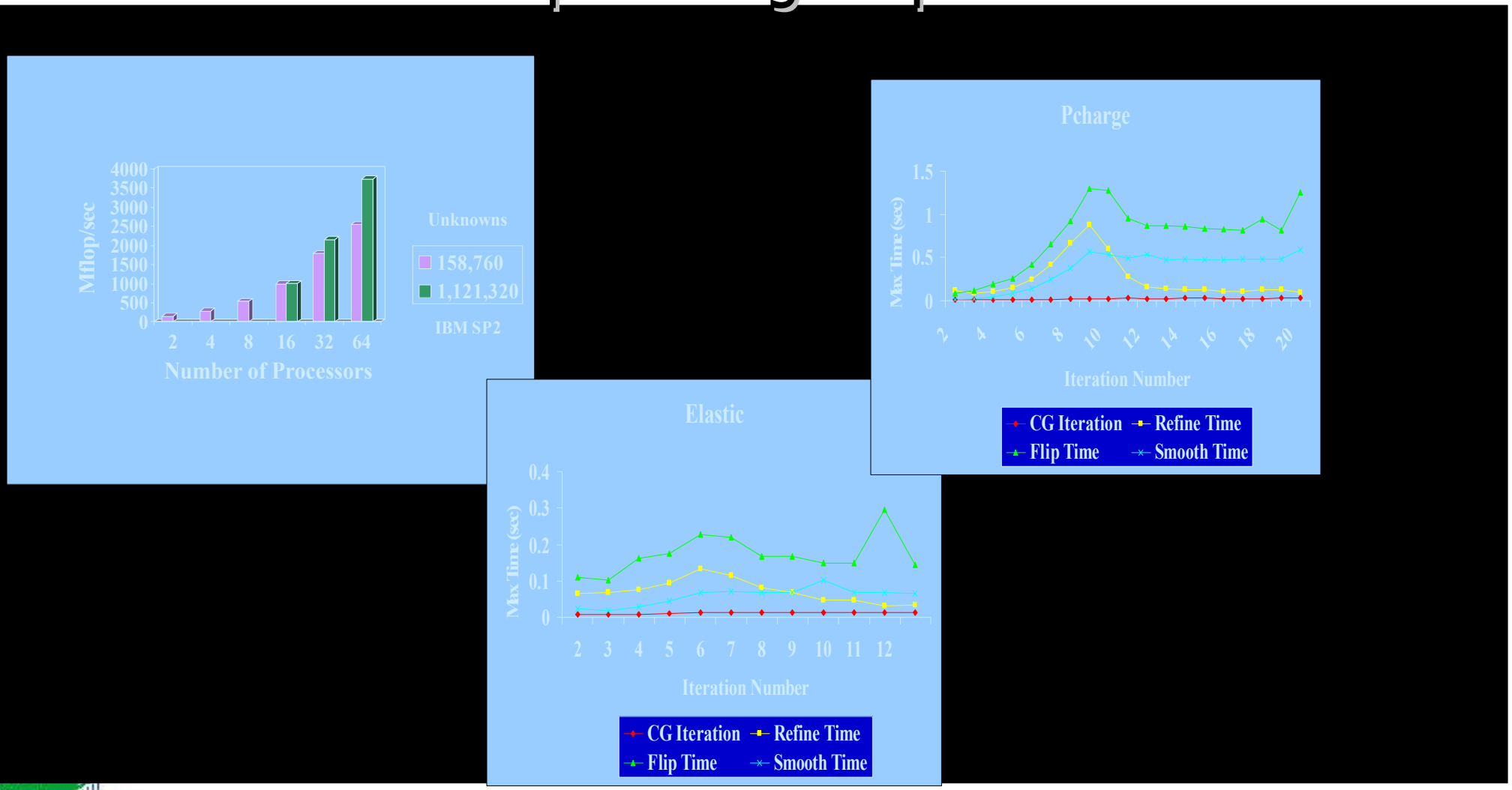
- Qu'est-ce que c'est ?
- Pourquoi VTK?
- Documentation
- Syntaxe
- POO : Quelques concepts de base
- Modèles d'objets VTK
  - »Modèle de visualisation
  - »Modèle graphique
  - »Modèle de traitement
- Exemple d'une application
- Tcl/Tk & Vtk & C++

Librairie de visualisation scientifique

- Qu'est-ce que c'est ?
- Pourquoi VTK?
  - Interpréter les données en termes visuels:
    - Les données sont trop complexes
    - Il y en a trop
  - Cela ne se substitue pas à l'analyse statistique, au filtrage, à échantillonnage
  - Fait appel aux capacités sensorielles de l'humain:
    - Reconnaissance de forme
    - Découverte de tendances

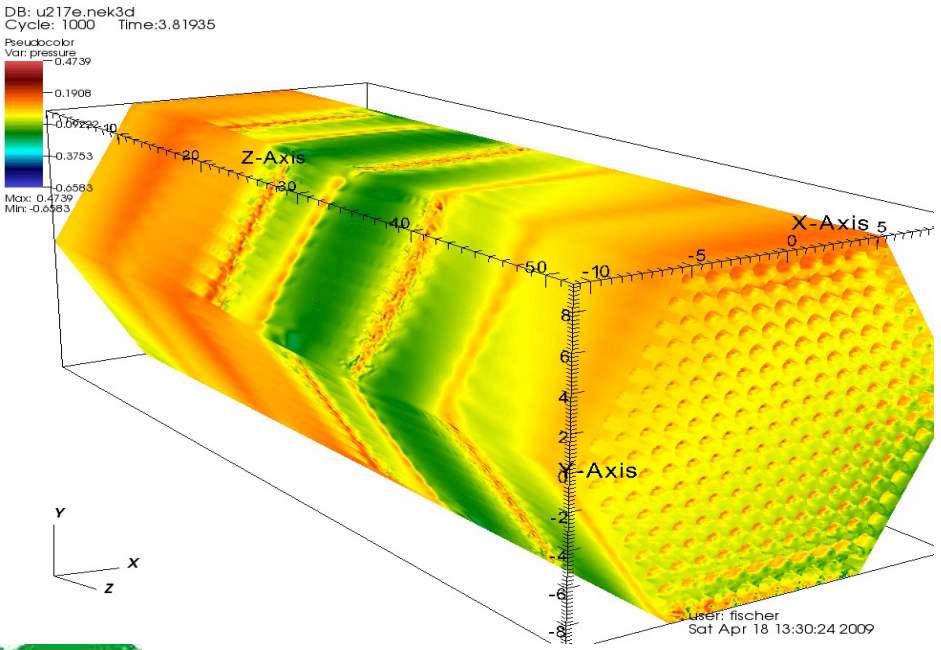
Librairie de visualisation scientifique

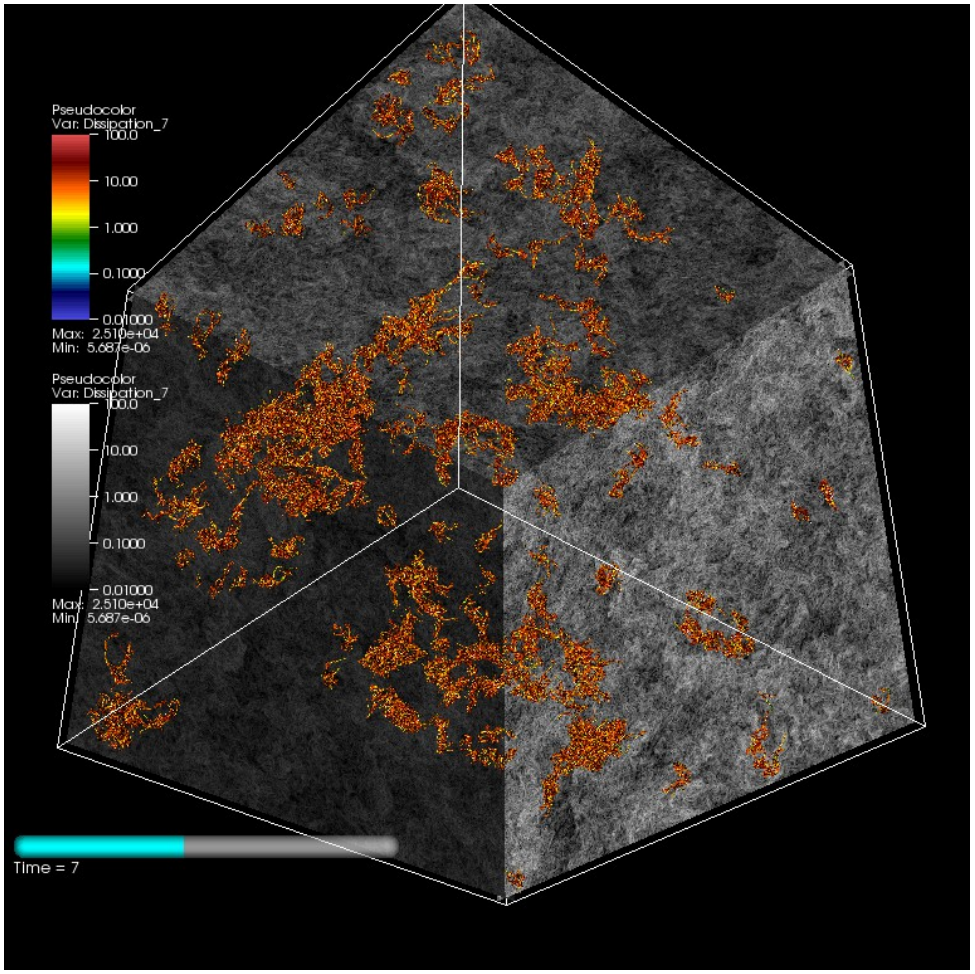
- Ceci existe depuis longtemps



Librairie de visualisation scientifique

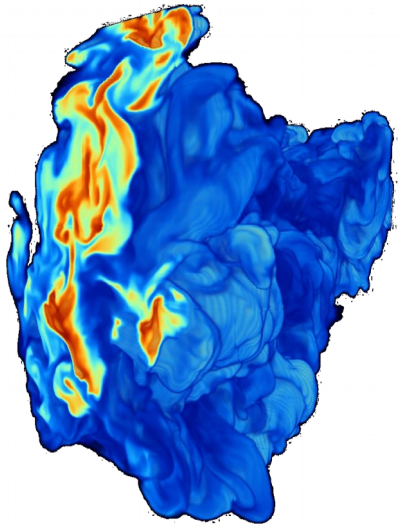
- Nous nous intéressons aux données plus complexes
  - Geométrie complexe
  - multi-dimentionnelles
  - Mesurées
  - Calculées

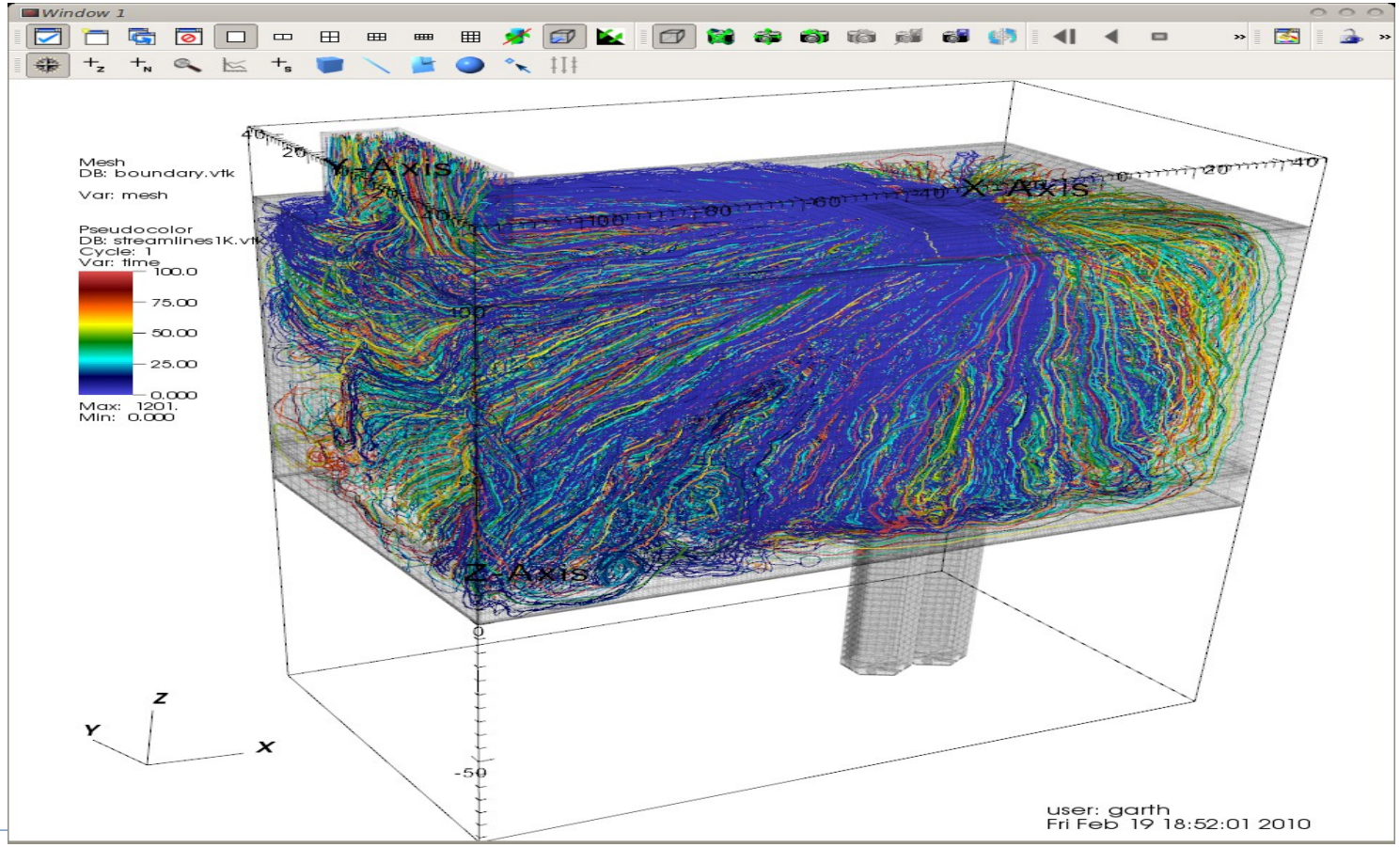




Librairie de visualisation scientifique

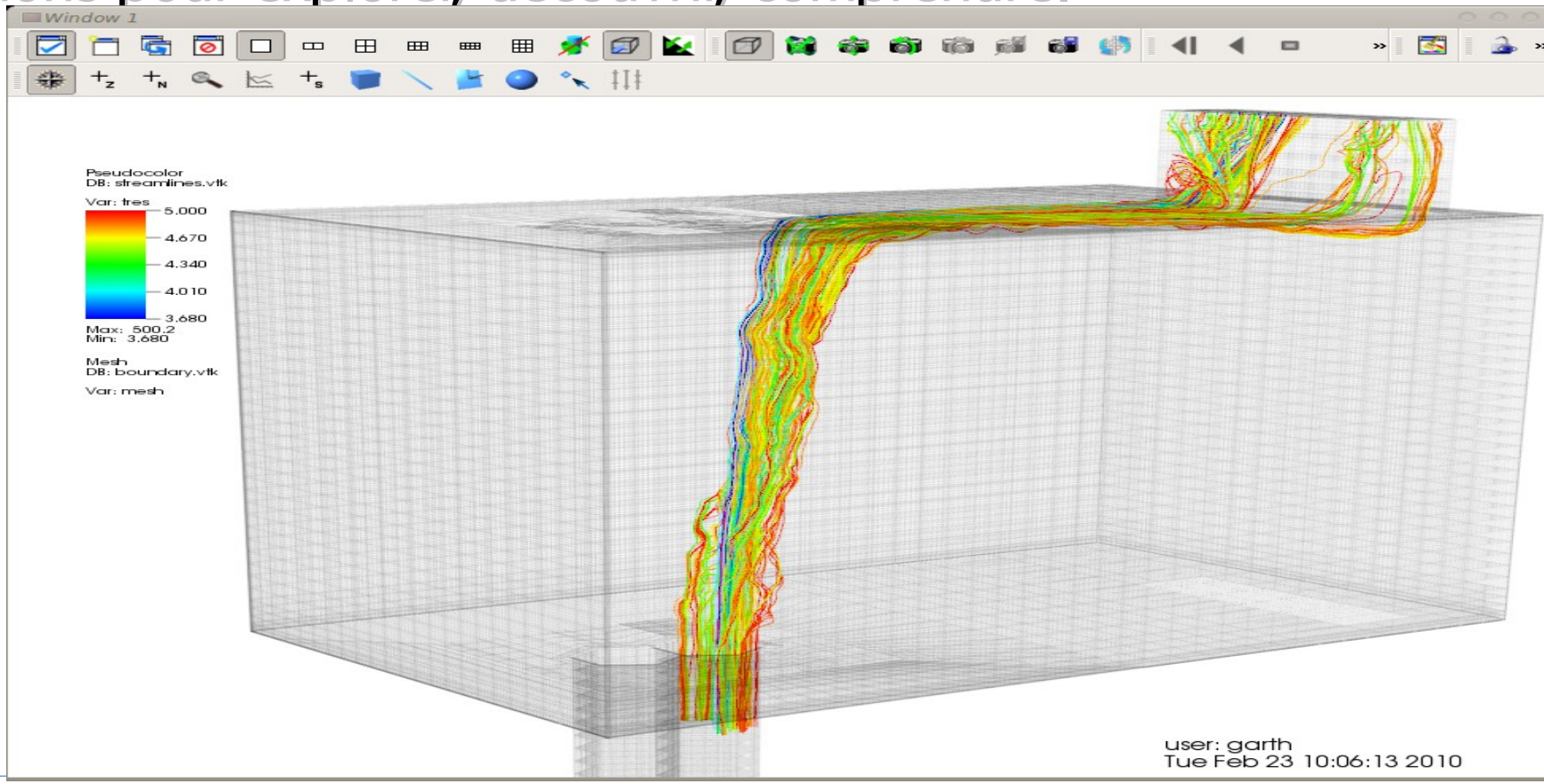
- Nous nous intéressons aux données plus complexes
  - Geométrie complexe
  - multi-dimentionnelles
  - Mesurées
  - Calculées



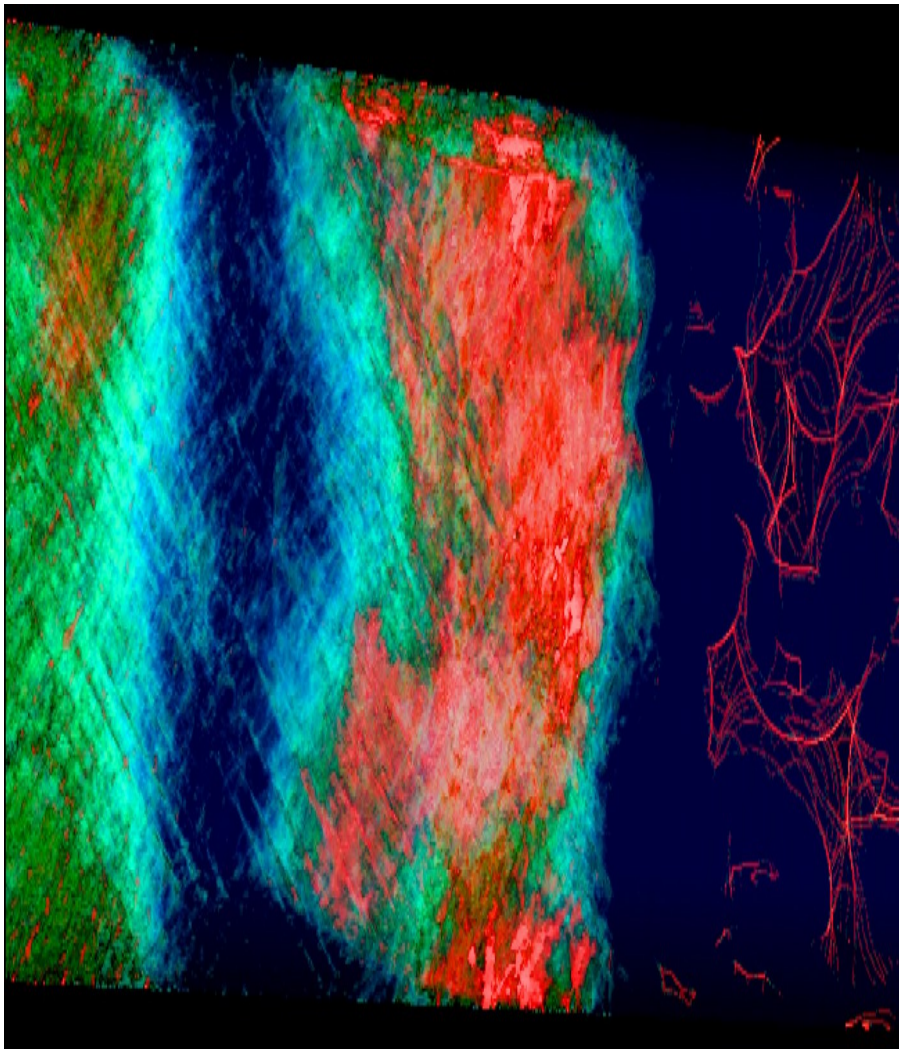


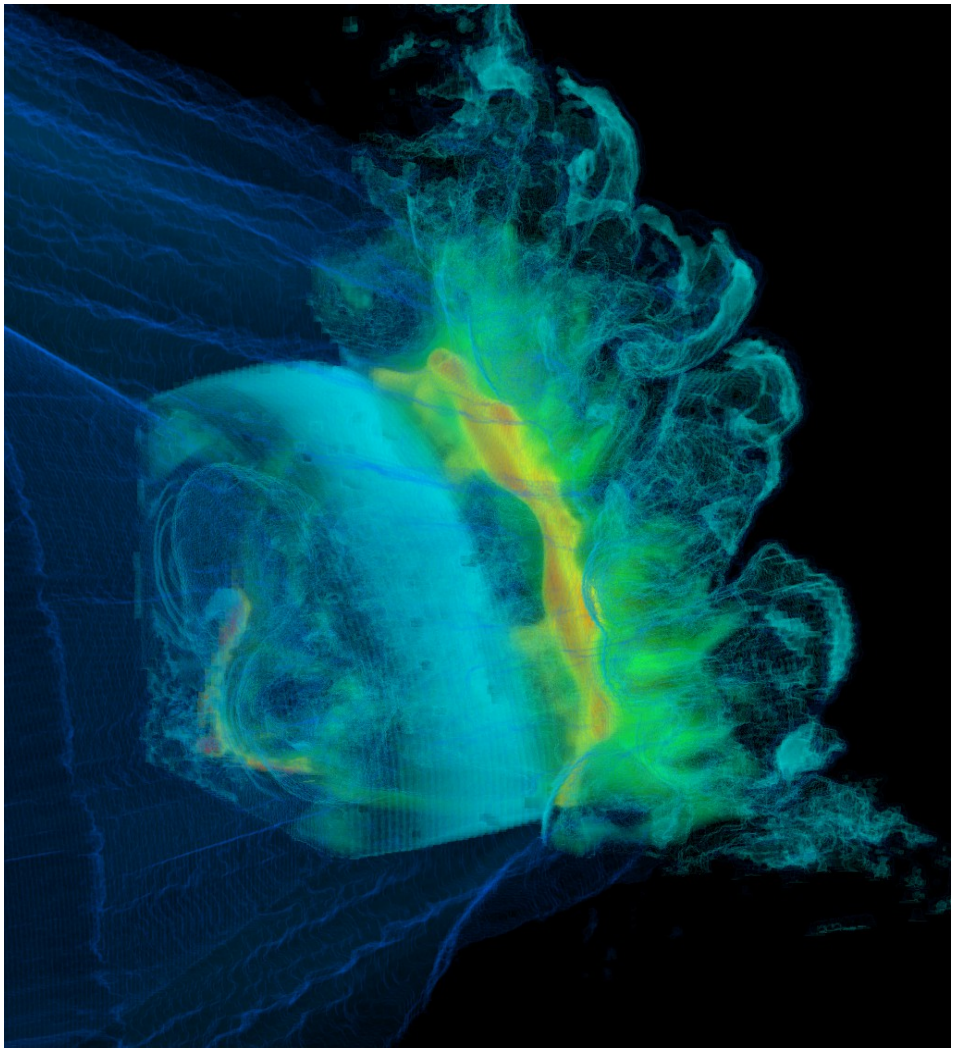
La Visualisation est plus que le graphisme

- Données en 3D ou plus.
- Transformations des données fréquente
- Interactions pour explorer, découvrir, comprendre.



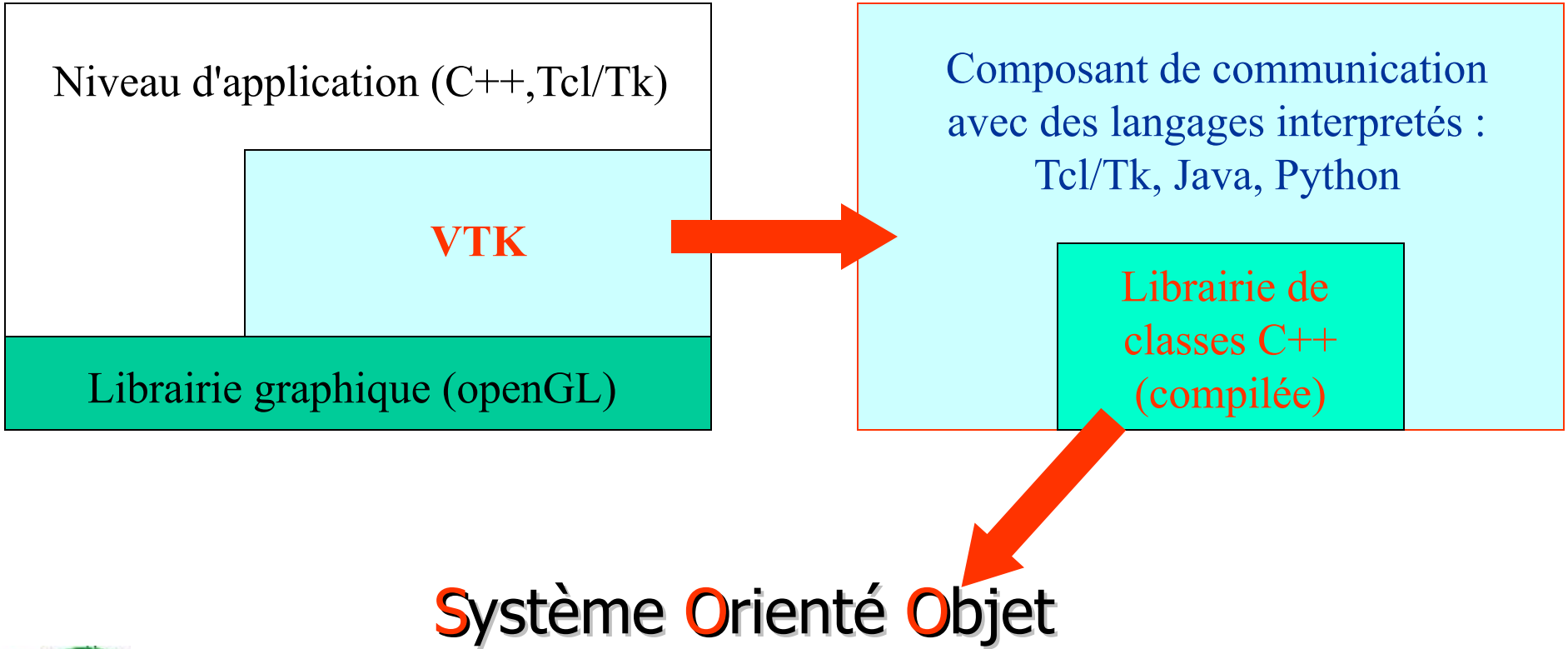
La Visualisation est plus que le graphisme





Qu'est-ce que c'est ?

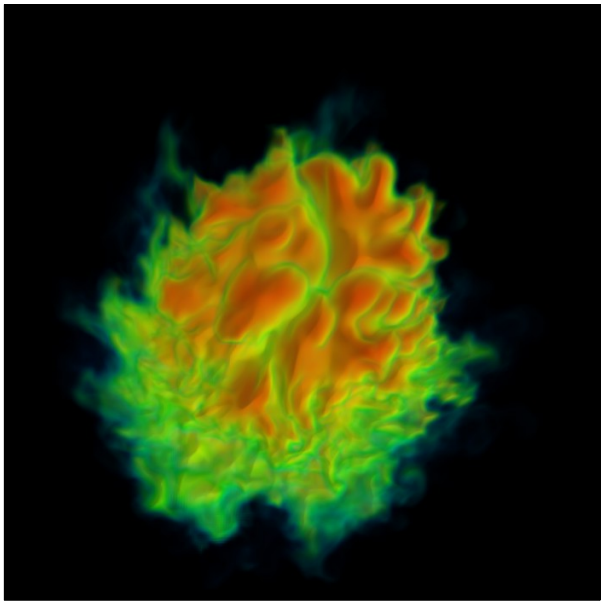
Visualization ToolKit



Exemple 2

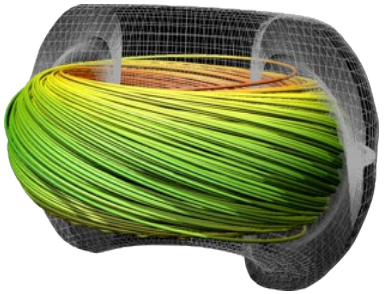
```
    vtkRenderer *ren = vtkRenderer::New();
    ren->AddActor(coneActor);
    ren->AddActor(sphereActor);


   vtkRenderWindow *renwin = vtkRenderWindow::New();
   renwin->AddRenderer(ren);
  renwin->SetSize( 400, 500 );



vtkTransform* transform2 = vtkTransform::New();



   renwin->Render();
```

Pourquoi VTK?



- Gratuit
- Code source C++
- POO
- Extensible
  - Possibilité de création de nouvelles classes
- Plate-forme/librairie indépendante
- Algorithmes avancés et nombreux
- Conversion des données en images
- Nombreux utilisateurs + liste de discussion

Documentation



- Livres
  - The VisualizationToolkit

    Schroeder, Martin, Lorenson. Prentice Hall.
  - VTK User's Guide

    Schroeder, Martin. Kitware, Inc.

- Page web
  - http://www.kitware.com /vtk.html

    Software, FAQ's, listes de discussion, exemples, links, etc

- Nombreux exemples sur le web

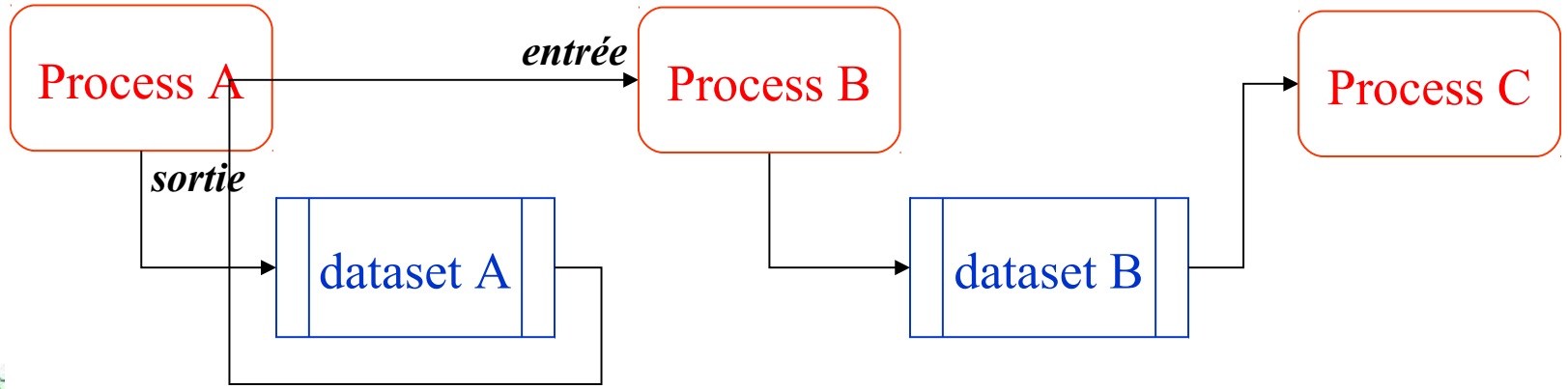
Modèle de visualisation

- Deux types d'objets:
  - Objets "data"

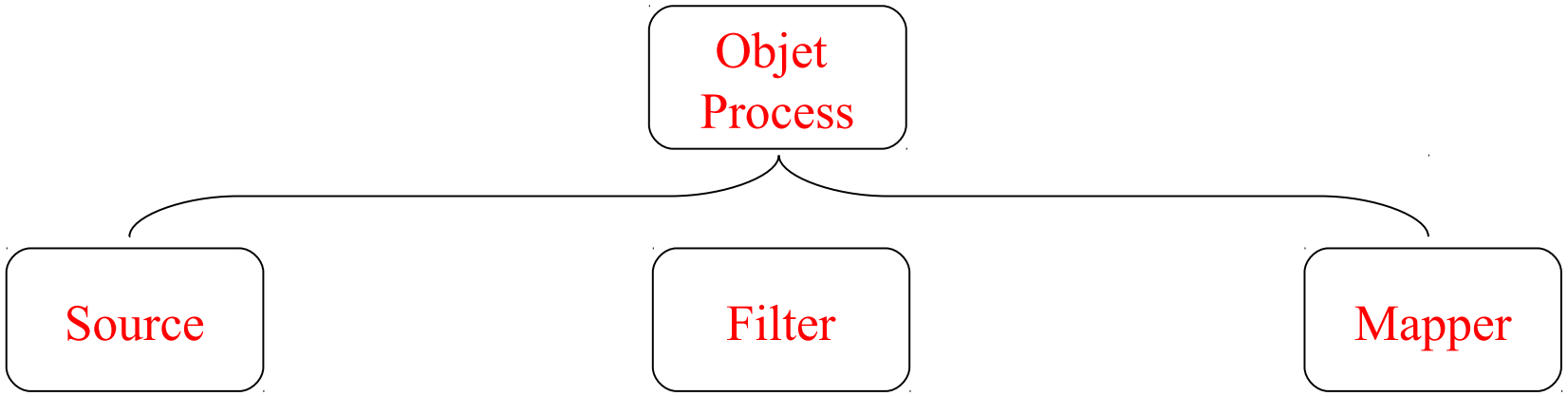
    Données qui circulent dans le pipeline

    Appelés datasets
  - Objets "process"

    Modules ou composants algorithmiques du pipeline



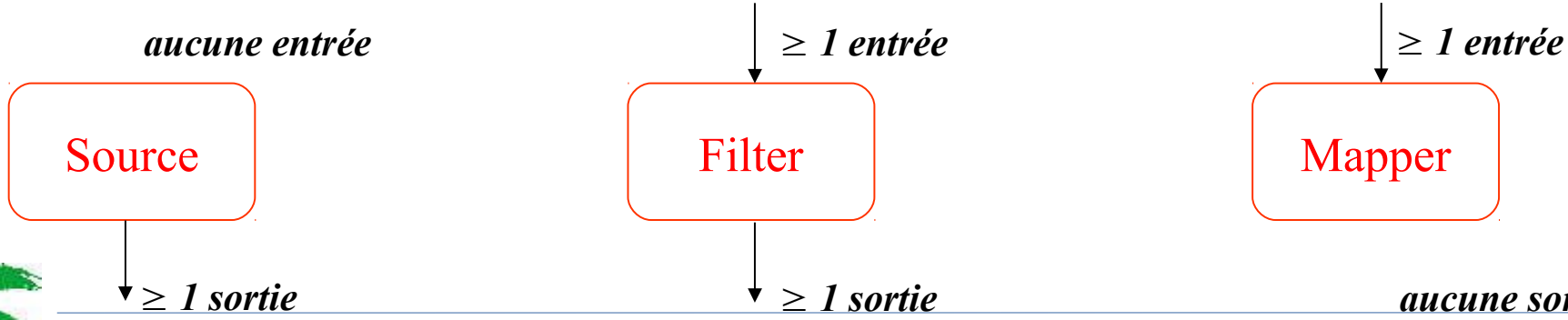
Objets Process



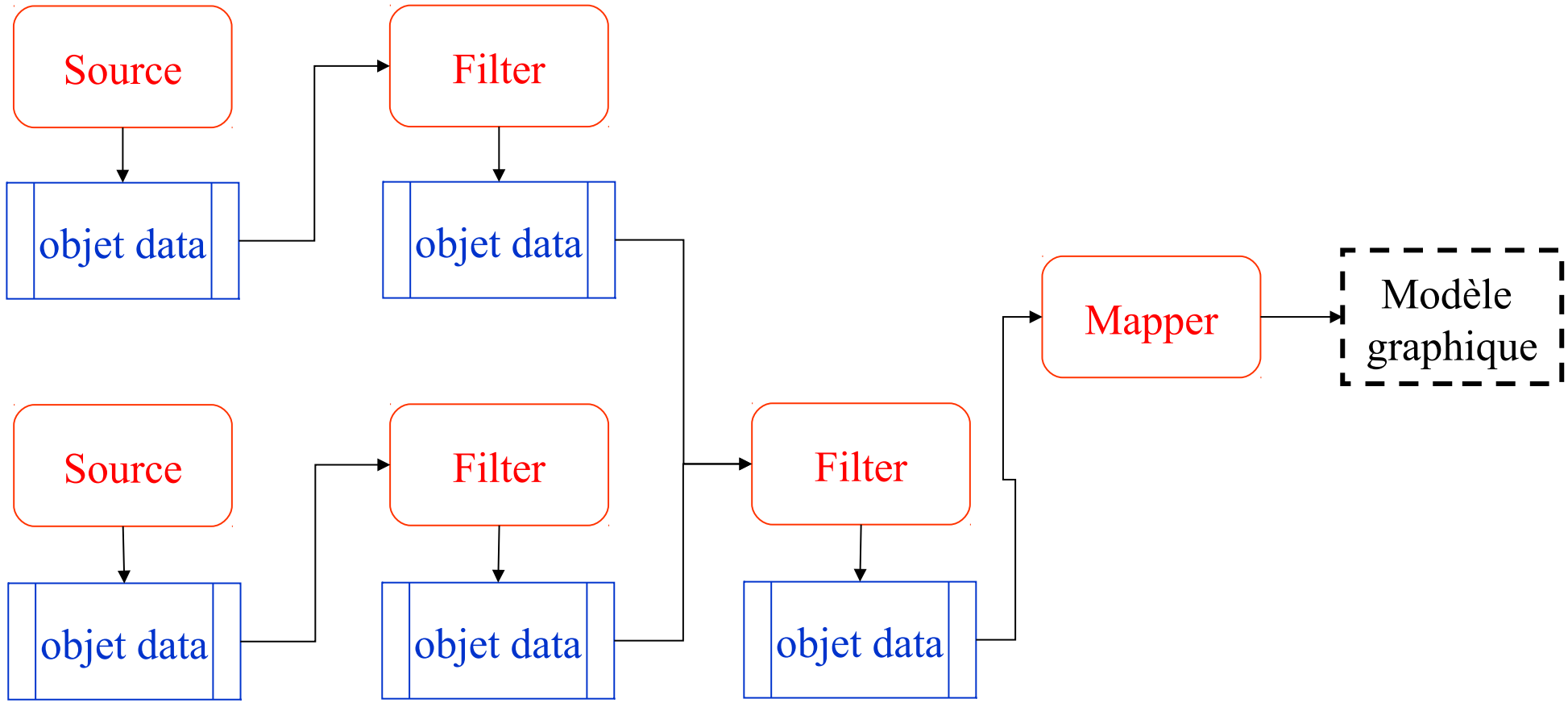
Démarre le pipeline
- Lecture de données externes
- Génération de nouvelles données (graphiques)

Traite les objets donnée
- Reçoit une ou plusieurs entrées
- Génère une ou plusieurs sorties

Finit le pipeline
- Génère des primitives graphiques
- Communique le pipeline de visualisation avec le modèle graphique

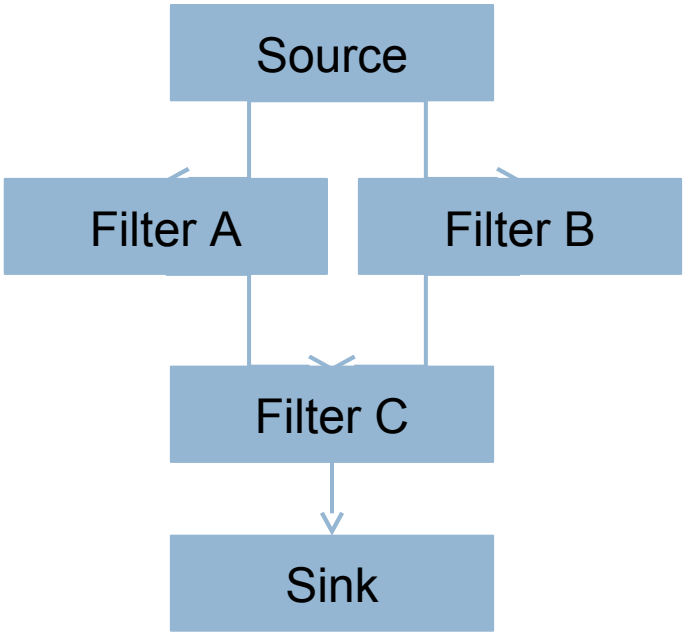


Pipeline de visualisation



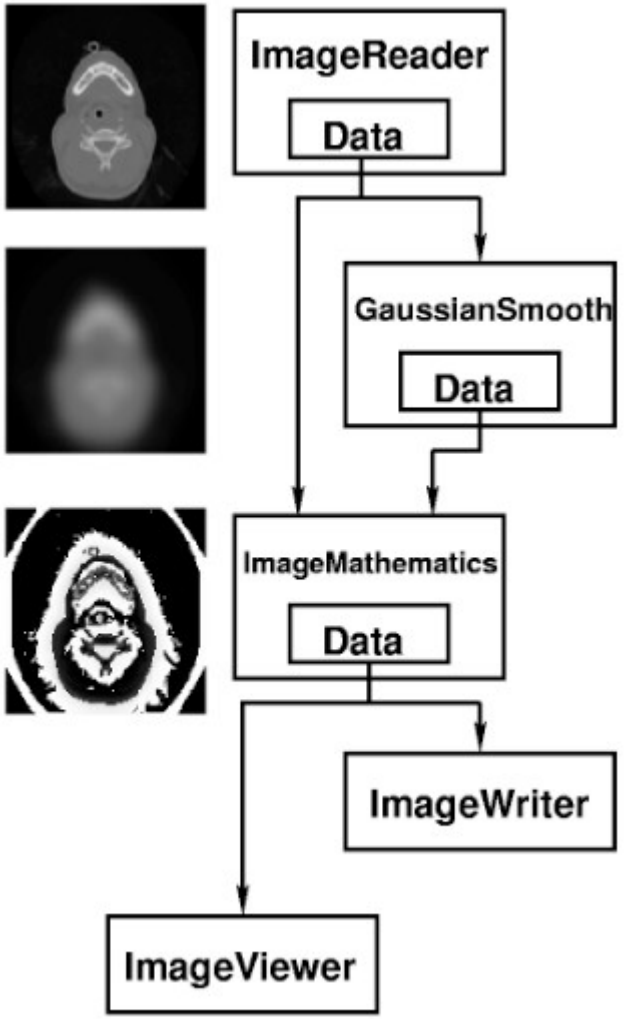
Vtk – Modèle DataFlow

- Avantages:
  - Flexible
  - plusieurs I/O possibles
  - Interopérabilité des modules
  - Parallélisme intrinsèque
  - Facile à étendre

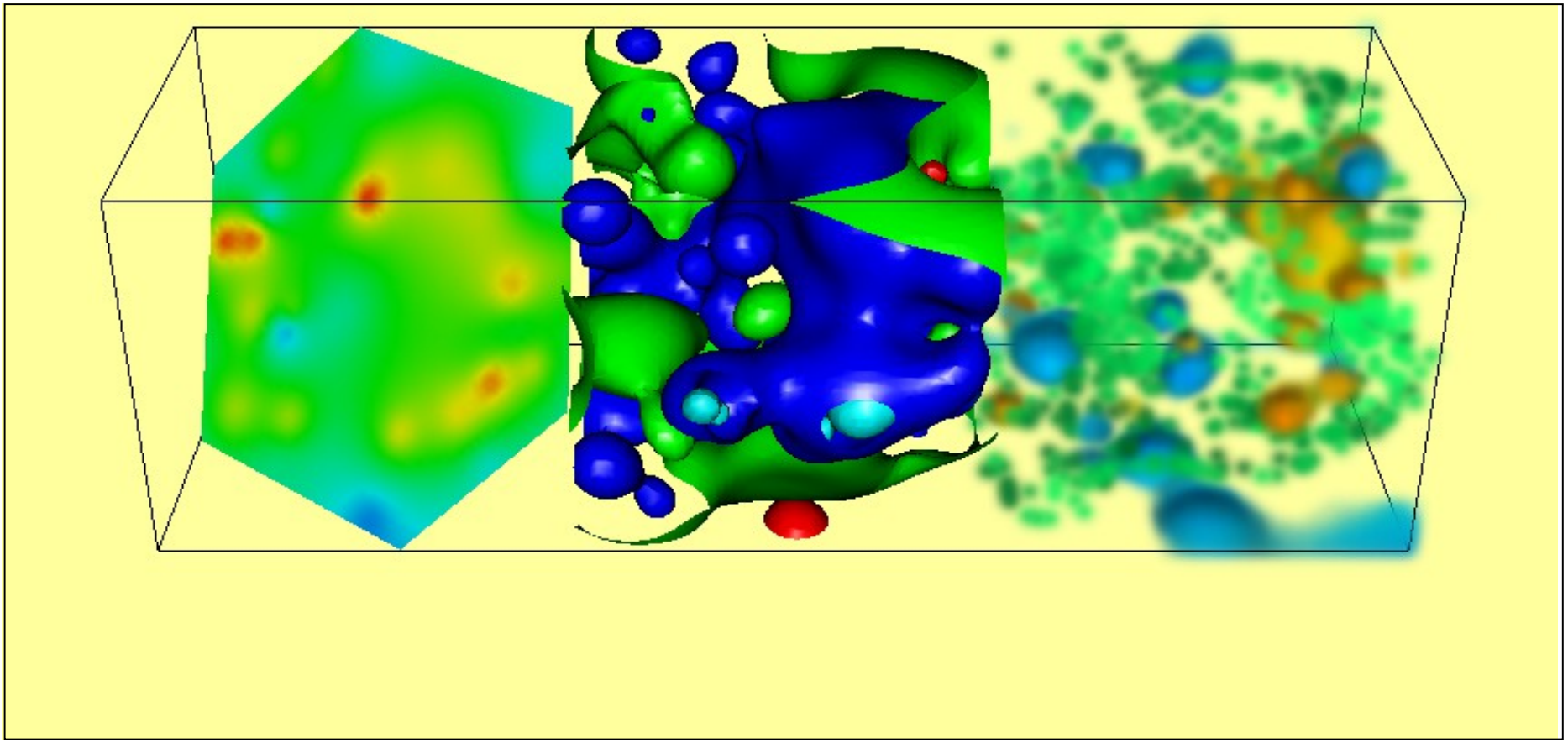


Vtk – Modèle DataFlow

- Avantages:
  - Flexible
  - plusieurs I/O possibles
  - Interopérabilité des modules
  - Parallélisme intrinsèque
  - Facile à étendre

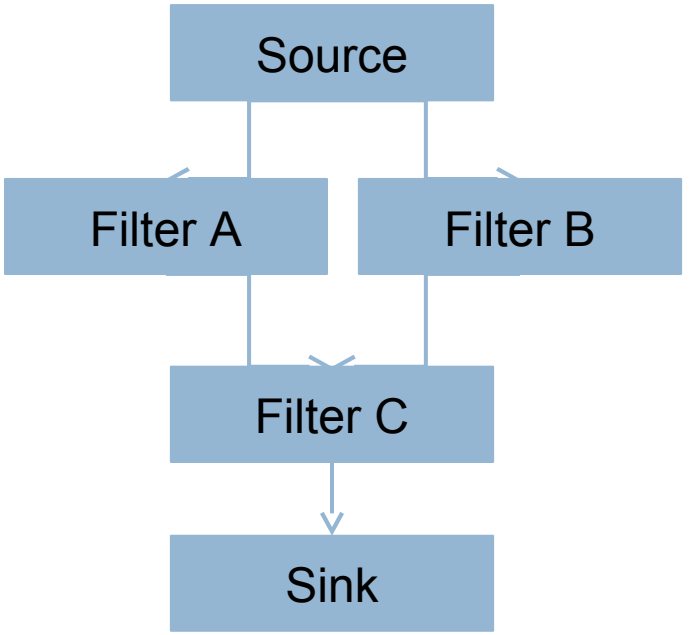


Exemple - Volume

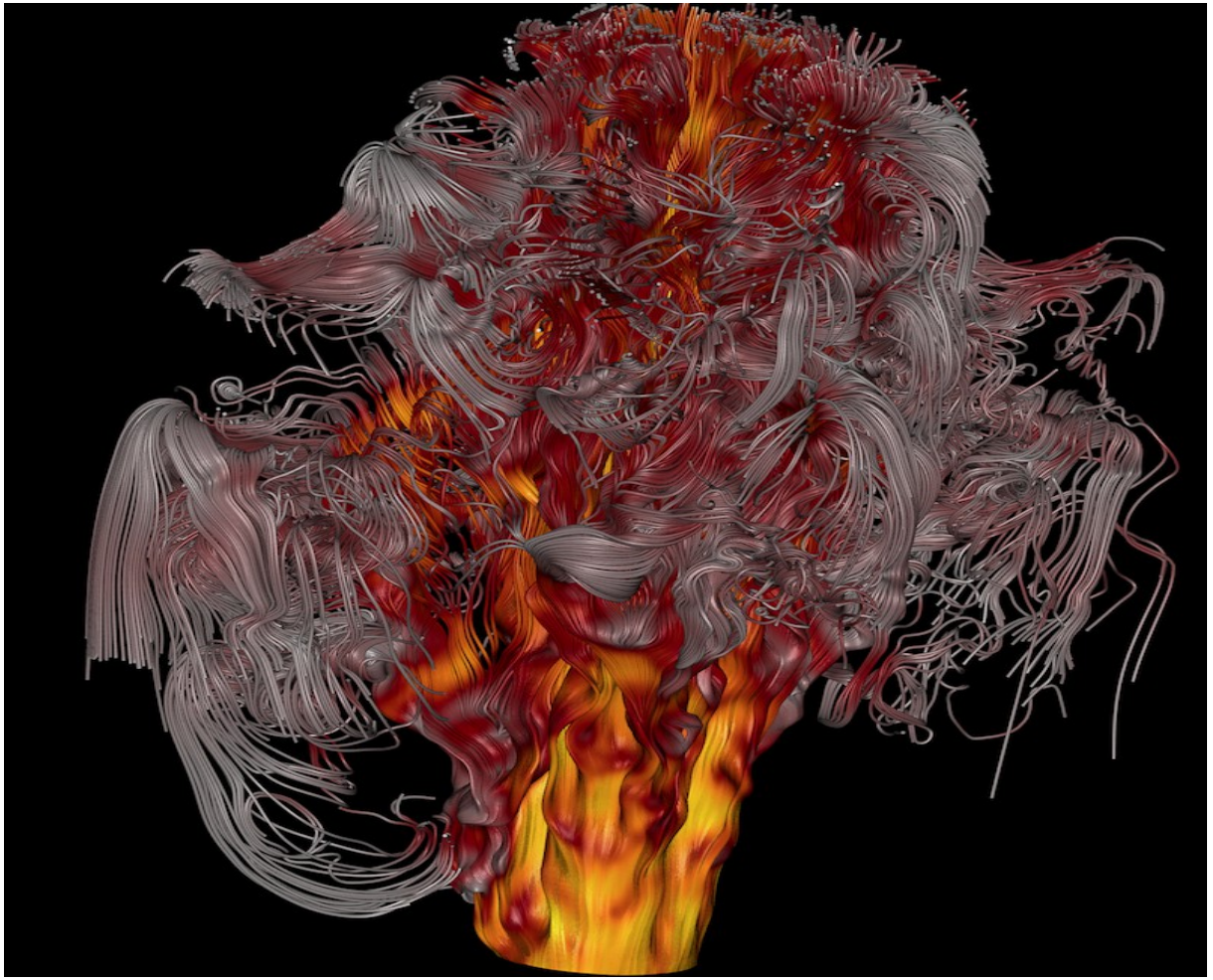


Vtk – Modèle DataFlow

- Contre:
  - exécution en étages
  - Algos exécutés l'un après l'autre (pb du cache)
  - Pb de l'occupation mémoire
  - Souplesse du modèle de données?



VTK
Un système de Visualisation Orienté Objet



Vtk - Système de Visualisation

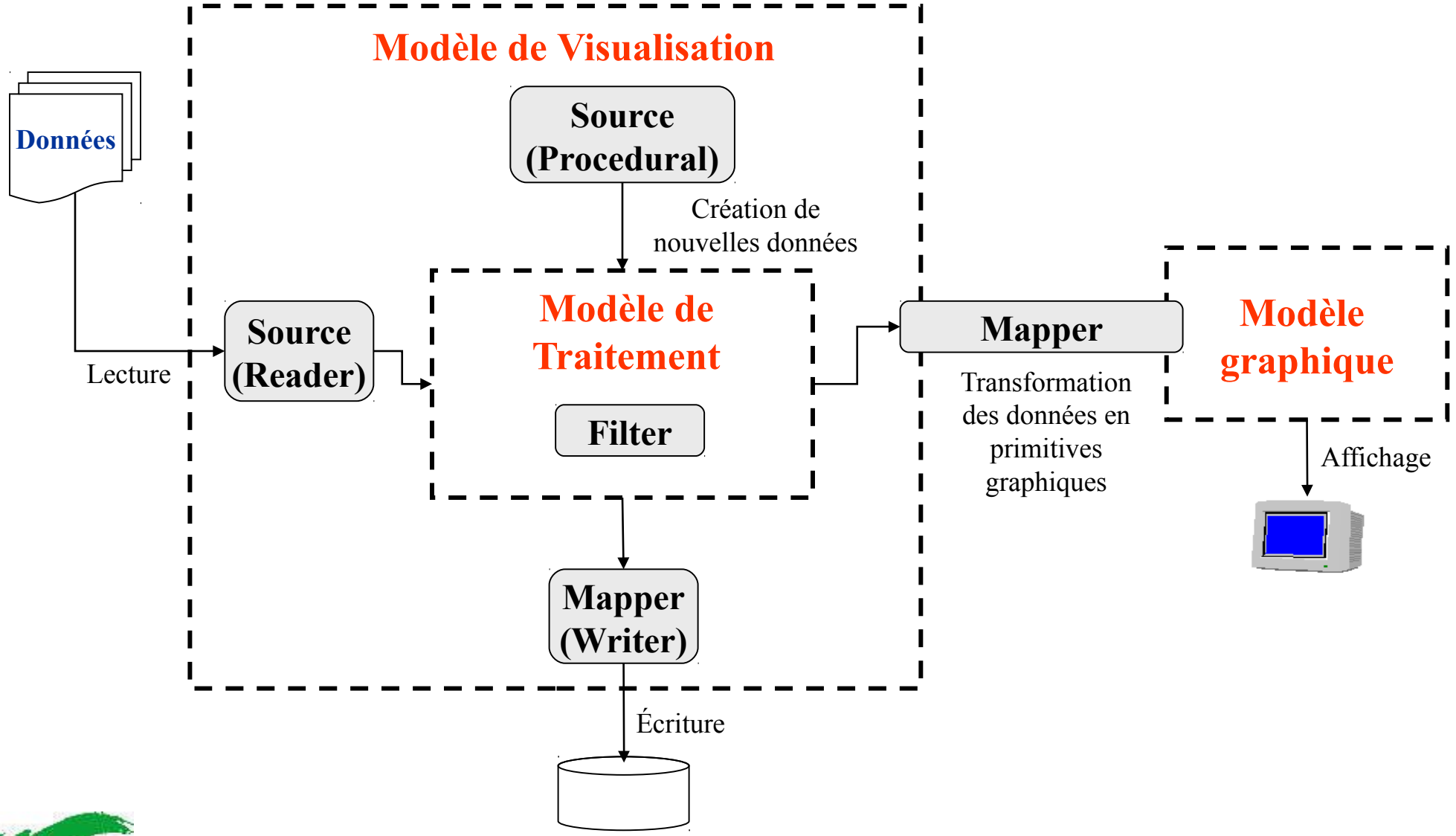
- Les objets VTK sont organisés en trois modèles:
  - Modèle de visualisation

    Étape 1 : Représentation géométrique des données
  - Modèle graphique

    Étape 2 : Rendu de la représentation géométrique
  - Modèle de traitement

    Traitement d'images

Vtk - Système de Visualisation

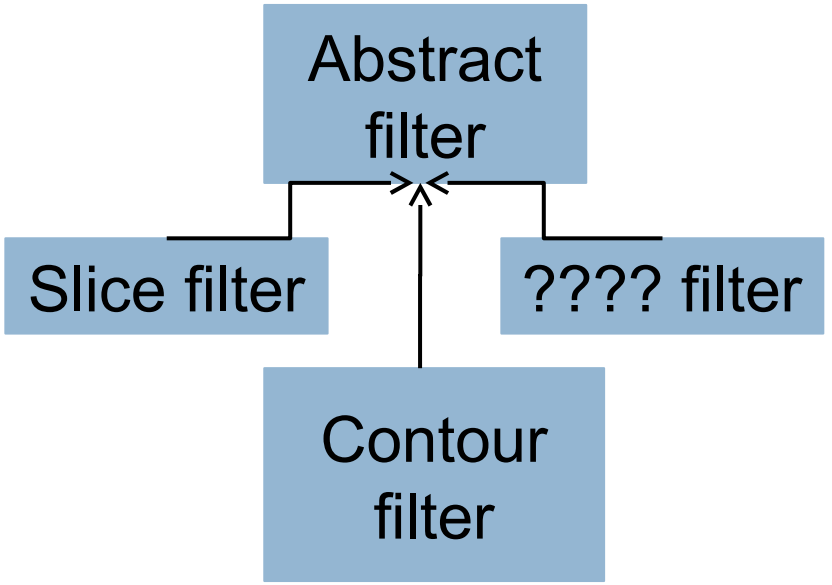


Vtk - Système de Visualisation

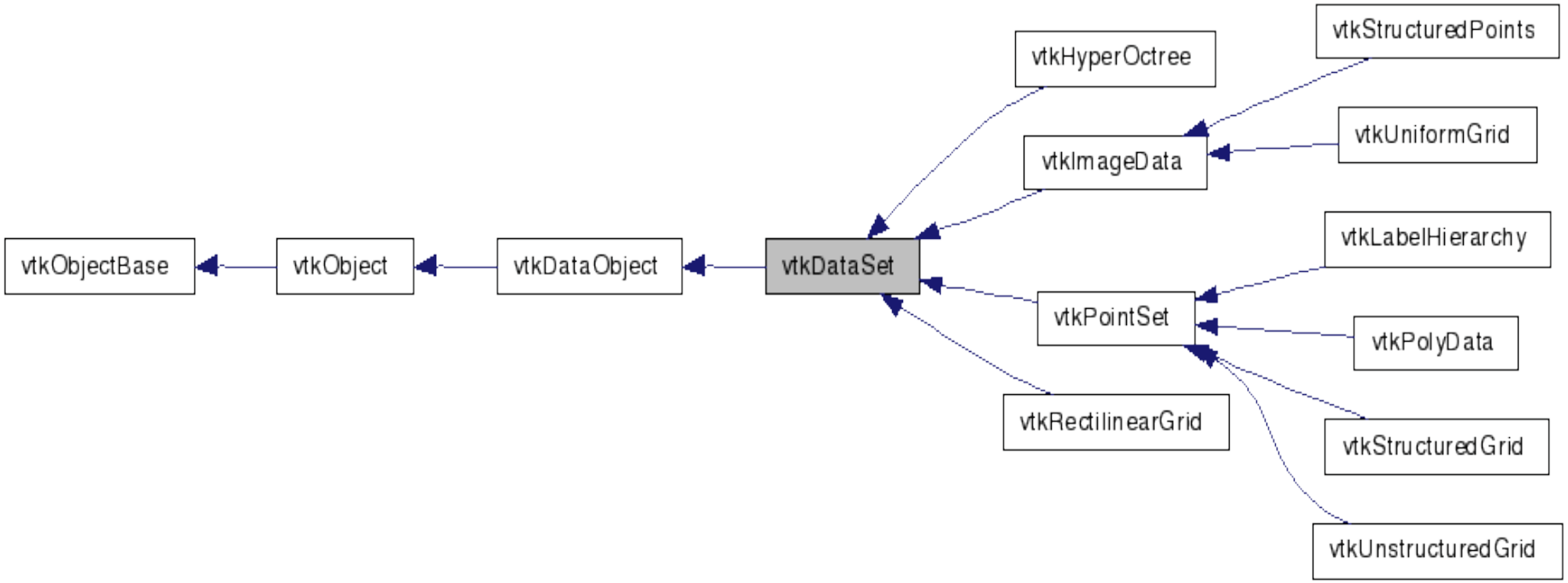

```
vtkSphereSource *sphere = vtkSphereSource::new();
vtkElevationFilter *colorIt = vtkElevationFilter::New();
   colorIt->SetInputConnection(Sphere->GetOutputPort());
vtkDataSetMapper *mapper = vtkDataSetMapper::New();
   mapper->SetInputConnection(colorIt->GetOutputPort());
vtkActor *actor = vtkActor::New();
   actor->SetMapper(mapper);
```

Vtk – Modèle DataFlow

- Pour:
  - avec la POO:
  - Facilité d'extension
  - Dérivation:
    - filtres
    - sources
    - puits



POO massive


vtkObjectBase
vtkObject
vtkDataObject
vtkDataSet
vtkHyperOctree
vtkImageData
vtkStructuredPoints
vtkUniformGrid
vtkLabelHierarchy
vtkPointSet
vtkPolyData
vtkRectilinearGrid
vtkStructuredGrid
vtkUnstructuredGrid

Exemple

```
vtkRenderer *renderer = vtkRenderer::New();
  vtkRenderWindow *renWin = vtkRenderWindow::New();
    renWin->AddRenderer(renderer);
  vtkRenderWindowInteractor *iren = vtkRenderWindowInteractor::New();
    iren->SetRenderWindow(renWin);

  vtkSphereSource *sphere = vtkSphereSource::New();
    sphere->SetPhiResolution(12); sphere->SetThetaResolution(12);

  vtkElevationFilter *colorIt = vtkElevationFilter::New();
    colorIt->SetInputConnection(sphere->GetOutputPort());
    colorIt->SetLowPoint(0,0,-1);
    colorIt->SetHighPoint(0,0,1);

  vtkDataSetMapper *mapper = vtkDataSetMapper::New();
    mapper->SetInputConnection(colorIt->GetOutputPort());

  vtkActor *actor = vtkActor::New();
    actor->SetMapper(mapper);

  renderer->AddActor(actor);
  renderer->SetBackground(1,1,1);
  renWin->SetSize(450,450);

  renWin->Render();
```

espace de rendu

Fenetre

Interaction

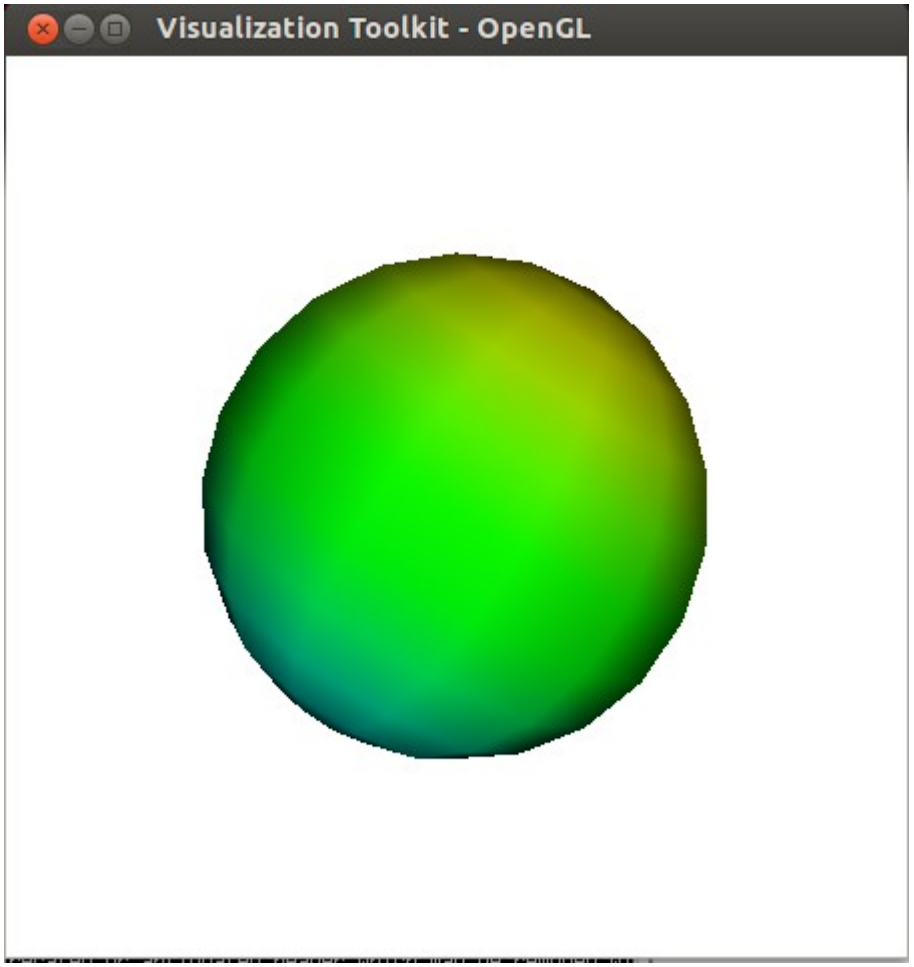
Données

Modification

Passage aux données graphiques

Création de l'acteur

Lien avec le renderer


Visualization Toolkit - OpenGL

Objets Data ou Datasets

- Un dataset est composé de :
  - Structure topologique (liste de cells)

    Détermine la forme de l'objet (triangle, sphère, maillage, etc.)

    Ne varie pas avec certaines transformations géométriques (translation, rotation et changement d'échelle)

    Un objet contient une ou plusieurs cells
  - Structure géométrique (liste de points)

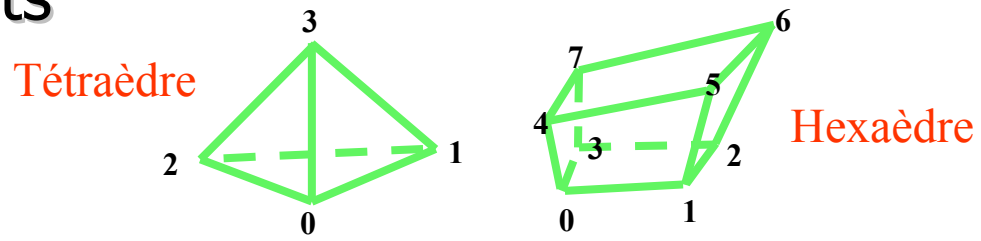
    Instanciation de la structure topologique

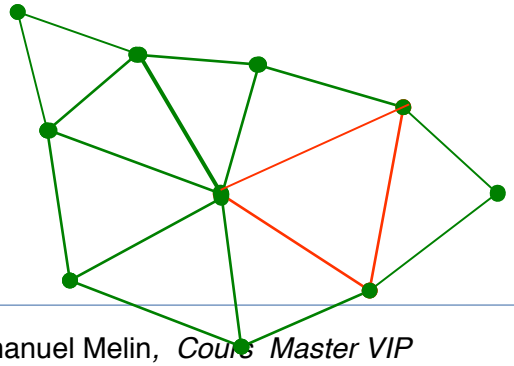
    Coordonnées des points qui conforment les cells
  - Attributs

    Information complémentaire associée aux points ou aux cells (température en un point, masse d'une cell, etc)

Cells

- Atomes qui conforment un dataset
- Une cell est une organisation topologique de points (coordonnées x,y,z)
- Définie par :
  - Type
  - Liste ordonnée de points

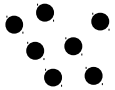
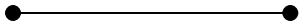
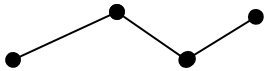
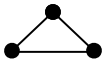


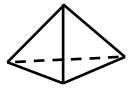
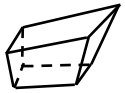
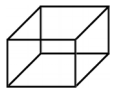


Objet ➡ Maillage = liste de triangles

Cell ➡ Triangle = liste de points

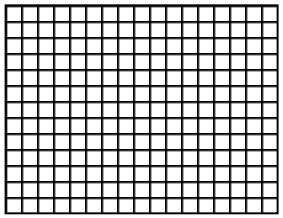
Types de cells

- Vertex 
- Polyvertex 
- Line 
- Polyline 
- Triangle 
- Triangle Strip 
- Quadrilateral 

- Pixel 
- Polygon 
- Tetrahedron 
- Hexahedron 
- Voxel 

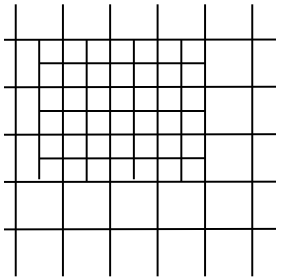
Types de Datasets

Points structurés



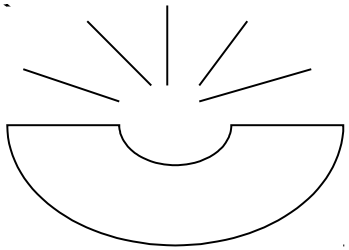
vtkStructuredPoints
Images 2D et 3D

Grille rectiligne



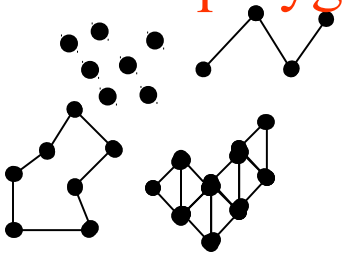
vtkRectilinearGrid
Maillage carré

Grille structurée



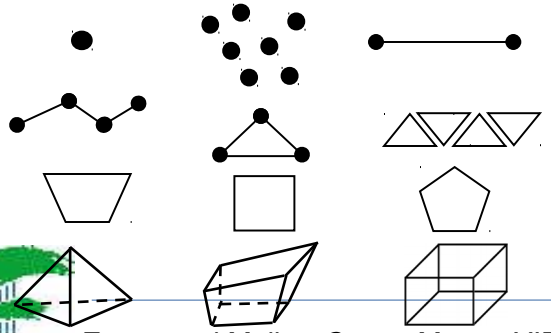
vtkStructuredGrid
Maillage 2D

Données polygonales



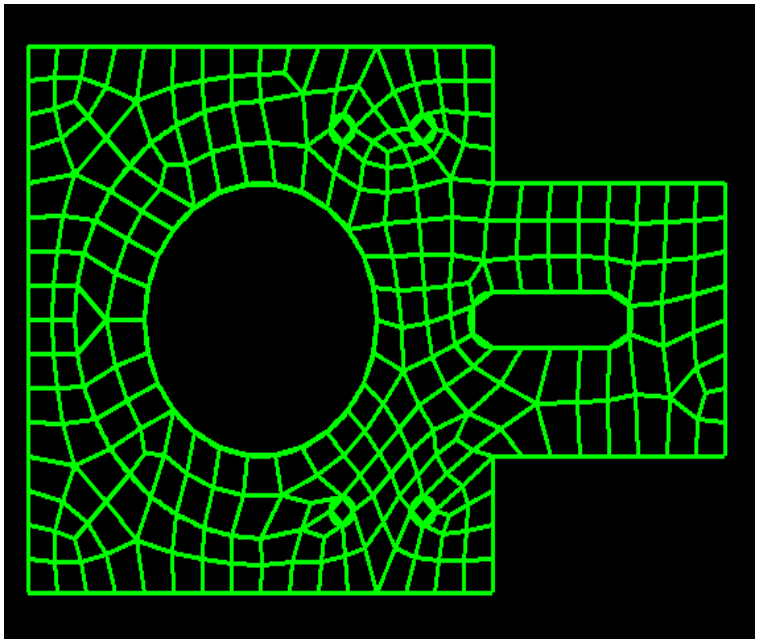
vtkPolyData
Points, lignes, polygones

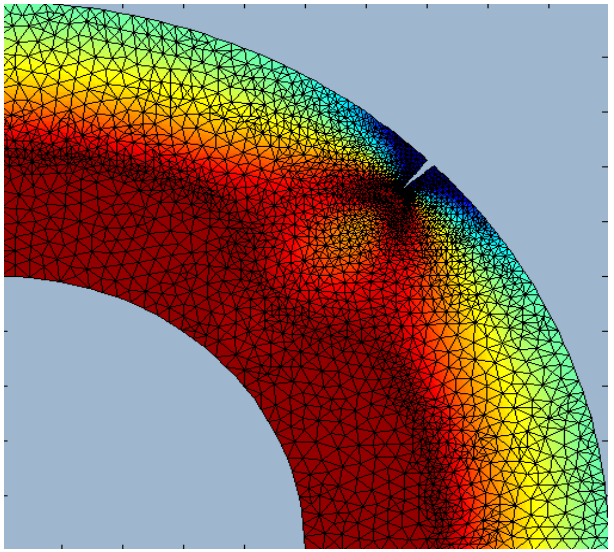
Grille non structurée

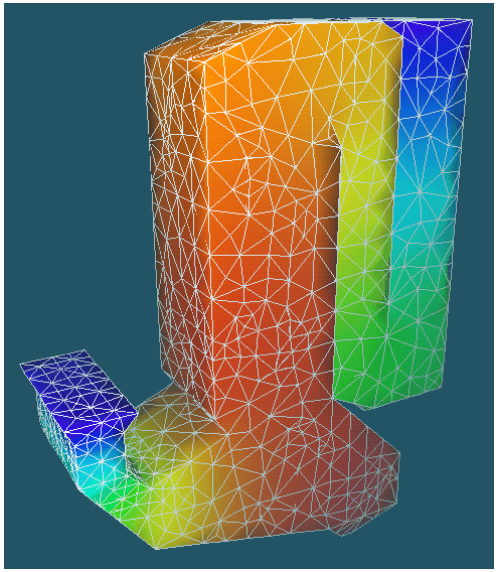


vtkUnstructuredGrid
Maillage 2D / 3D non structuré

Exemples de maillages







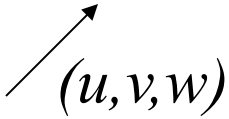
- Les cells peuvent avoir des différentes formes et tailles
  - 2D : triangles, quadrilatéres, ...
  - 3D : tétraèdres, hexaèdres, pyramides, …
- Les maillages peuvent contenir un ou plusieurs type de cells

Type des attributs associés aux points et aux cells d'un dataset

Scalaire

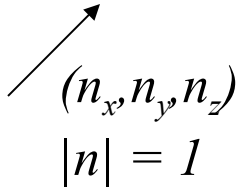
vtkScalars
Valeur simple

Vecteur



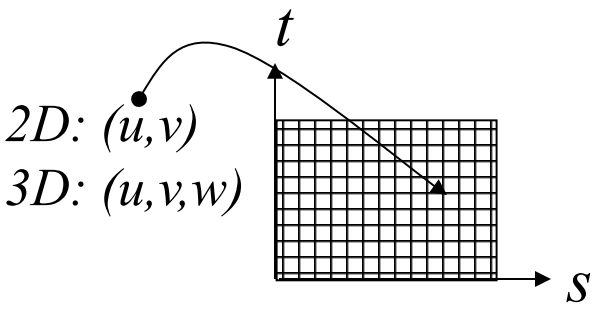
vtkVectors
Magnitude et direction (3D)

Normal



vtkNormals
Direction (3D)

Coordonnée de texture



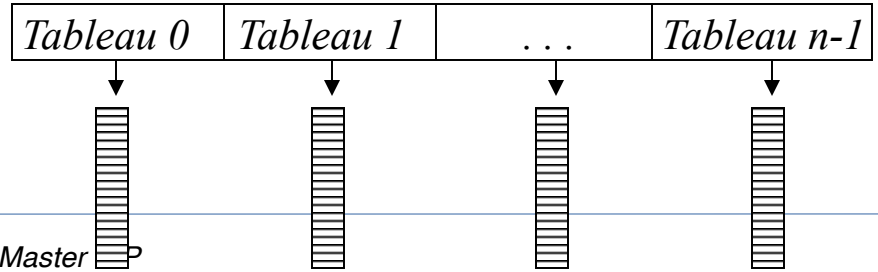
vtkTCoords
Correspondance entre un indice et une carte de textures

Tenseur

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

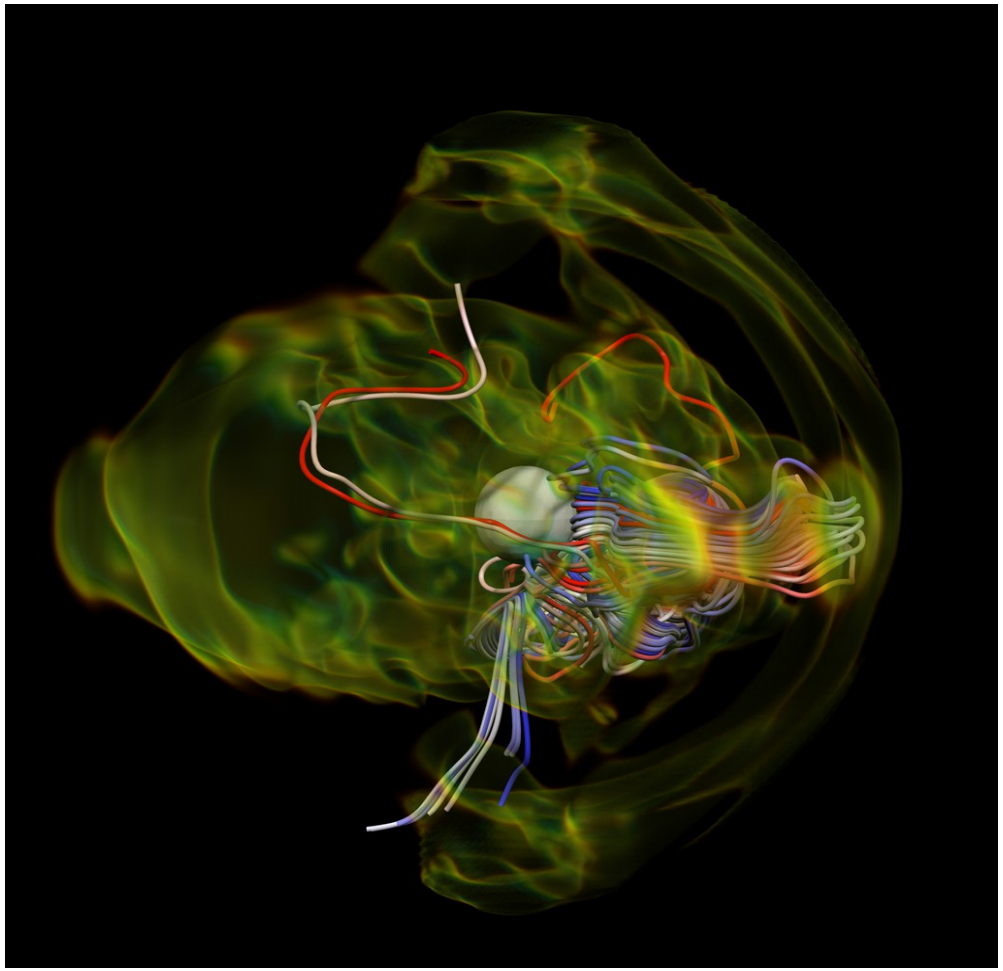
vtkTensors
Matrice n x n

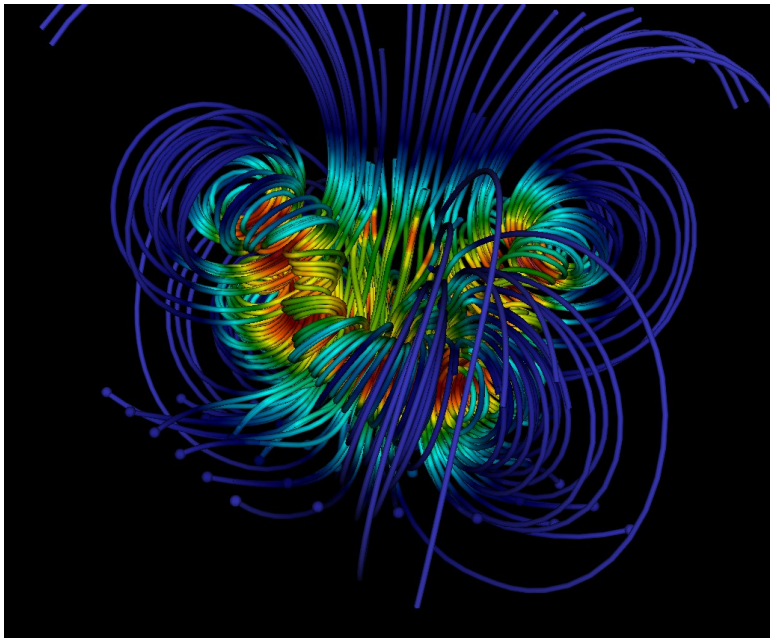
Tableau de données

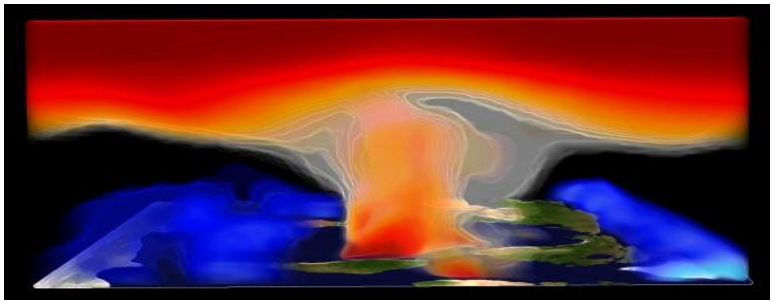


vtkFieldData
Tableau de tableaux
Chaque tableau peut être de différent type

Modèle Graphique



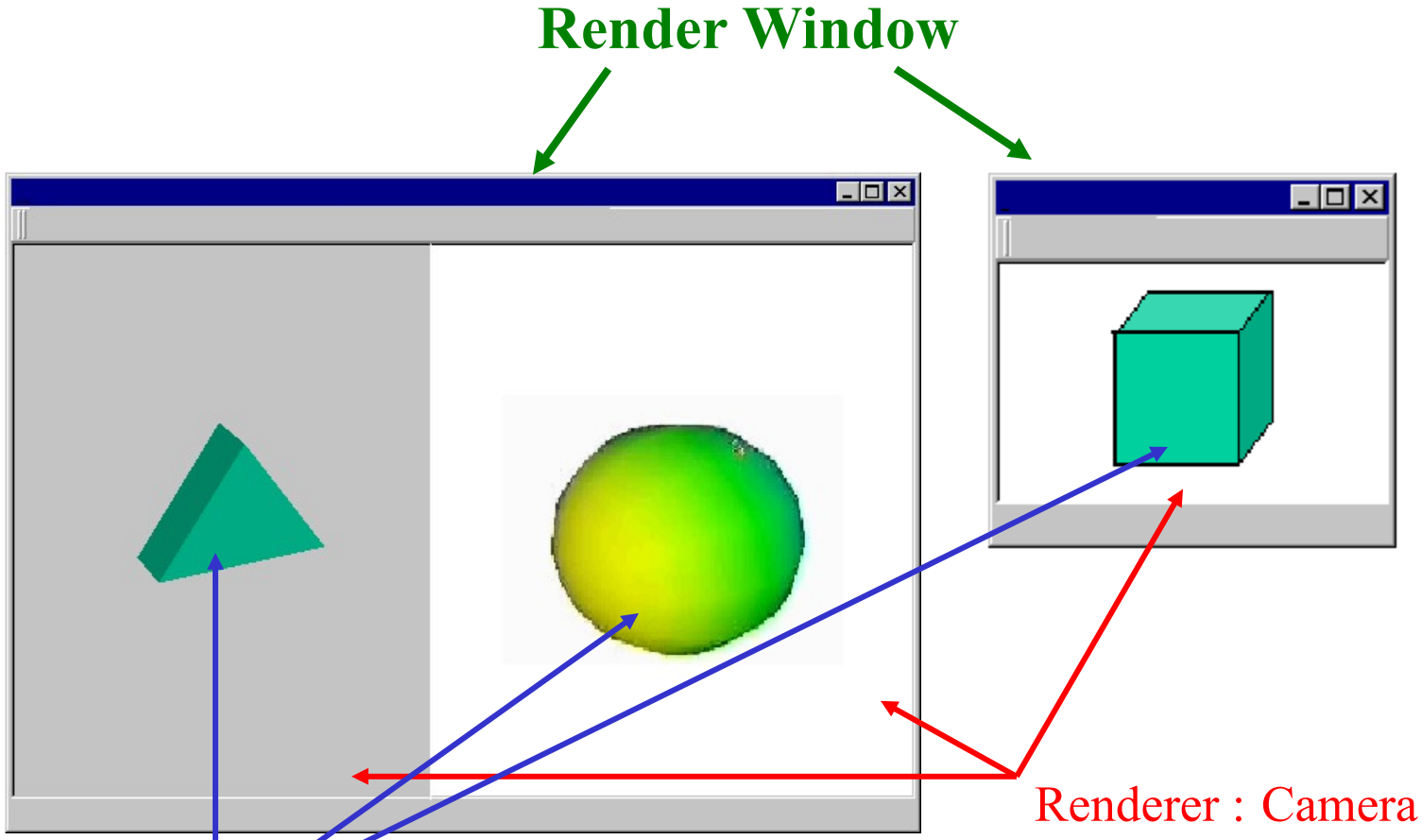




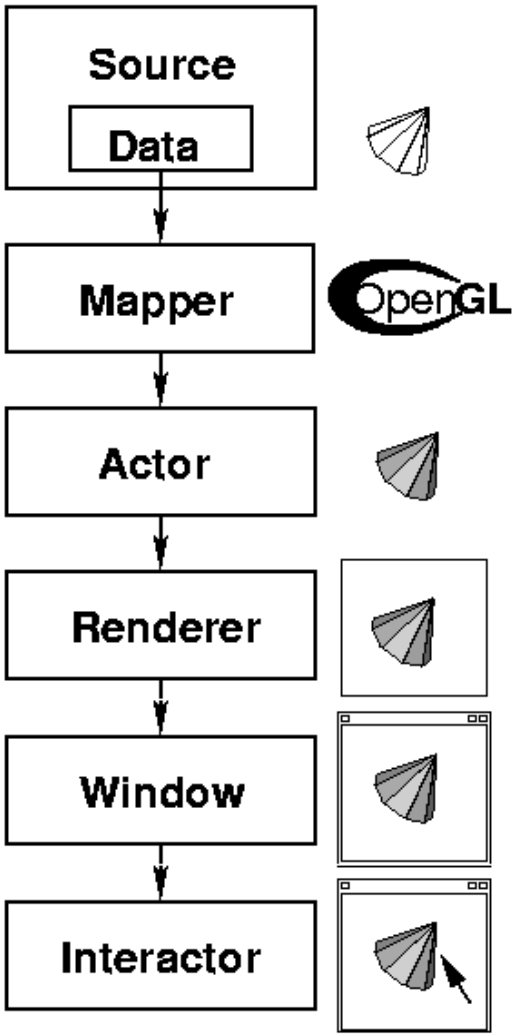
Modèle graphique

- Objectif
  - Transformer des données graphiques en images et les afficher à l'écran
- Regroupe les caractéristiques d'un système graphique 3D (infographie)
- Les principales classes VTK du modèle graphique:
  - Render Window
    - Renderer
      - Light
      - Camera
      - Actor
        - Property
        - Transform
        - Mapper

Modèle graphique



Cone.py Pipeline Diagram (type "python Cone.py" to run)



Either reads the data from a file or creates the data from scratch.

Moves the data from VTK into OpenGL.

For setting colors, surface properties, and the position of the object.

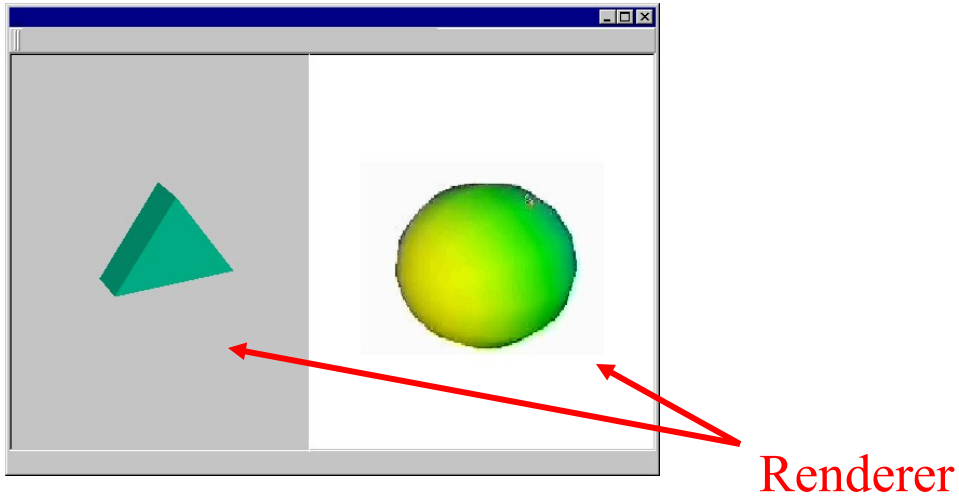
The rectangle of the computer screen that VTK draws into.

The window, including title bar and decorations.

Allows the mouse to be used to interact wth the data.

```
from vtkpython import *

cone = vtkConeSource()
cone.SetResolution(10)

coneMapper = vtkPolyDataMapper()
coneMapper.SetInput(cone.GetOutput())

coneActor = vtkActor()
coneActor.SetMapper(coneMapper)

ren = vtkRenderer()
ren.AddActor(coneActor)

renWin = vtkRenderWindow()
renWin.SetWindowName("Cone")
renWin.SetSize(300,300)
renWin.AddRenderer(ren)

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

Render Window



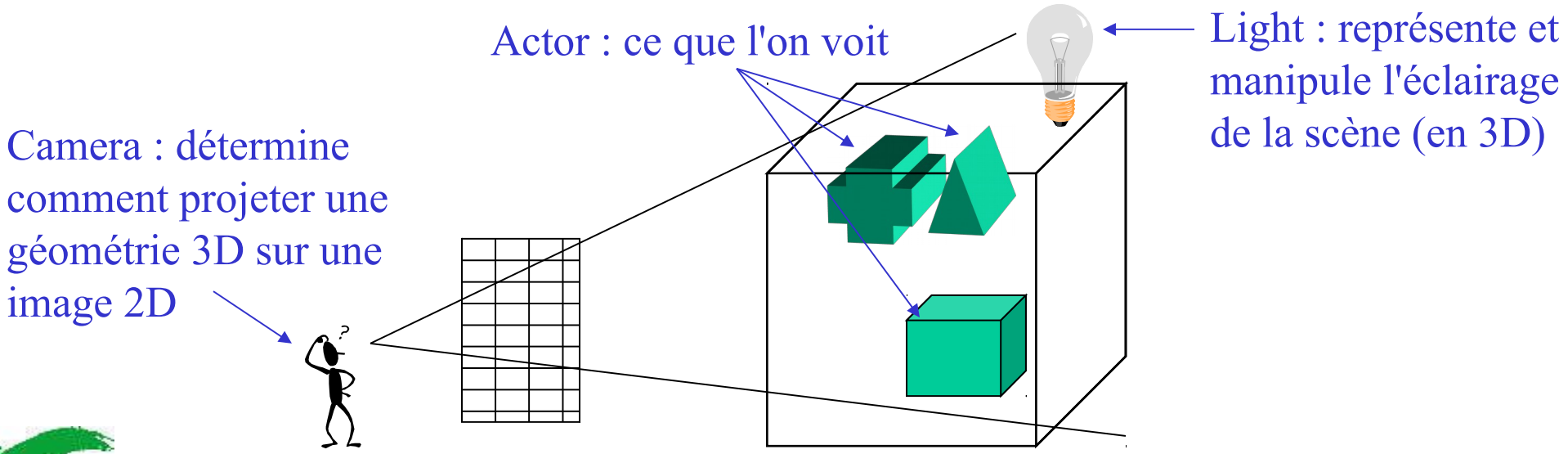
- Gère la(s) fenêtre(s) dans laquelle on va afficher les images ou les objets graphiques
- Fonctionnalité par défaut d'une fenêtre Linux
- Indépendante des dispositifs graphiques
- Gère l'ensemble de renderers contenus dans la fenêtre
  - Plusieurs renderers peuvent dessiner dans une même fenêtre (render window) pour créer une scène (image finale)

Classe vtkRenderWindow

- Méthodes
  - Ajouter / Supprimer un renderer
    - AddRenderer (vtkRenderer)
    - RemoveRenderer (vtkRenderer)
  - Configuration de l'écran
    - FullScreenOn ()
    - FullScreenOff ()
    - BordersOn ()
    - BordersOff ()

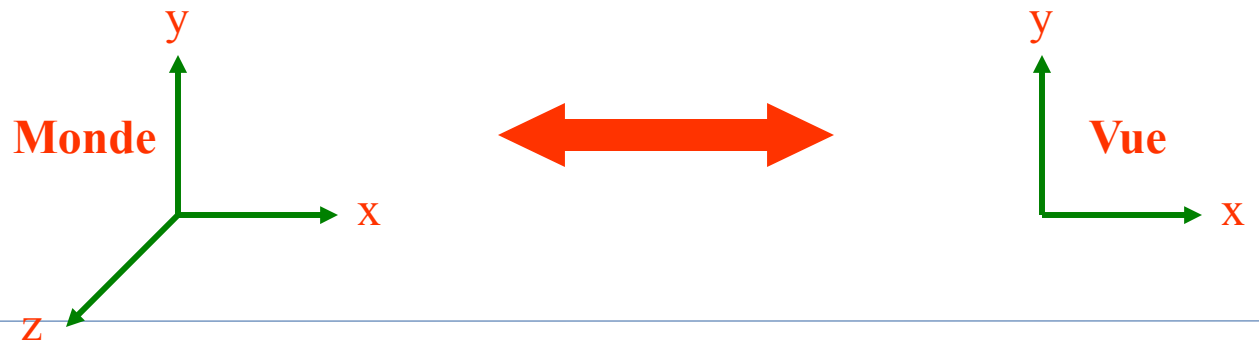
Renderer

- Coordonne la(s) source(s) de lumière, la camera et les actors pour générer l'image d'une scène
- Une scène comporte :
  - Au moins un actor, une camera et une source de lumière
  - Si les objets camera et light ne sont pas définis, ils sont créés automatiquement par le renderer



Classe vtkRenderer

- Méthodes
  - Ajouter/Supprimer des actors et des lights
    - AddActor (vtkActor) / RemoveActor (vtkActor)
    - AddLight (vtkLight) / RemoveLight (vtkLight)
  - Déterminer la camera à utiliser pour le rendu
    - SetActiveCamera (vtkCamera)
  - Créer l'image résultant du rendu
    - Render
  - Conversion de coordonnées
    - ViewToWorld (float, float, float) / void WorldToView (float, float, float)

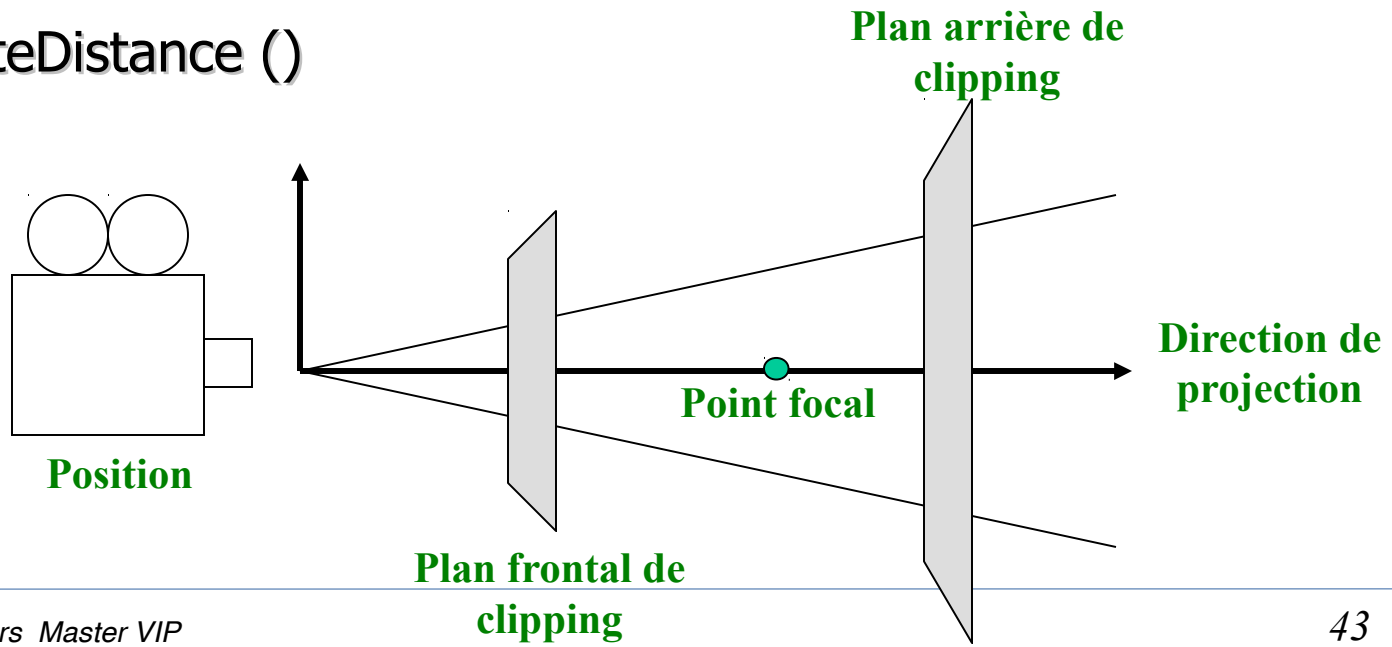


Classe vtkLight

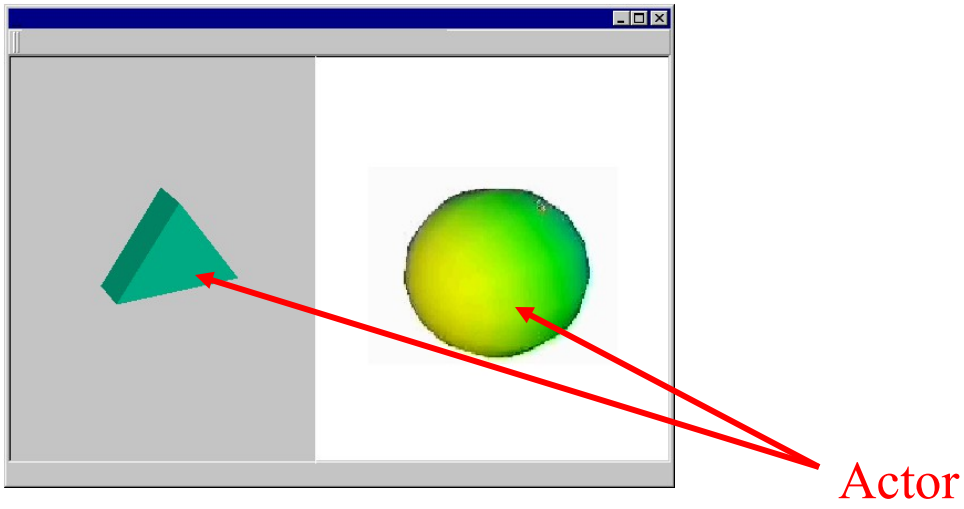
- Méthodes
  - Fixer la couleur de la lumière
    - SetColor (float ,float ,float) / GetColor ()
  - Déterminer la position de la source
    - SetPosition (float ,float ,float) / GetPosition ()
  - Définir l'intensité (de 0 à 1)
    - SetIntensity (float) / GetIntensity ()
  - Allumer / Éteindre
    - SwitchOn ()
    - SwitchOff ()

Classe vtkCamera

- Méthodes
  - Fixer la position de la camera en coordonnées monde
    - SetPosition (float ,float ,float) / GetPosition ()
  - Déterminer la position du point focal de la camera
    - SetFocalPoint (float ,float ,float) / GetFocalPoint ()
  - Calculer la distance entre la position de la camera et celle du point focal
    - ComputeDistance ()



Actor



- Objet dessiné par un renderer dans une scène
- Un actor ne représente pas directement sa géométrie ni son apparence
- Celles-ci sont définies en termes de :
  - Property
  - Mapper
  - Transform

Classe vtkActor

- Méthodes
  - Associer l'objet property qui détermine les propriétés d'apparence de l'actor
    - SetProperty (vtkProperty) / GetProperty ()
      - Un objet property est crée par défaut
      - Plusieurs actors peuvent partager le même objet property
  - Associer l'objet mapper qui détermine la géométrie de l'actor
    - SetMapper (vtkMapper) / GetMapper ()
  - Définir la matrice de transformations qui détermine l'échelle, la position et l'orientation de l'actor
    - SetUserMatrix (vtkMatrix4x4 *)
- Généralement
  - Il ne faut pas définir explicitement les propriétés ni les transformations. Des valeurs par défaut son déterminées lors de la création de l'actor

Property

- Détermine l'apparence de la surface d'un actor
- Classe VTK : vtkProperty
- Méthodes
  - Représentation de la géométrie de la surface
    - SetRepresentationToPoints ()
    - SetRepresentationToWireframe ()
    - SetRepresentationToSurface ()
  - Déterminer la couleur de la surface
    - SetColor (float ,float ,float)
  - Fixer l'opacité de la surface (0 transparent, 1.0 opaque)
    - SetOpacity (float)

Mapper

- Chargé du rendu (rendering)
- Lie le modèle de visualisation et le modèle graphique
  - Détermine la géométrie de l'actor

    Combinaison de points (sommets), lignes, polygones, etc
  - Définit la couleur des sommets

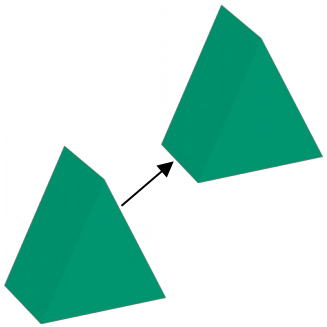
    Fait référence à une palette de couleurs
- Tout actor doit avoir un mapper associé pour pouvoir être affiché à l'écran
- Classes VTK : vtkMapper, vtkPolyDataMapper, vtkDataSetMapper

Classe vtkPolyDataMapper

- Méthodes
  - Spécifier les données d'entrée au mapper (données à afficher)
    - SetInput (vtkPolyData)
  - Associer la palette de couleurs
    - SetLookupTable (vtkLookupTable)
  - Créer une palette de couleurs par défaut
    - CreateDefaultLookupTable ()
  - Déterminer si le rendu est fait de manière immédiate ou pas
    - ImmediateModeRenderingOn ()
    - ImmediateModeRenderingOff ()

Transform

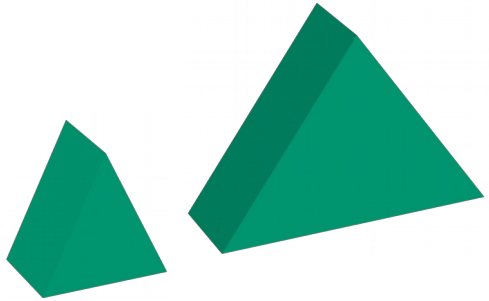
- Garde une pile de matrices de transformation
- Une seule matrice courante de transformation
- Fournit de méthodes pour effectuer les opérations de translation, changement d'échelle et rotation



Translation

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & tx \\ 0 & 1 & 0 & ty \\ 0 & 0 & 1 & tz \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

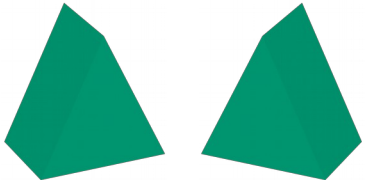
Translation (tx,ty,tz)



Changement d'échelle

$$Ts = \begin{bmatrix} sx & 0 & 0 & 0 \\ 0 & sy & 0 & 0 \\ 0 & 0 & sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Chgt d'échelle (sx,sy,sz)



Rotation

$$Tr = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) & 0 \\ 0 & \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation θ autour de x

Classe vtkTransform

- Méthodes
  - Créer d'une matrice identité
    - Identity ()
  - Créer une matrice de rotation et la concaténer avec la matrice courante de transformation
    - RotateX (float), RotateY (float), RotateZ (float)
  - Changer d'échelle la matrice courante de transformation
    - Scale (float ,float ,float)
  - Déplacer la matrice courante de transformation
    - Translate (float ,float ,float)
  - Transposer la matrice courante de transformation
    - Transpose ()
  - Invertir la matrice courante de transformation
    - Inverse ()
- A utiliser conjointement avec: vtkTransformFilter ou plus simplement: `actor->SetUserMatrix`

Vtk & C++

- VTK est une librairie de classes C++
- Pour programmer :
  - Inclure les fichier "headers" correspondants (.h)
  - Programmer en C++

```
#include <vtkRenderWindow.h>
#include <vtkRenderer.h>
#include <vtkRenderWindowInteractor.h>
#include <vtkDataSetReader.h>
#include <vtkDataSetMapper.h>
#include <vtkActor.h>
#include <vtkLookupTable.h>
#include <vtkContourFilter.h>
```

Exemple

```
vtkRenderer *renderer = vtkRenderer::New();
  vtkRenderWindow *renWin = vtkRenderWindow::New();
    renWin->AddRenderer(renderer);
  vtkRenderWindowInteractor *iren = vtkRenderWindowInteractor::New();
    iren->SetRenderWindow(renWin);

  vtkSphereSource *sphere = vtkSphereSource::New();
    sphere->SetPhiResolution(12); sphere->SetThetaResolution(12);

  vtkElevationFilter *colorIt = vtkElevationFilter::New();
    colorIt->SetInputConnection(sphere->GetOutputPort());
    colorIt->SetLowPoint(0,0,-1);
    colorIt->SetHighPoint(0,0,1);

  vtkDataSetMapper *mapper = vtkDataSetMapper::New();
    mapper->SetInputConnection(colorIt->GetOutputPort());

  vtkActor *actor = vtkActor::New();
    actor->SetMapper(mapper);

  renderer->AddActor(actor);
  renderer->SetBackground(1,1,1);
  renWin->SetSize(450,450);

  renWin->Render();
```

espace de rendu

Fenetre

Interaction

Données

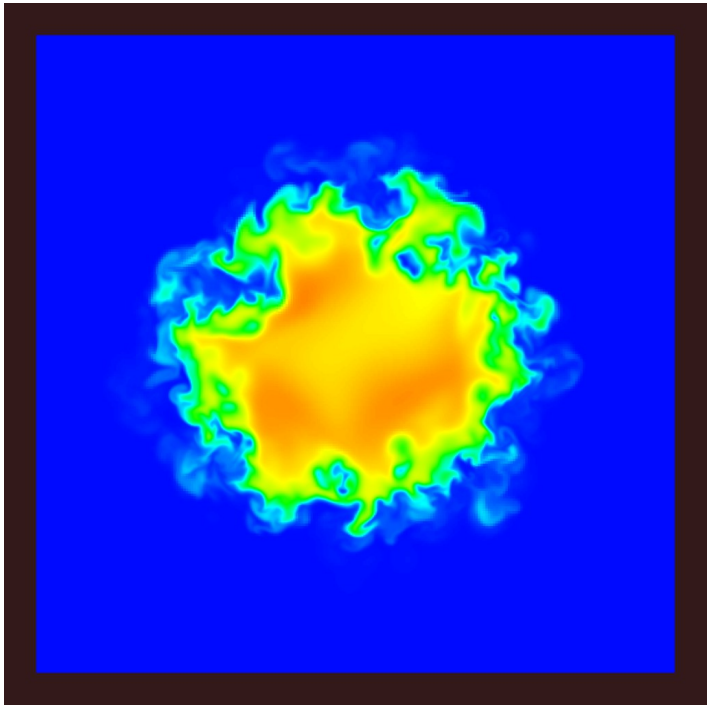
Modification

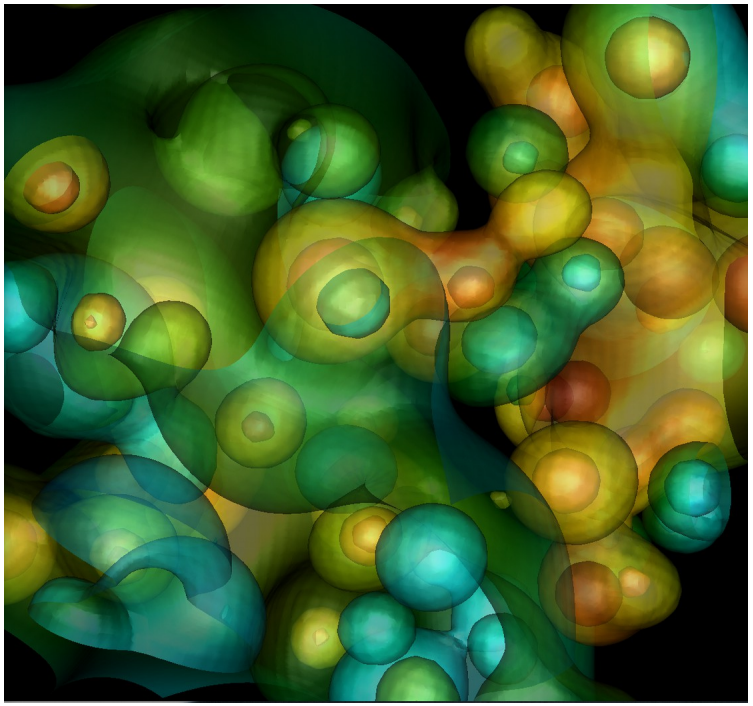
Passage aux données graphiques

Création de l'acteur

Lien avec le renderer

Modèle de Traitement





Modèle de traitement

- Objectif
  - Traitement d'images
- Implémentation particulière du pipeline de visualisation pour le traitement de datasets de type "Points structurés" (vtkStructuredPoints)
  - 2D

    Images

    Bitmaps
  - 3D

    Volumes (piles de datasets 2D)
- Classes spécialisées pour :
  - Lecture
  - Ecriture

- Affichage
- Traitement

Lecture d'images (2D et 3D)

- vtkImageReader
  - Lecture de n'importe quel type d'image
  - Il faut indiquer les dimensions et le type de l'image (pour sauter l'entête)
  - Les volumes sont lus comme une séquence d'images
  - Possibilité de lire un région d'intérêt (ROI/VOI)
- vtkBMPReader
  - Images BMP
- vtkTIFFReader
  - Images TIFF

Écriture d'images

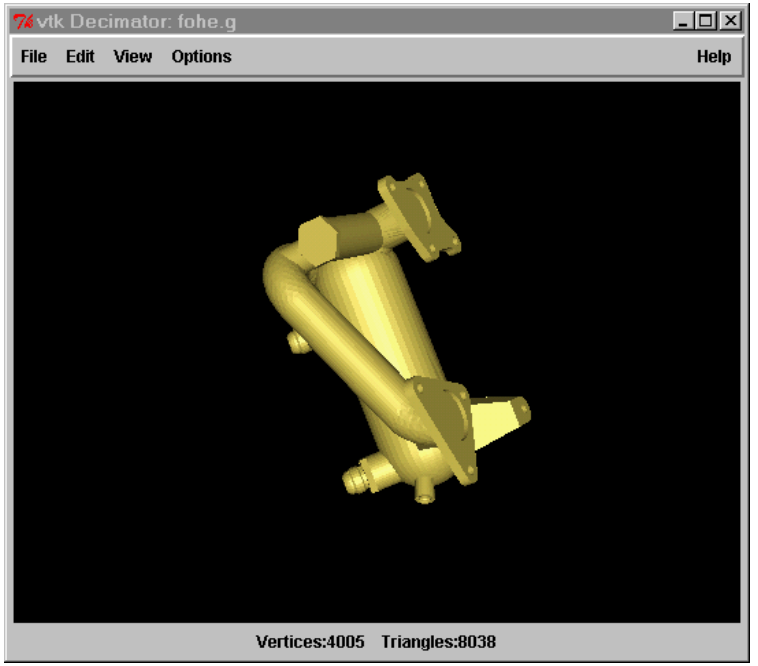
- vtkImageWriter
  - Ecriture de n'importe quel type d'image
  - Le type de l'image à sauvegarder est le même que celui de l'image en entrée
  - Les dimensions déterminent si les données doivent être sauvegardés en plusieurs fichiers
- vtkBMPWriter
- vtkTIFFWriter

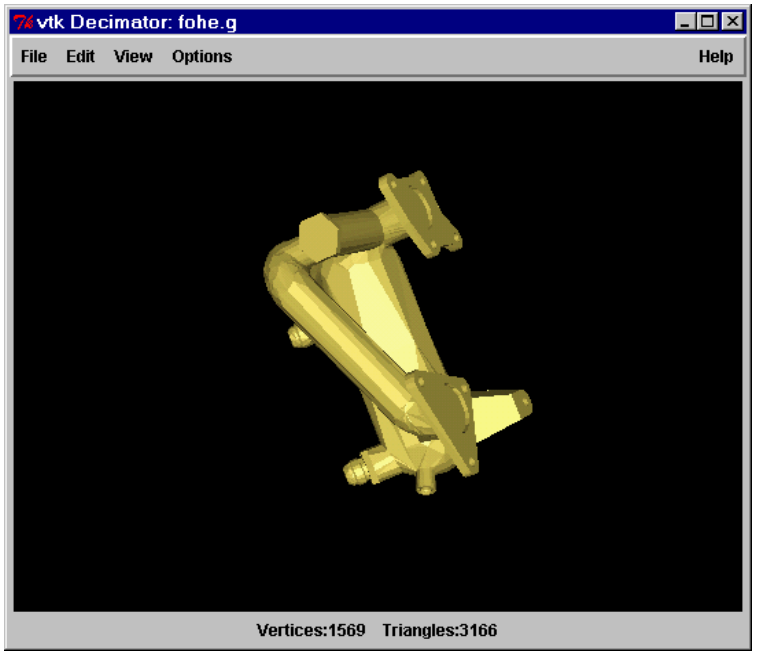
Affichage d'images

- Mapper
  - vtkImageMapper
  - vtkDataSetMapper
  - vtkPolyDataMapper
  - vtkVolumeMapper
- Actor
  - vtkActor
  - vtkActor2D
- Renderer
  - vtkRenderer

Traitement

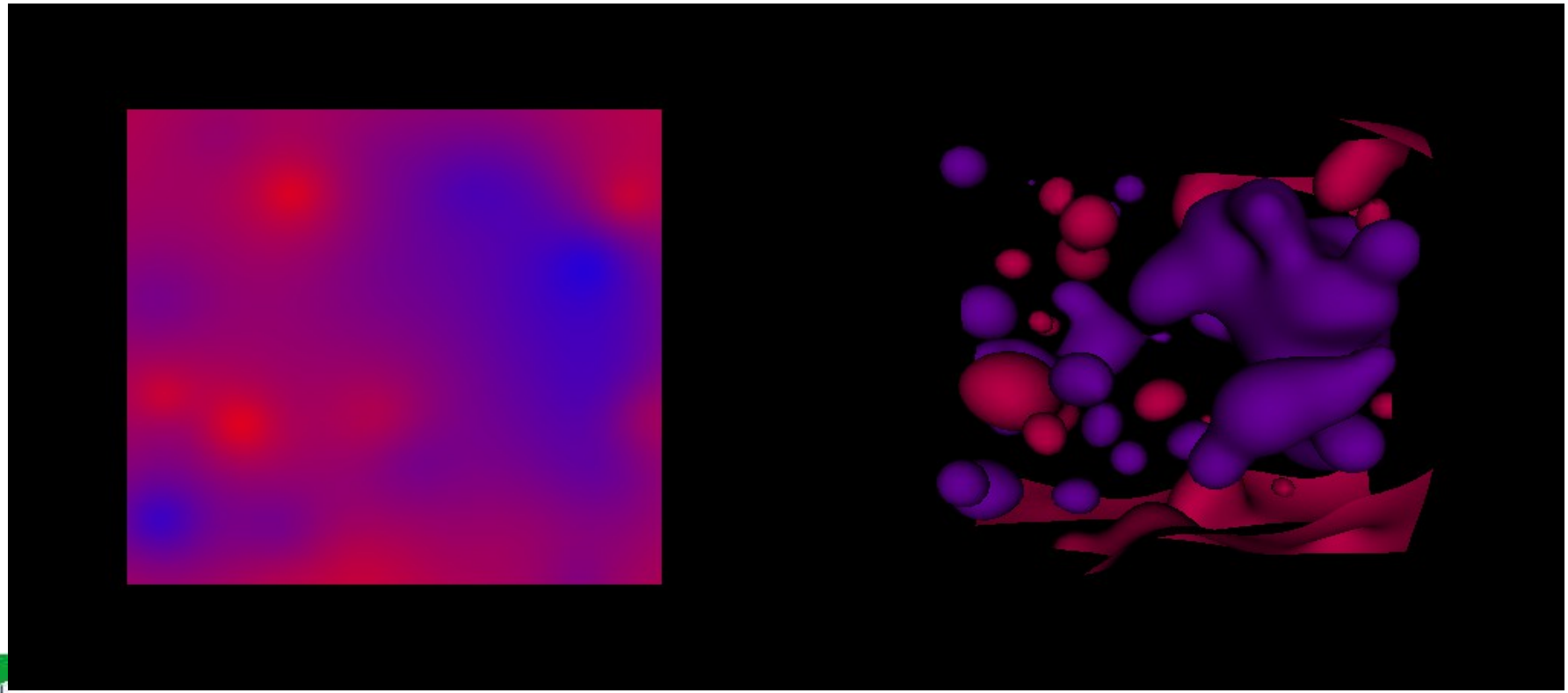
- vtkImageGradient
  - Gradient de l'image
- vtkImageMarchingCubes
  - Génération d'isosurfaces (rendu de surfaces)
- vtkImageMedian3D
  - Filtrage médian
- vtkImageResample
  - Re-échantillonnage d'une image avec interpolation linaire



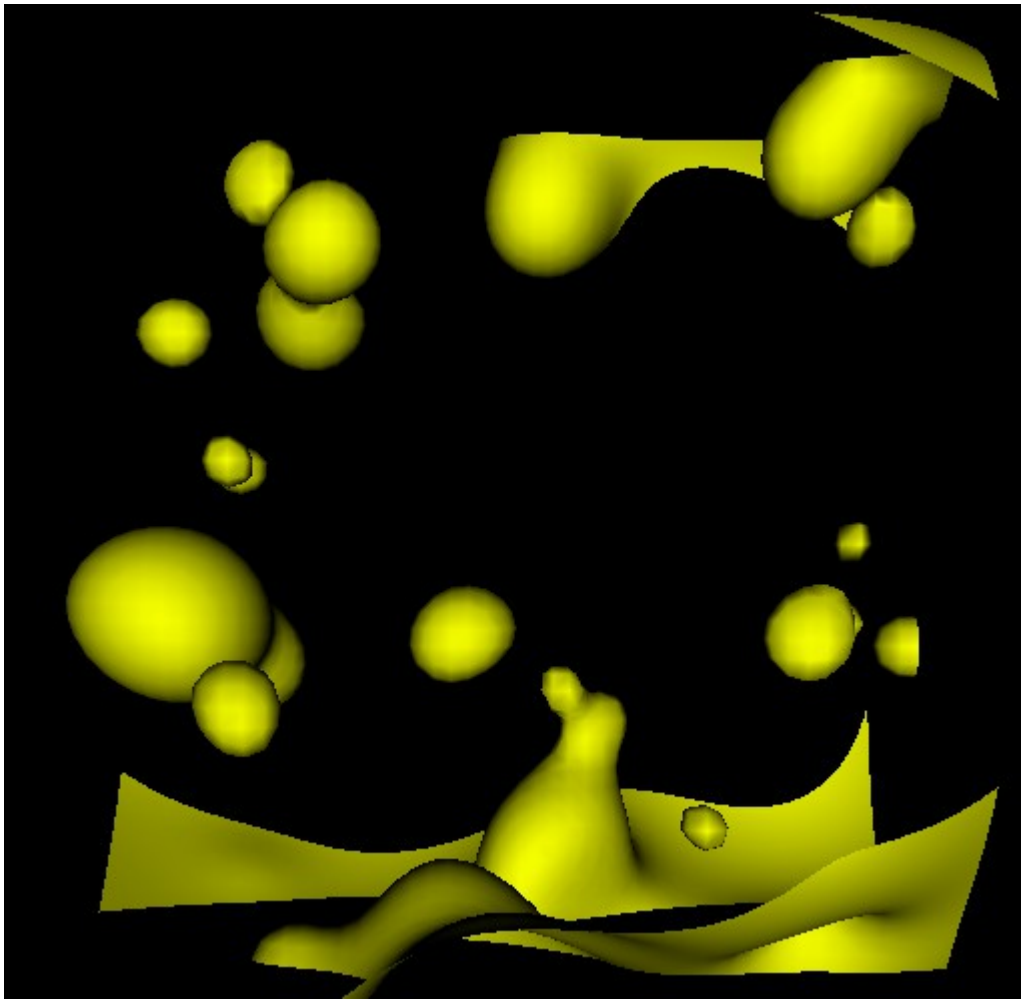


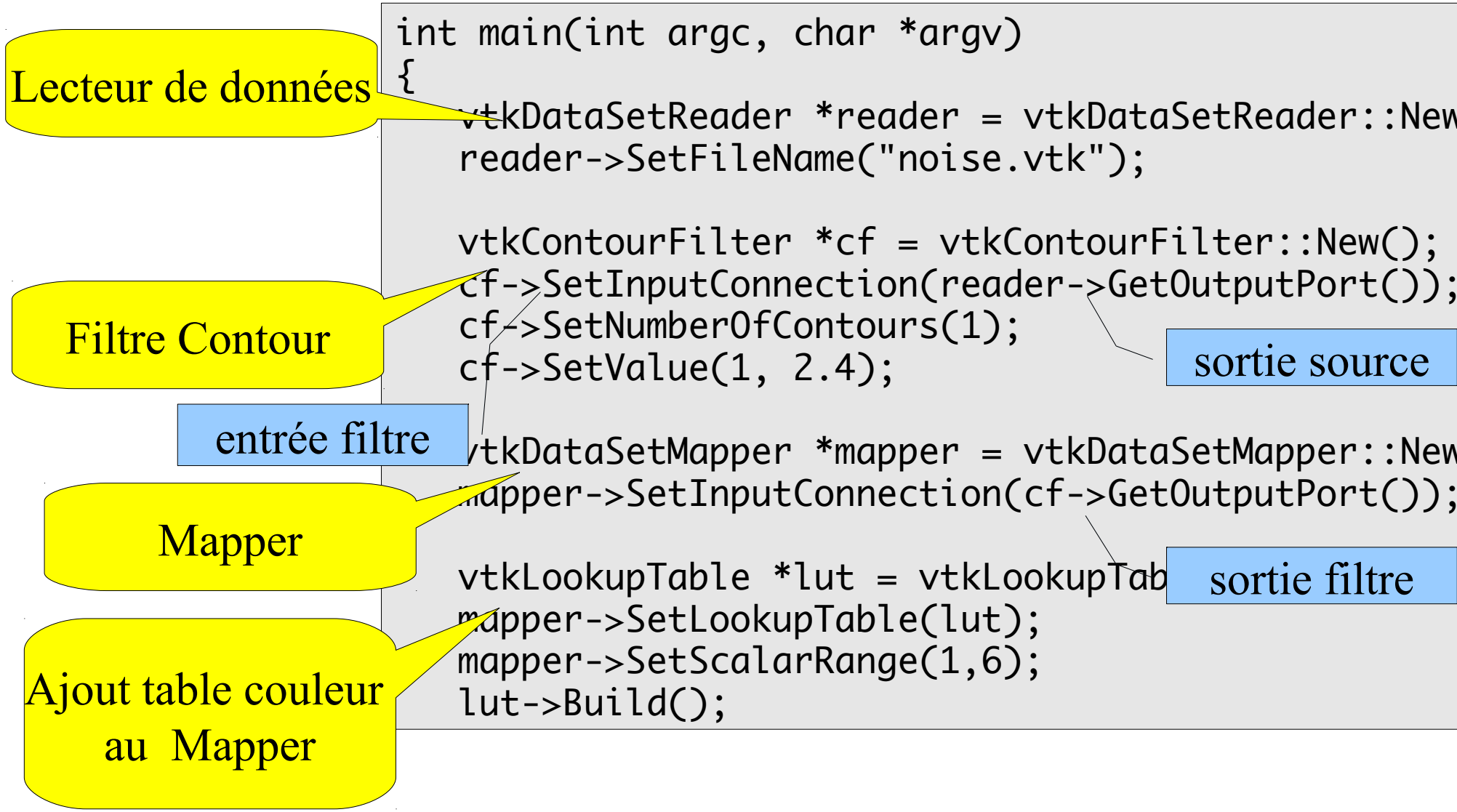
Traitement

- vtkImageThreshold
  - Seuillage
- vtkVolumeRayCastIsosurfaceFunction
  - Rendu surfacique d'un volume



Exemple d'une Application




Lecteur de données
Filtre Contour
entrée filtre
Mapper
Ajout table couleur au Mapper
sortie source
sortie filtre
int main(int argc, char *argv)
{
vtkDataSetReader *reader = vtkDataSetReader::New
reader->SetFileName("noise.vtk");
vtkContourFilter *cf = vtkContourFilter::New();
cf->SetInputConnection(reader->GetOutputPort());
cf->SetNumberOfContours(1);
cf->SetValue(1, 2.4);
vtkDataSetMapper *mapper = vtkDataSetMapper::New
mapper->SetInputConnection(cf->GetOutputPort());
vtkLookupTable *lut = vtkLookupTab
mapper->SetLookupTable(lut);
mapper->SetScalarRange(1,6);
lut->Build();

association d'un acteur au mapper

Création du renderer et ajout de l'acteur

Création fenêtre de rendu

Association d'un interacteur à la fenêtre

```
 vtkActor *actor = vtkActor::New();
   actor->SetMapper(mapper);

   vtkRenderer *ren = vtkRenderer::New();
   ren->AddActor(actor);

   vtkRenderWindow *renwin =
    vtkRenderWindow::New();
   renwin->SetSize(768, 768
   renwin->AddRenderer(ren)

   vtkRenderWindowInteractor *iren =
    vtkRenderWindowInteractor::New();
   iren->SetRenderWindow(renwin);
   renwin->Render();
   iren->Start();
}
```

Exemple 2

```
vtkConeSource *cone = vtkConeSource::New();
  cone->SetHeight( 3.0 );
  cone->SetRadius( 1.0 );
  cone->SetResolution( 10 );

   vtkSphereSource *sphere = vtkSphereSource::New();
   sphere->SetRadius(1.0);
   sphere->SetCenter(0.0,0.0,0.0);

 vtkPolyDataMapper *coneMapper = vtkPolyDataMapper::New();
  coneMapper->SetInputConnection( cone->GetOutputPort() );

  vtkPolyDataMapper *sphereMapper = vtkPolyDataMapper::New();
  sphereMapper->SetInputConnection(sphere->GetOutputPort());

 vtkActor *coneActor = vtkActor::New();
  coneActor->SetMapper( coneMapper );


  vtkActor *sphereActor = vtkActor::New();
  sphereActor->SetMapper( sphereMapper );
  sphereActor->GetProperty()->SetColor(0,1,0);
```

```
coneActor->RotateY(0);

while(true) {
      renwin->Render();
    // rotate the active camera by one degree

transform2->RotateZ(1);

 coneActor->SetUserMatrix(transform2->GetMatrix());
 //     ren->GetActiveCamera()->Azimuth( 1 );
 }
```