Visualisation avec VTK

EPITA - 13/06/2025 – Roman Fenioux

kitware

Visualisation?

Qu’est ce que VTK?

- **Une bibliothèque de visualisation**
  - Implémentée en utilisant les principes de programmation orienté objet
  - Codée en C++ (5M LOC), basé sur OpenGL
  - Automatiquement wrappé en Java, Tcl et Python ainsi que .Net (ActiViz)
  - Portable: Unix, Windows, MacOSX, mobile devices
  - Supporte 2D/3D visualisation, traitement d’image, rendu volumique, infoviz et geoviz
  - Environ 2500+ personnes sur les listes de diffusion
  - Utilisé par beaucoup d’industriels et académiques
  - Libre (BSD-like license)

kitware

Statistiques de VTK (openhub)

In a Nutshell, Visualization Toolkit...

... has had 73,044 commits made by 536 contributors
representing 5,650,305 lines of code

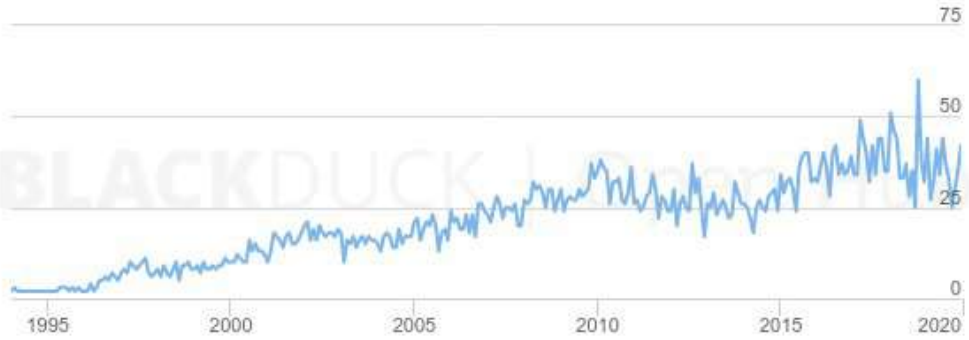
... is mostly written in C++
with an average number of source code comments

... has a well established, mature codebase
maintained by a very large development team
with stable Y-O-Y commits

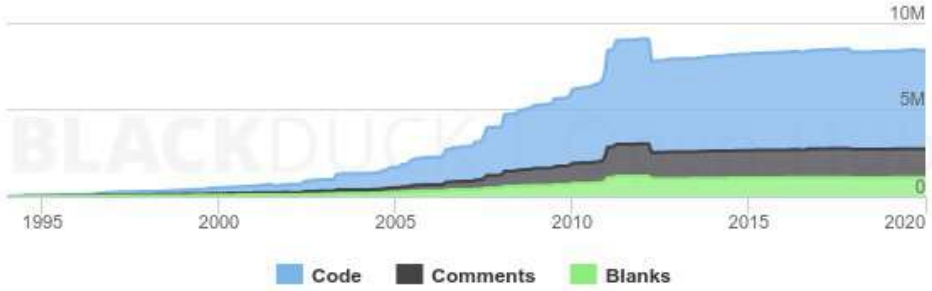
... took an estimated 1,727 years of effort (COCOMO model)
starting with its first commit in January, 1994
ending with its most recent commit about 2 months ago

Community

Contributors per Month



Lines of Code



30 Day Summary
Dec 15 2019 — Jan 14 2020

229 Commits
41 Contributors
including 5 new contributors

12 Month Summary
Jan 14 2019 — Jan 14 2020

3298 Commits
Up + 77 (2%) from previous 12 months

123 Contributors
Down -20 (13%) from previous 12 months

kitware

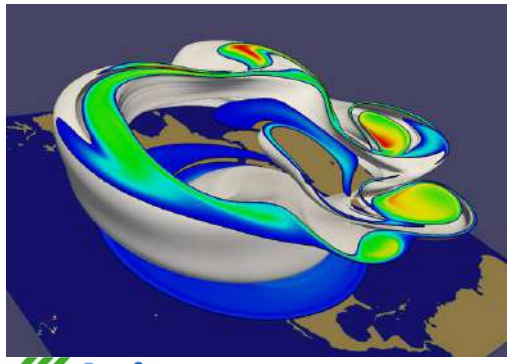
Philosophie du design de VTK

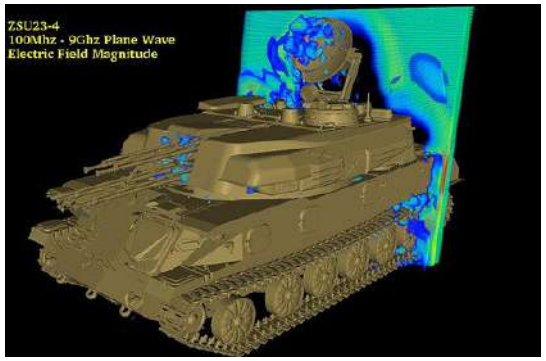
- **Architecture ouverte afin de créer des programmes (pas de UI)**
- **Architecture modulaire: chaque module fait une chose (correctement)**
  - Modules implémentés en utilisant de la programmation objets
  - Pipeline: les données passent par les modules
- **VTK a plusieurs thèmes**
  - Traitement de données
  - Interagir avec les données
  - Support pour les données massives

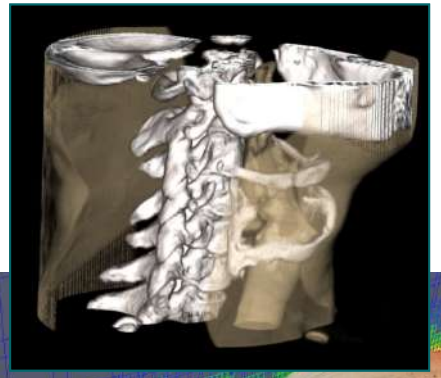
kitware

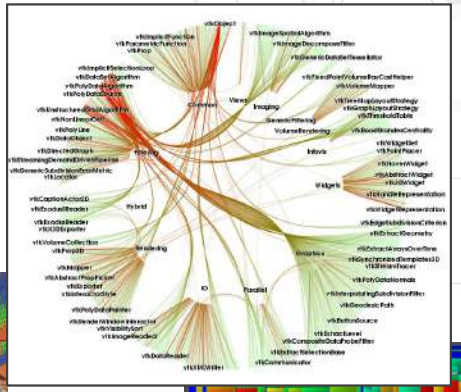
Que puis-je faire avec VTK?

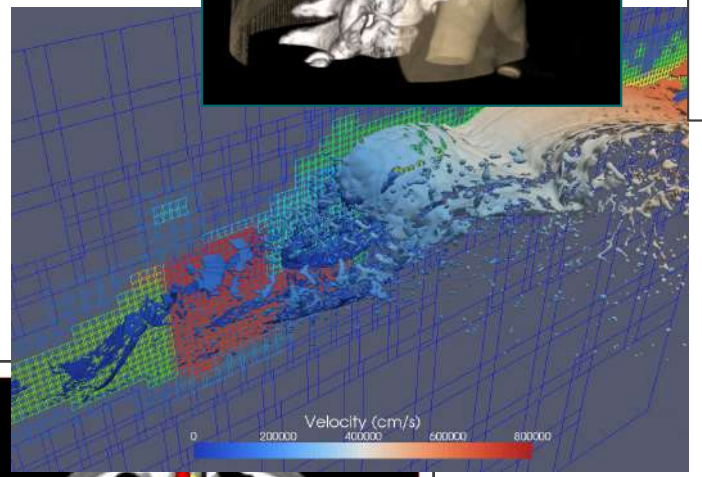
- **Visualisation Scientifique**
  - 0D: Visualisation d'information
  - 2D: Graphs/
  - 3D/4D: traitement de données et rendu
  - Traitement d'image
  - Rendu volumique

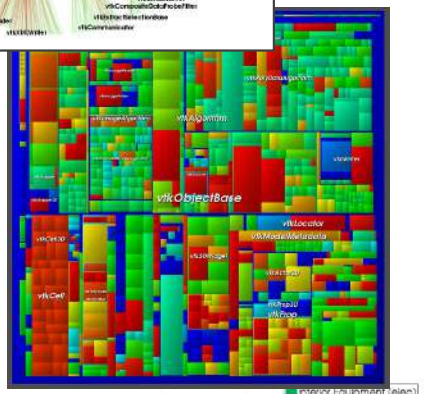


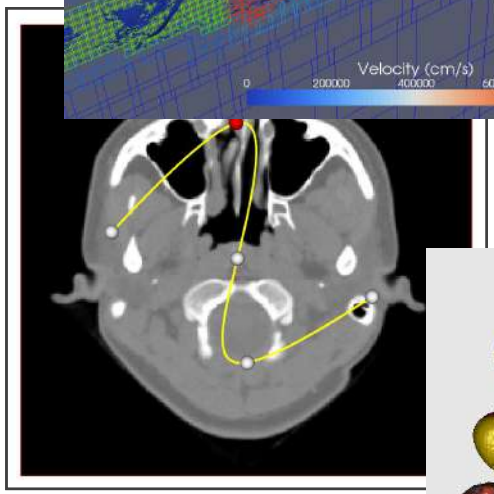


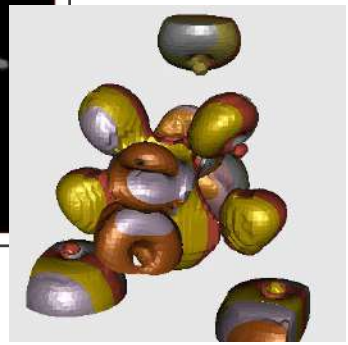


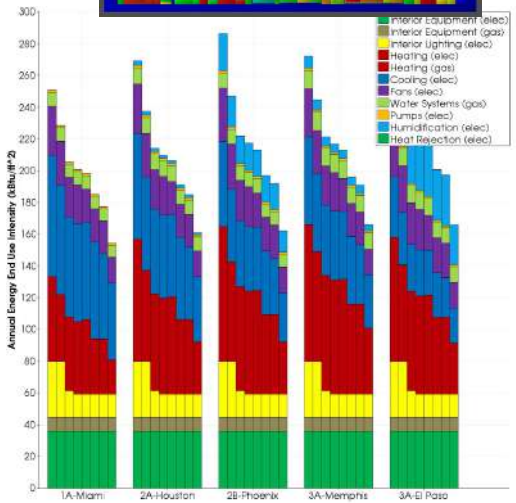






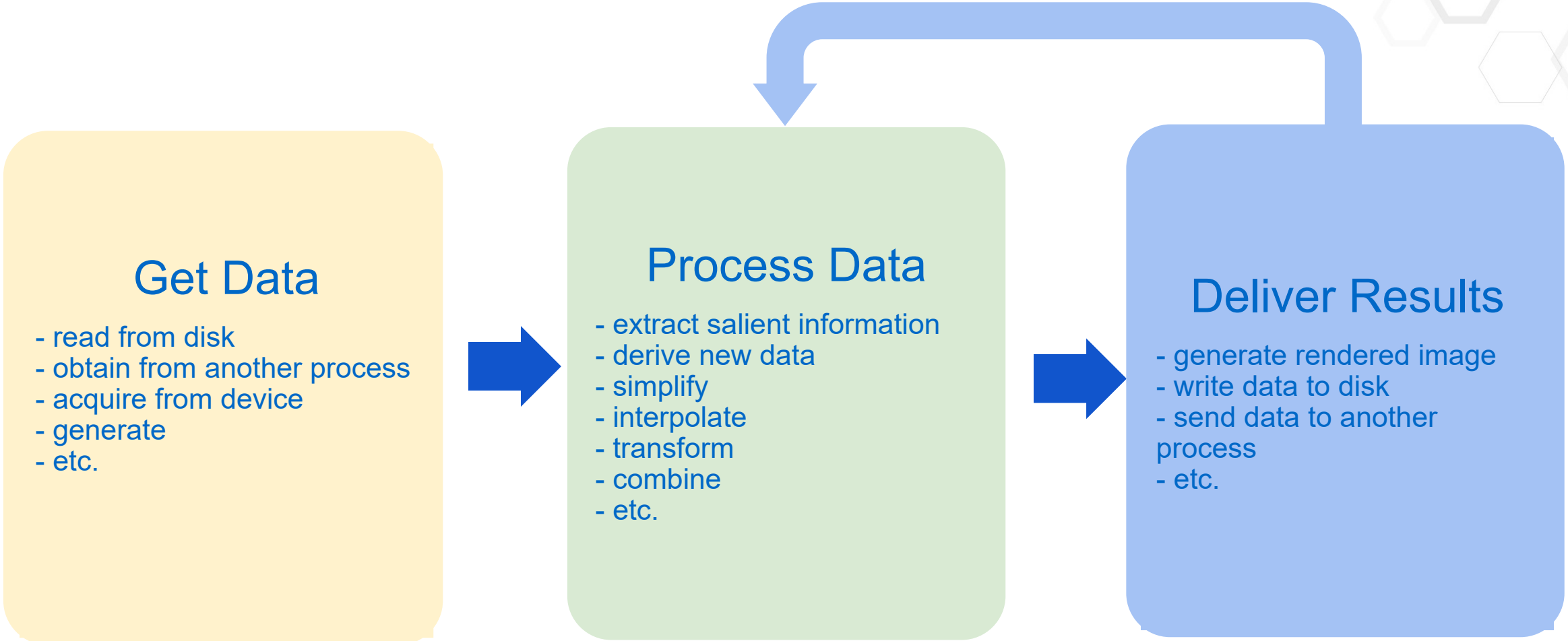






kitware

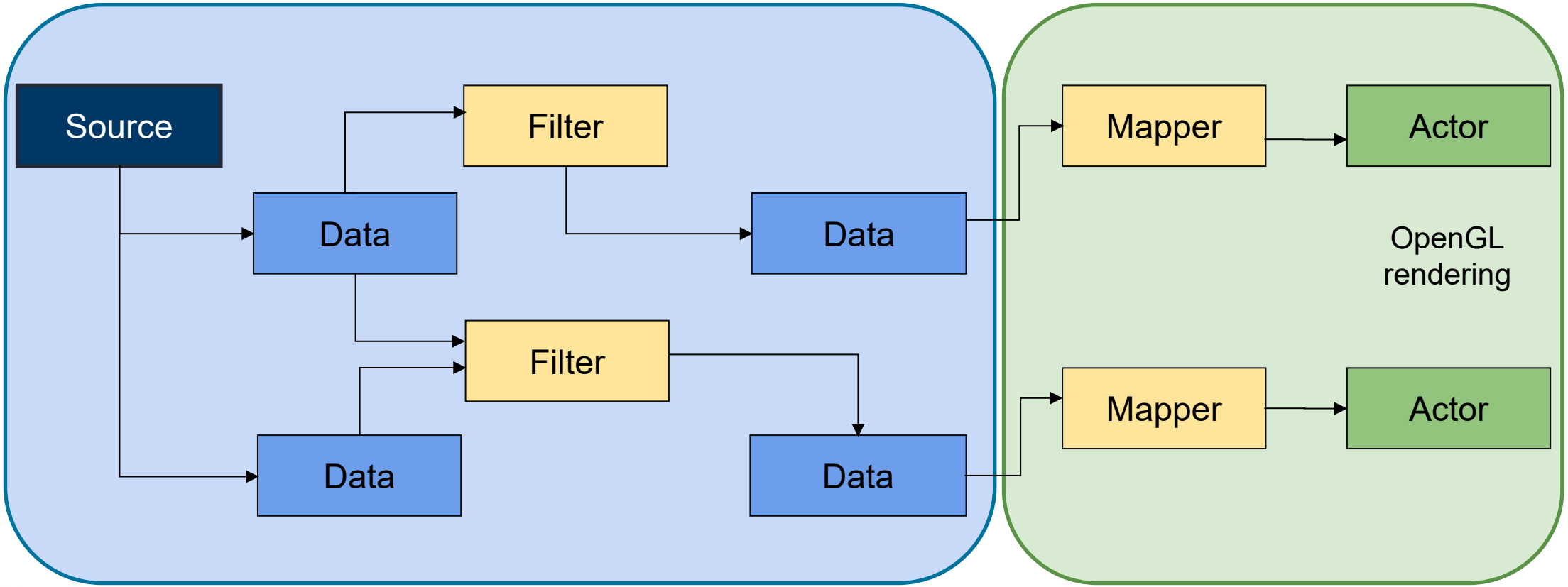
Visualisation



kitware

Architecture du pipeline de VTK

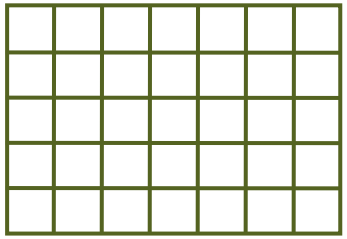
Une séquence d'algorithme qui opèrent sur des données.



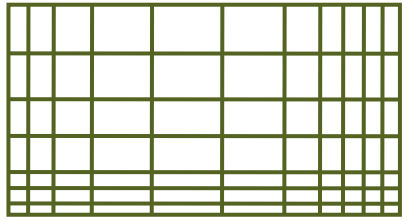
kitware

Type de données dans VTK

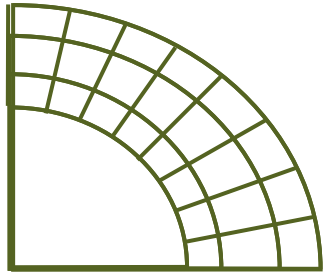
vtkImageData



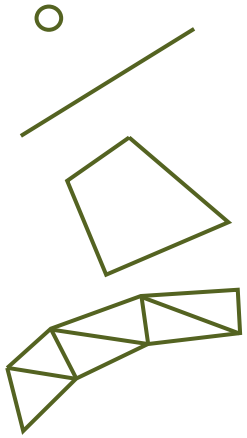
vtkRectilinearGrid



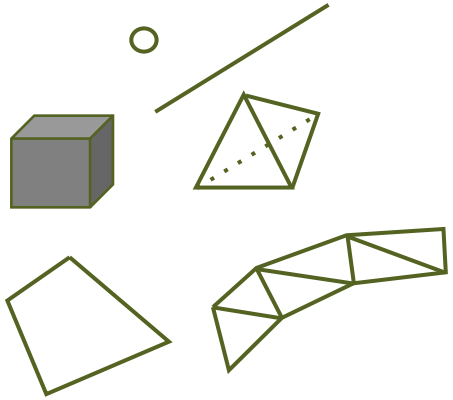
vtkStructuredGrid



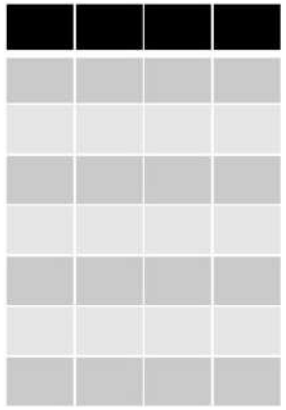
vtkPolyData



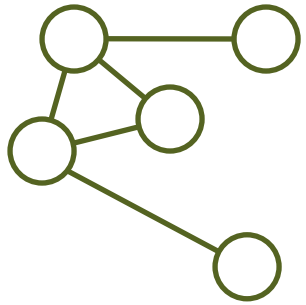
vtkUnstructuredGrid



vtkTable



vtkGraph



kitware

VTK Object Modeling Diagram

vtkObject

inherits from

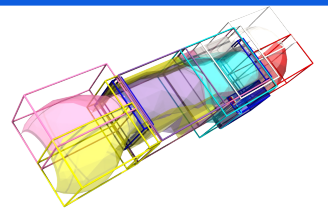
vtkDataObject

Name	Capitol	Population
California	Sacramento	
Texas	Austin	
…	…	…

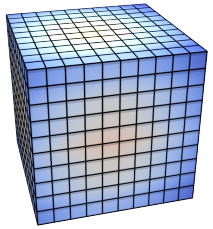
vtkTable

vtkDataSet

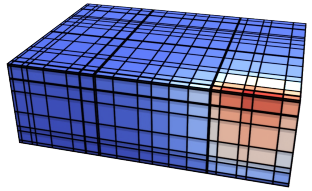
vtkCompositeDataObject



vtkImageData

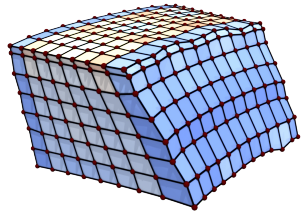


vtkRectilinearGrid

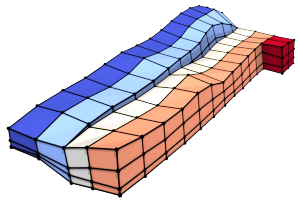


vtkPointSet

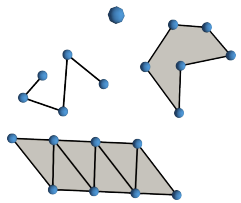
vtkStructuredGrid



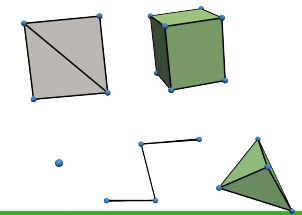
vtkExplicitStructuredGrid



vtkPolyData



vtkUnstructuredGrid



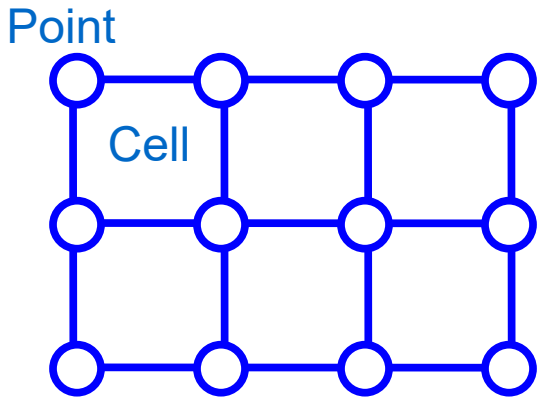
kitware

Type de données dans VTK

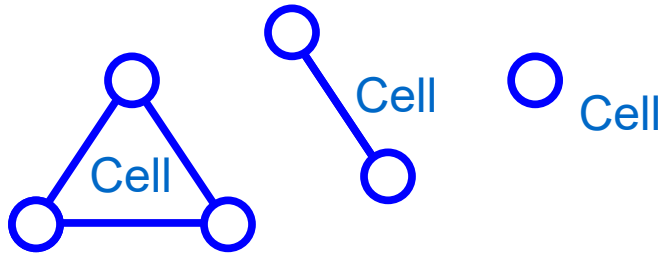
- **vtkDataObject: représente un « blob »de données**
  - Superclasses de tous les types de VTK
  - Aucune structure géométrique ou topologique
- **vtkDataSet: représente une donnée avec une géométrie et topologie**
  - Contient une géométrie (points) et une topologies (cellules)
  - Possède des attributs associées au points et aux cellules
- **vtkDataArrays: représente les attributs**
  - Ints, floats, chars, strings, variants, etc…

Structure Spatiale

- **Points (nœuds)**
  - Tableau de coordonnées x, y, z
- **Cellules**
  - Types et liste de points



Structured Data

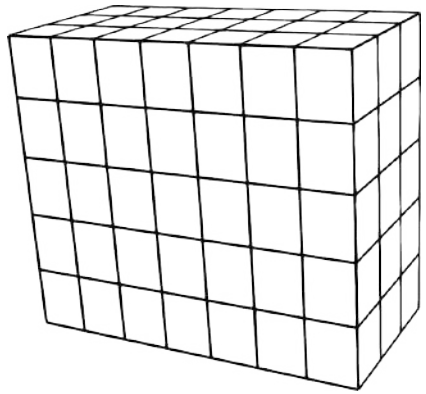


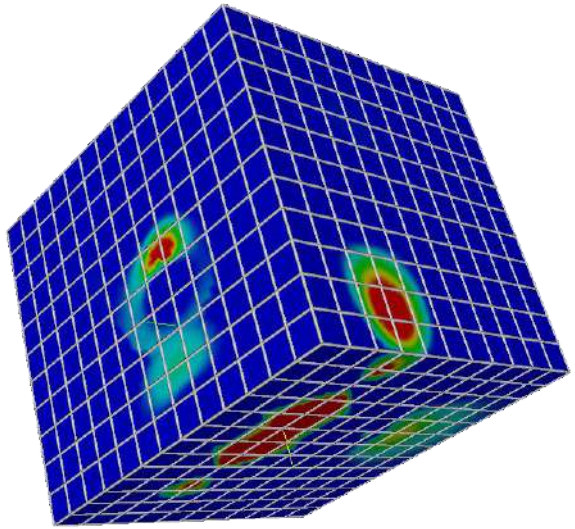
Unstructured Data

kitware

vtkImageData

- **Uniforme et rectilinéaire**
  - La géométrie et topologie sont implicite
  - Définie par un origin, dims et spacing

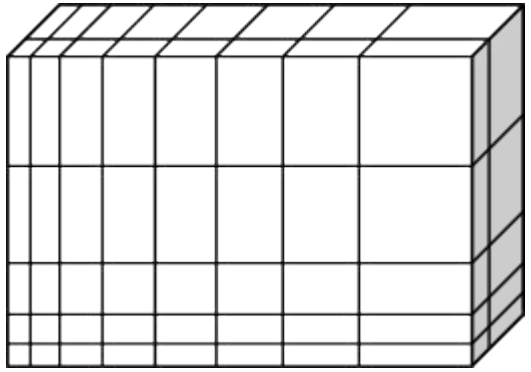


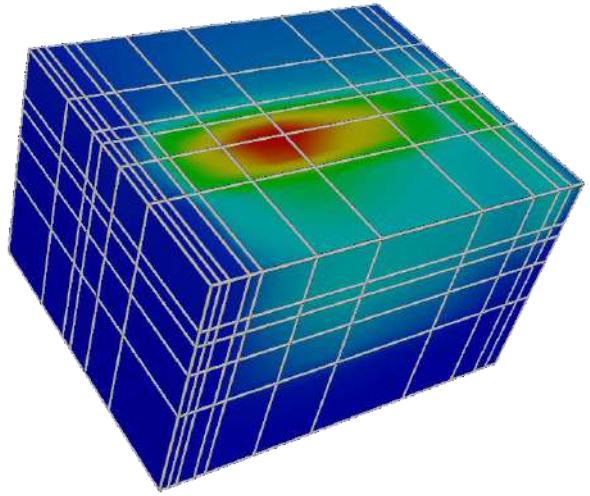


kitware

vtkRectilinearGrid

- **Non-uniforme mais rectilineaire**
  - La géométrie et topologie sont implicite
  - Définie par un origin, dims et spacing dans toutes les dimensions

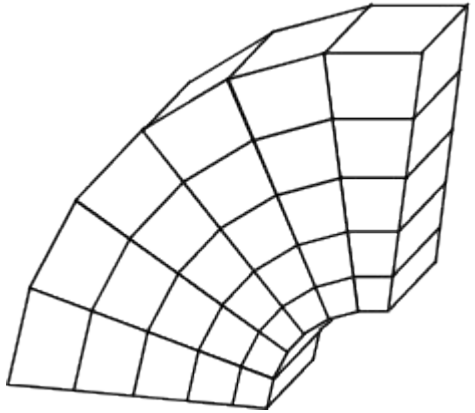


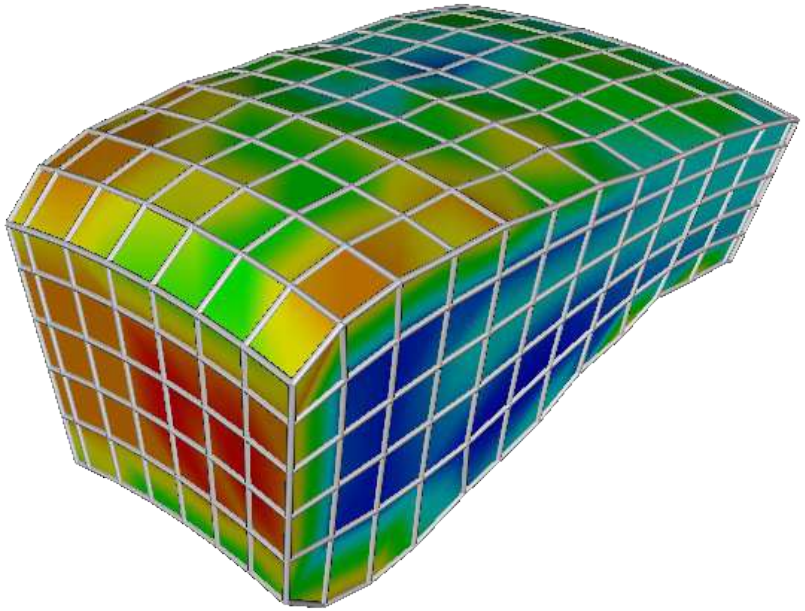


kitware

vtkStructuredGrid

- **Non-uniforme et curvilinéaire**
  - La géométrie est explicite et définie par des points
  - La topologie est implicite: les cellules peuvent être accéder par (i,j,k)

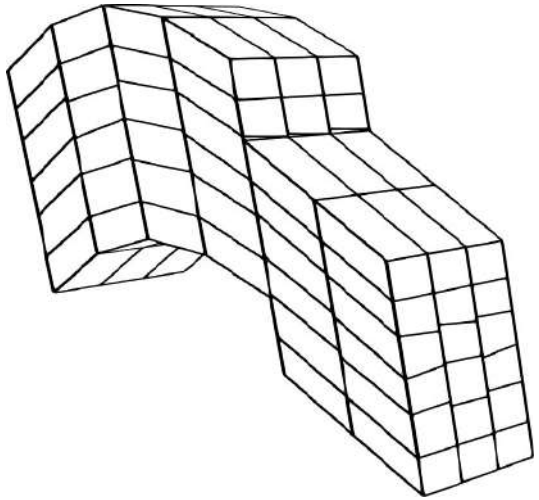


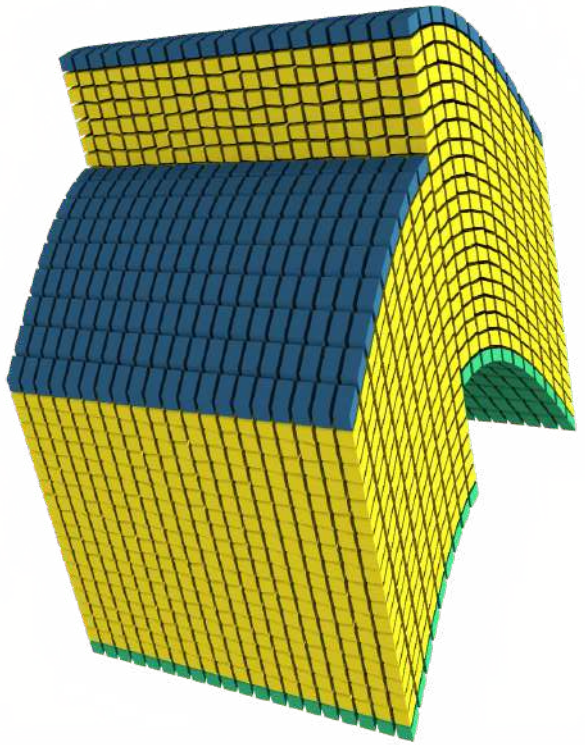


kitware

vtkExplicitStructureGrid

- **Non-uniforme et explicite**
  - Tableau d’hexaèdres
  - Les voisins ne partages pas nécessairement une face

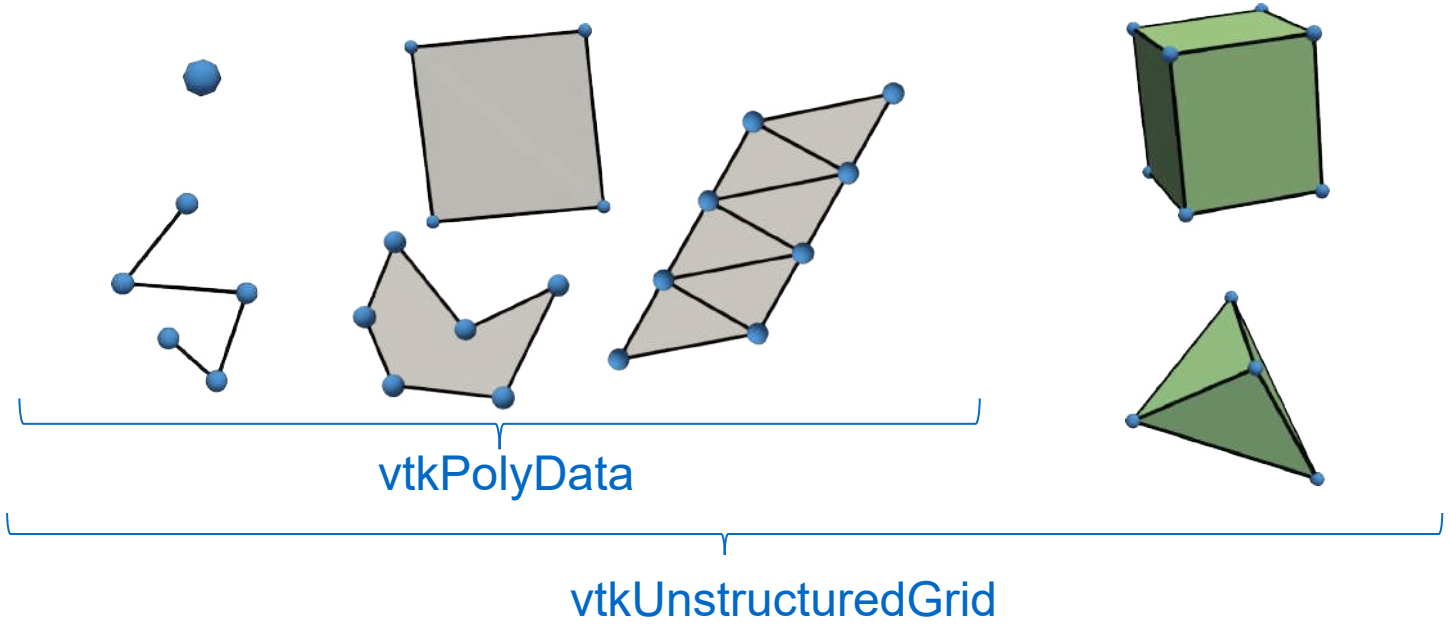




kitware

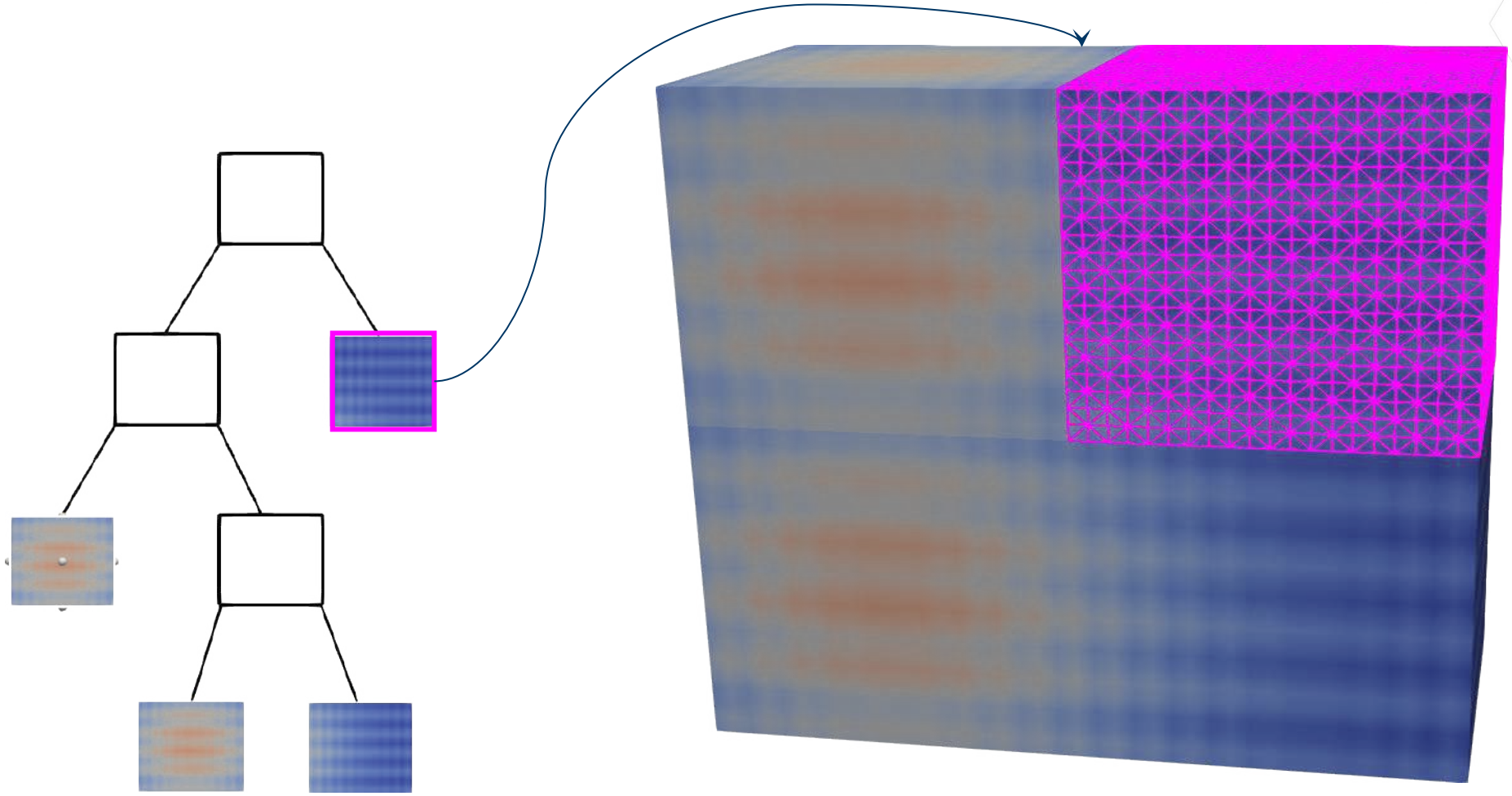
vtkPolyData et vtkUnstructuredGrid

- **Données non structurées**
  - La géométrie est explicite et définie par des points
  - La topologie est explicite et définie par des cellules



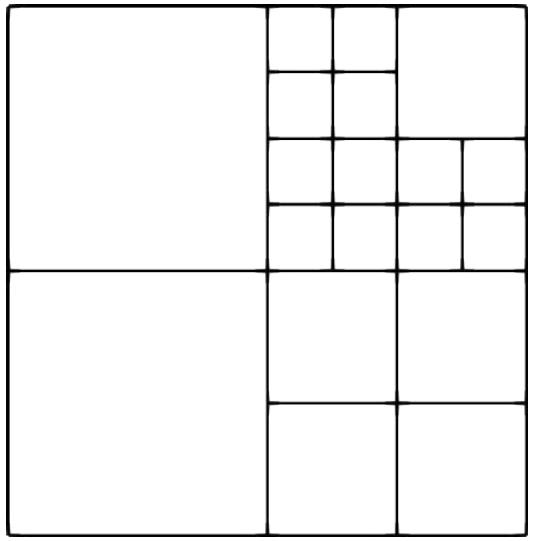
kitware

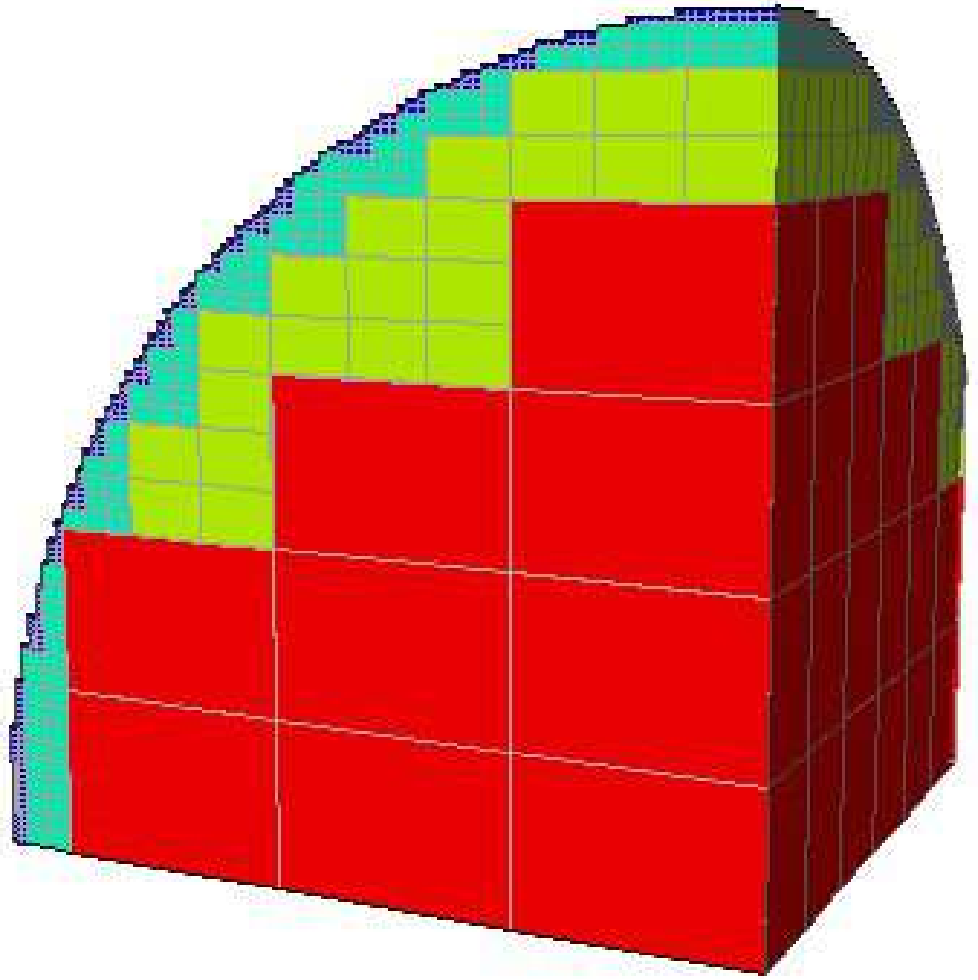
vtkMultiBlock



kitware

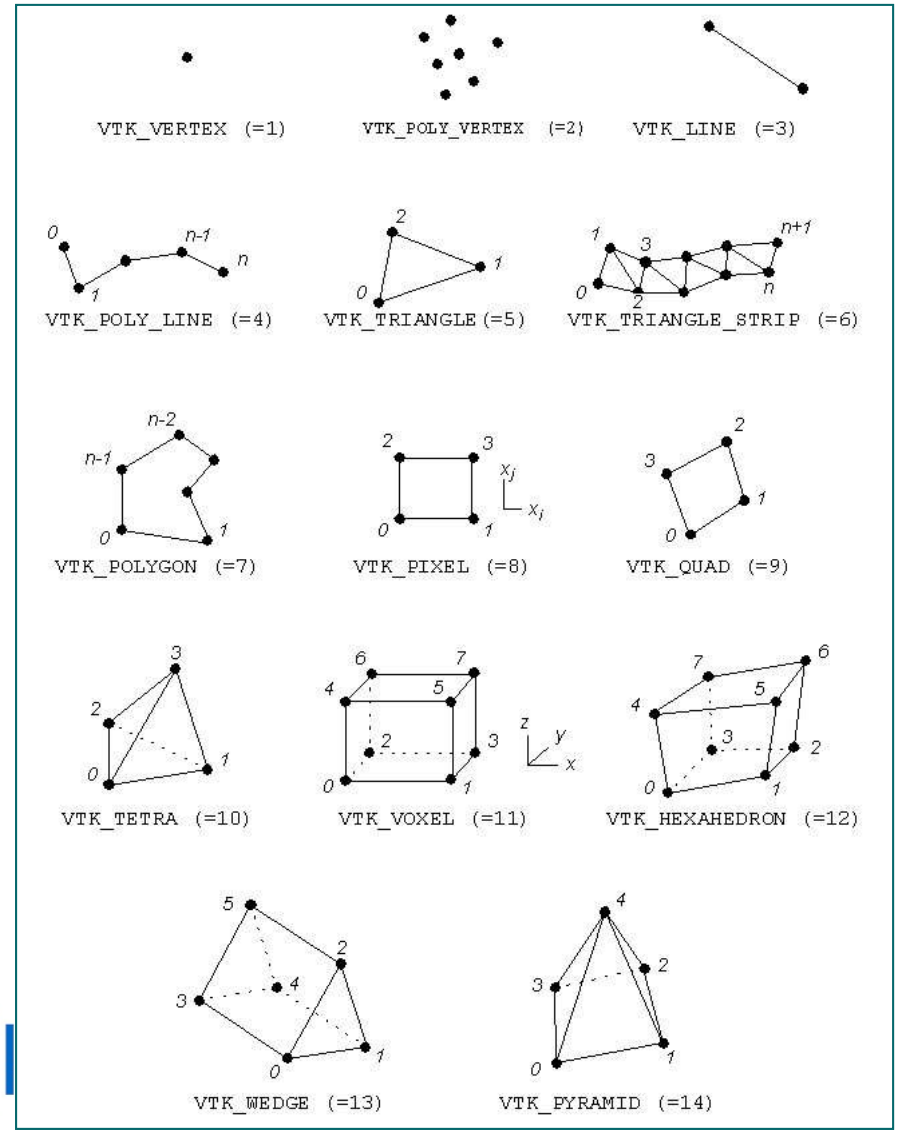
vtkHyperTreeGrid





kitware

Types de cellules



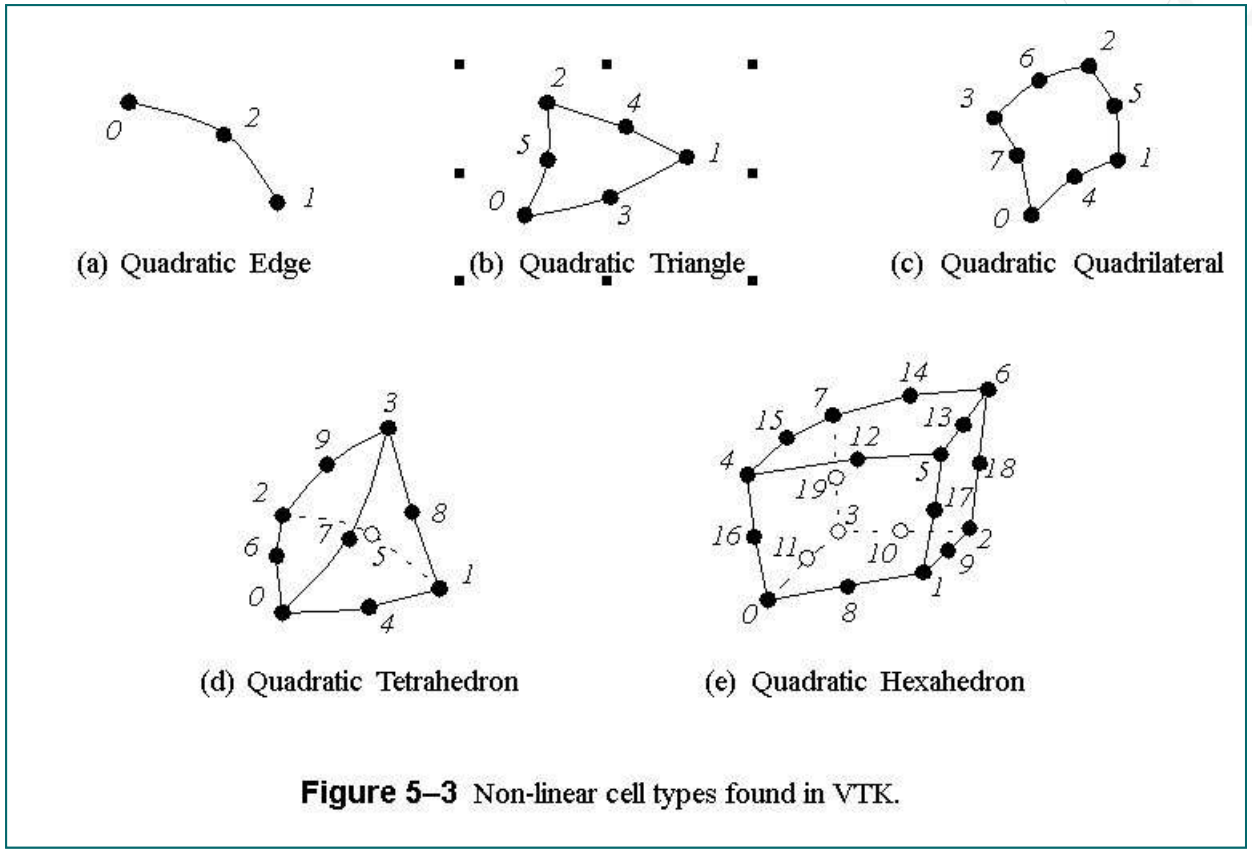
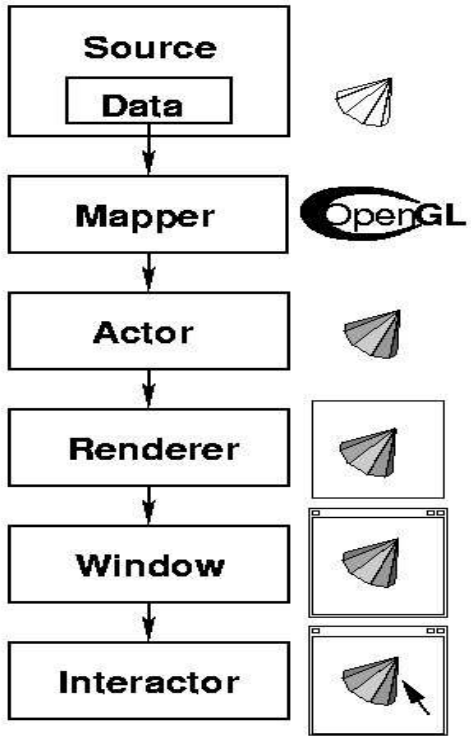
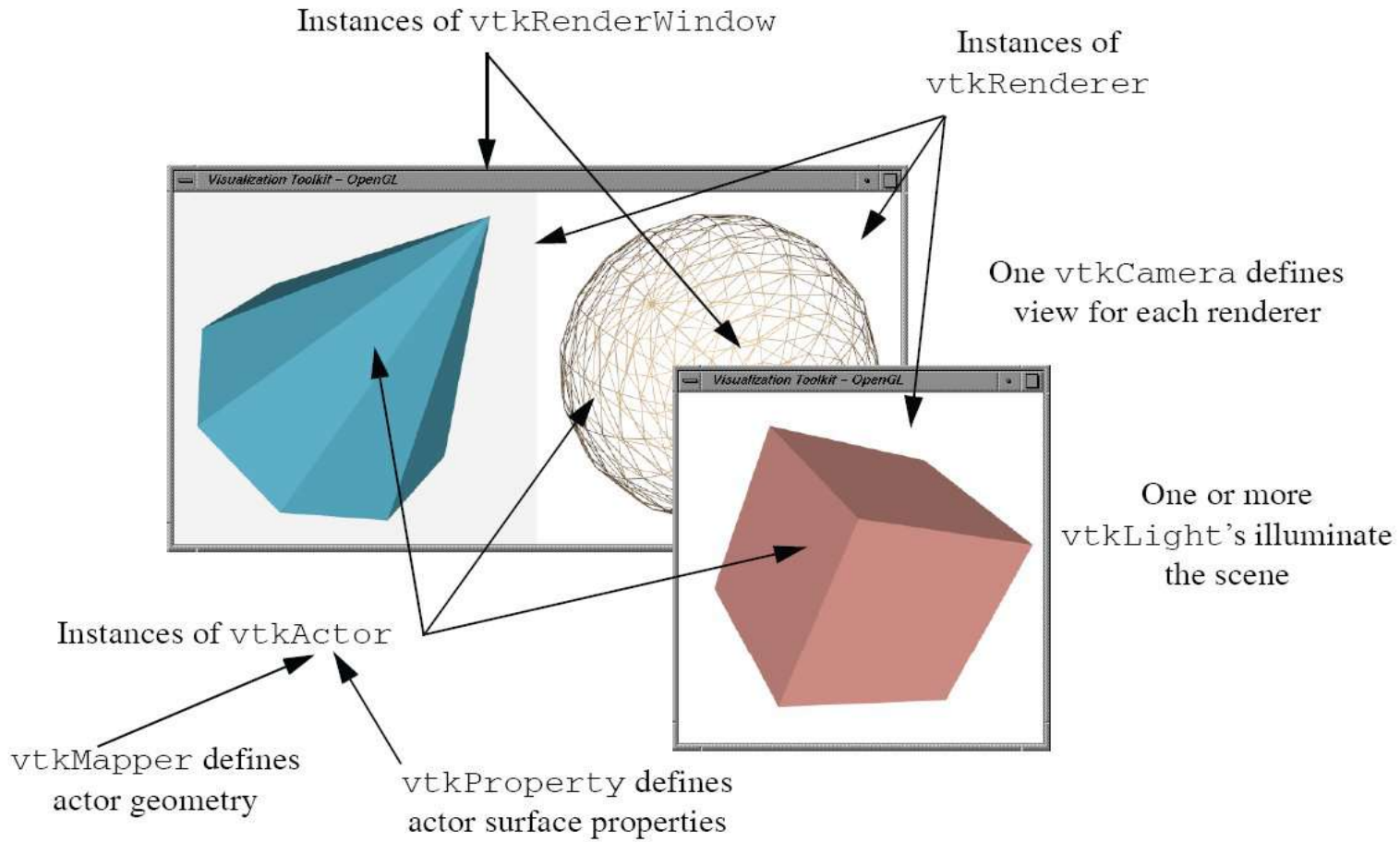


Figure 5–3 Non-linear cell types found in VTK.

Pipeline de Rendu

Rendu graphique avec VTK

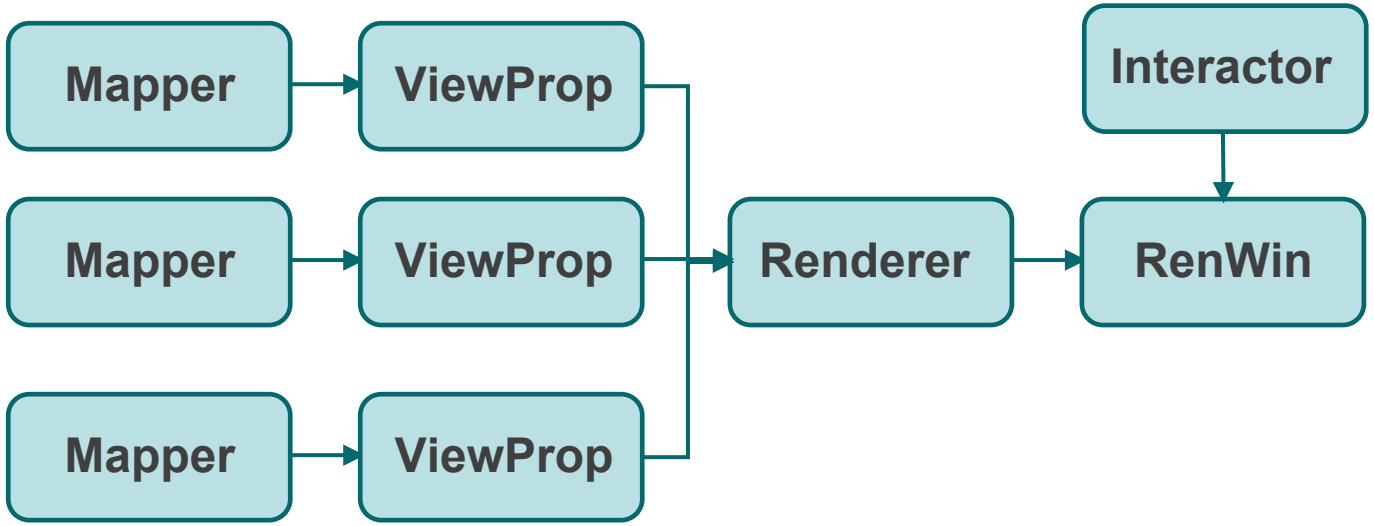




kitware

Du traitement ou pipeline de rendu

- Mapper (dernier vtkAlgorithm du pipeline de traitement)
- ViewProp (vtkActor, vtkImageActor, vtkVolume)
- Renderer (comment afficher)
- RenderWindow (où afficher)
- RenderWindowInteractor (ajout d'interaction)

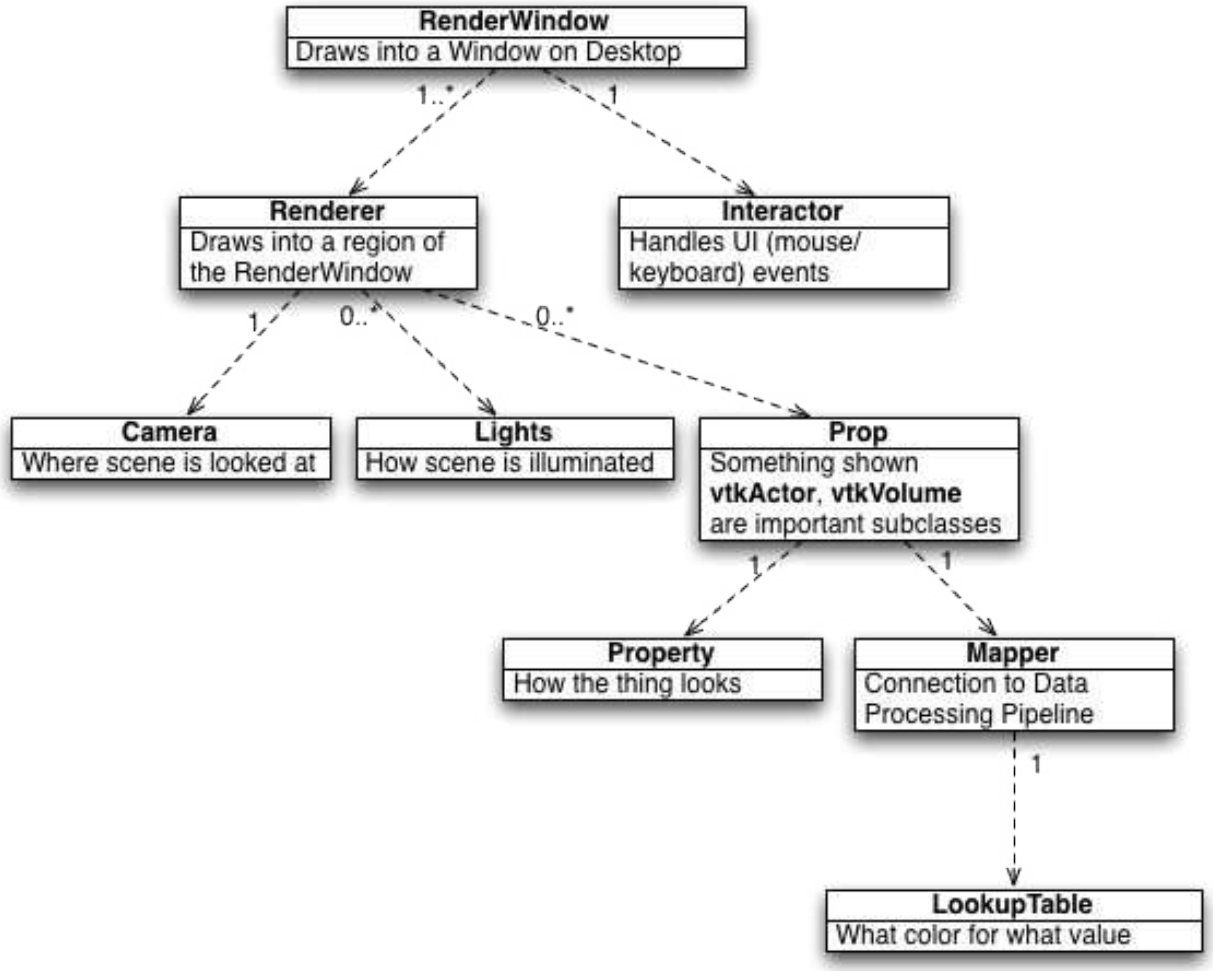


kitware

Scene VTK

- vtkRenderWindow – contient l'image finale
- vtkRenderer – dessine dans une vtkRenderWindow
- vtkActor – combine les propriétés et la géométrie
- vtkProp, vtkProp3D sont des superclasses
- vtkProperty – contient des propriétés
- vtkLights - illumine les acteurs
- vtkCamera – effectue le rendu de la scène
- vtkMapper - représente la géométrie
- vtkPolyDataMapper, vtkDataSetMapper sont des sous classes
- vtkTransform – positionne les acteurs

Moteur de rendu de VTK



kitware

vtkRenderWindow

- **Dessine dans une fenêtre:**
  - AddRenderer() – Ajoute un vtkRender
  - SetSize() – Définie la taille de la fenêtre
  - SetPosition() – Définie la position de la fenêtre
  - StereoType, StereoRenderOn/Off – contrôle la stéréo
  - DesiredUpdateRate – utiliser pour contrôler le niveau de détail
  - DoubleBuffer –double buffering on/off
  - Render() – mets à jour le pipeline de rendu

vtkRenderer

- Dessine dans une region de la vtkRenderWindow
  - SetViewport() – specifie ou rendre dans la render window
  - AddViewProp() – ajoute un objet qui doit être rendu
  - SetBackground() – specifie la couleur du fond
  - SetActiveCamera() – specifie la camera active
  - ResetCamera() – remet la camera de telle sorte que tous les acteurs soient visibles

VTK Camera

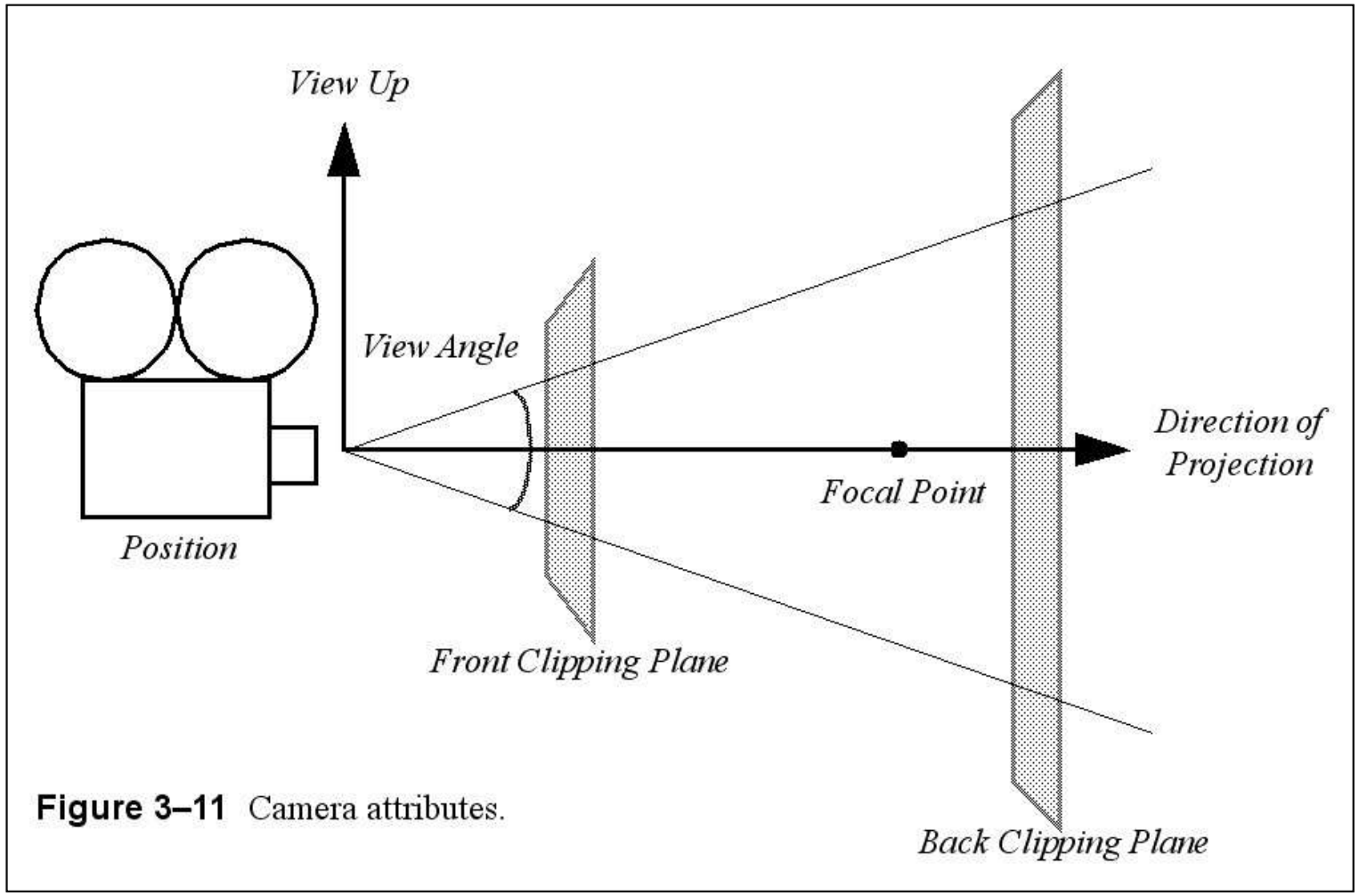
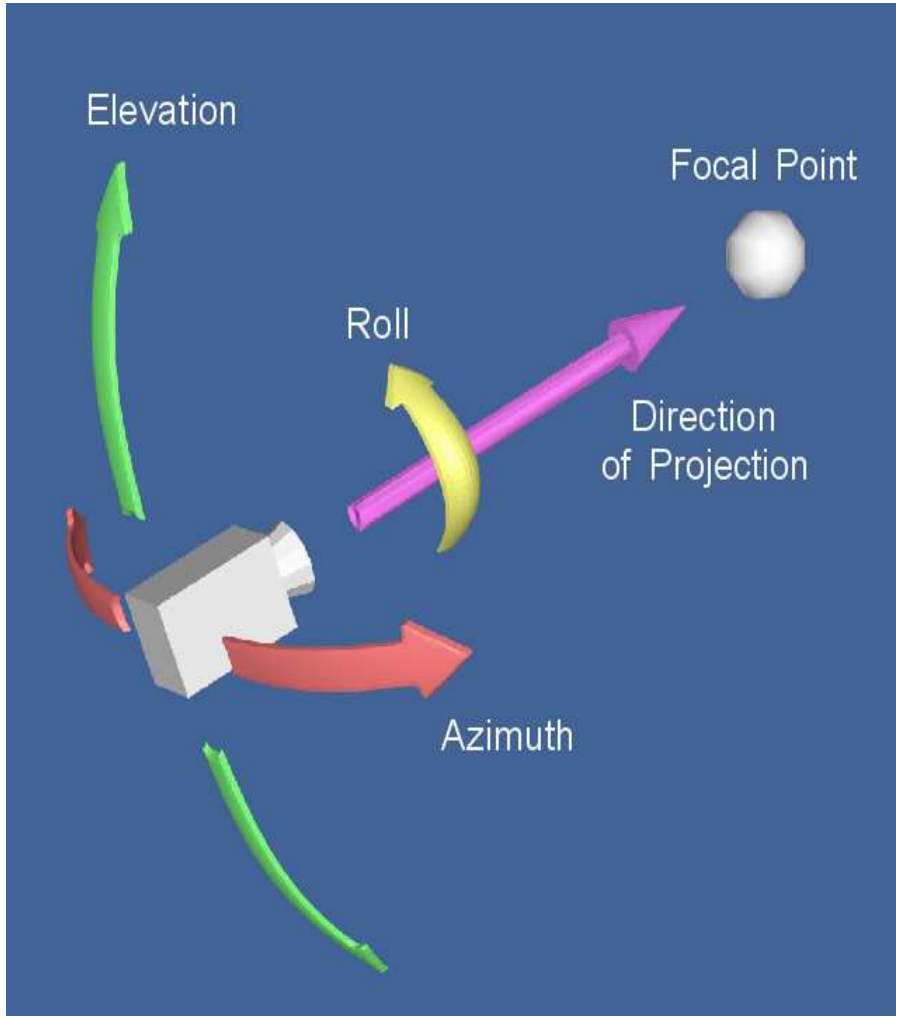
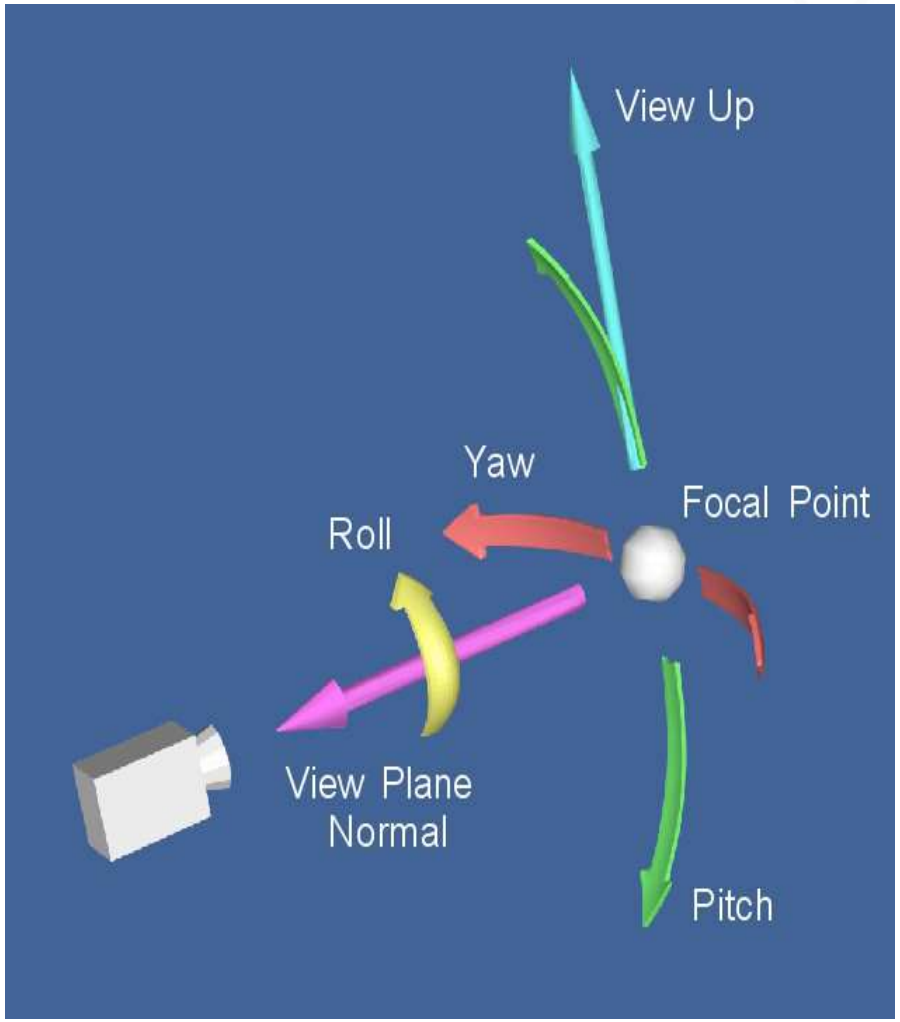


Figure 3–11 Camera attributes.

kitware

VTK Camera





kitware

vtkLight

- **Définie comment la scène est illuminée:**
  - Color – couleur de la lumière
  - Position – position de la lumière
  - FocalPoint – ou la lumière pointe
  - Intensity – intensité de la lumiere
  - Switch – off/on
  - ConeAngle – Cone des rayons qui sorte de la lumiere

vtkActor

- Objet qui doit être rendu

- Possède potentiellement une texture
- SetOrigin()/Scale/UserTransform – Specify comment cela doit être rendu
- Position – Position dans la scene
- RotateX, RotateY, RotateZ – rotation autour d'un axe

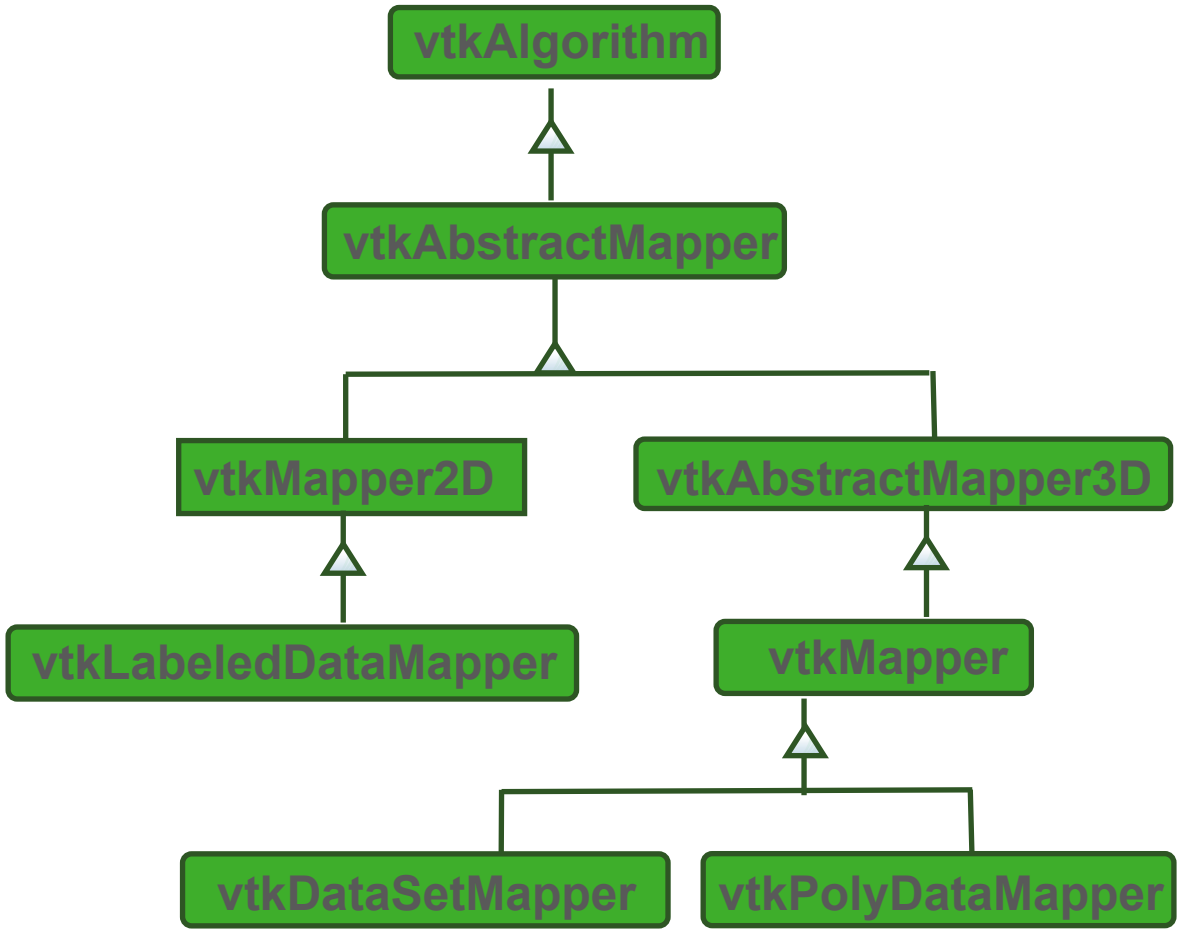
vtkProperty

- Ambient/Diffuse/SpecularColor – propriété du matériau
- Color – couleur du matériau
- Interpolation: Flat, Gouraud, PBR
- Representation: Points, Wireframe, Surface

- Opacity: contrôle la transparence

vtkMapper

- Fin du pipeline de traitement / Début du pipeline de visualisation



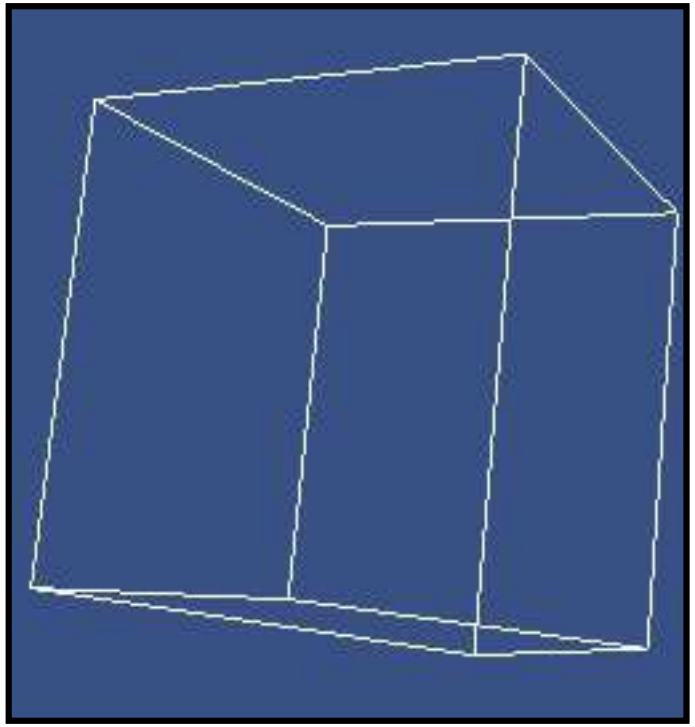
kitware

vtkProp

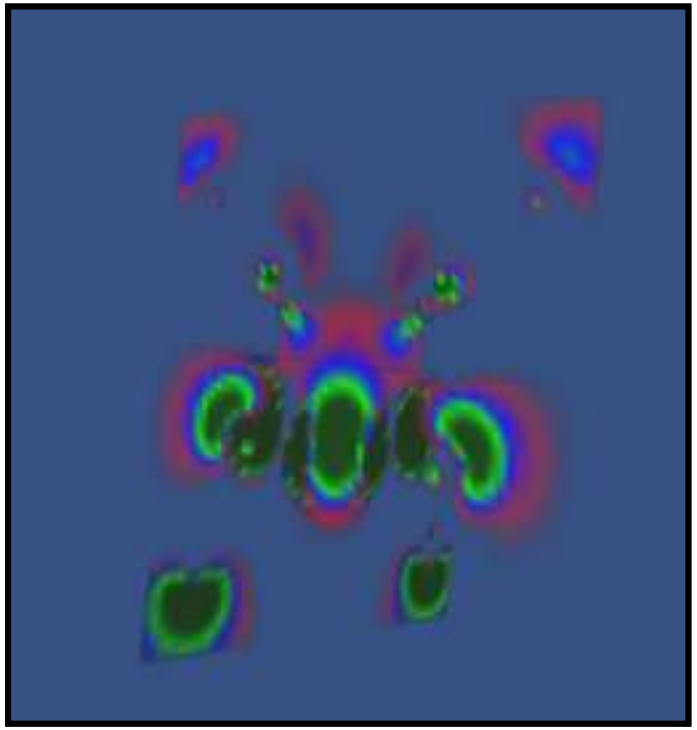
- vtkLODActor – Creation d'un acteur avec niveaux de détail
- vtkLODProp3D
- vtkFollower – toujours face à la camera
- vtkActor2D (eg. vtkTextActor)
- vtkAssembly – group of vtkProp3D
- vtkVolume – Rendu Volumique

vtkLODProp3D

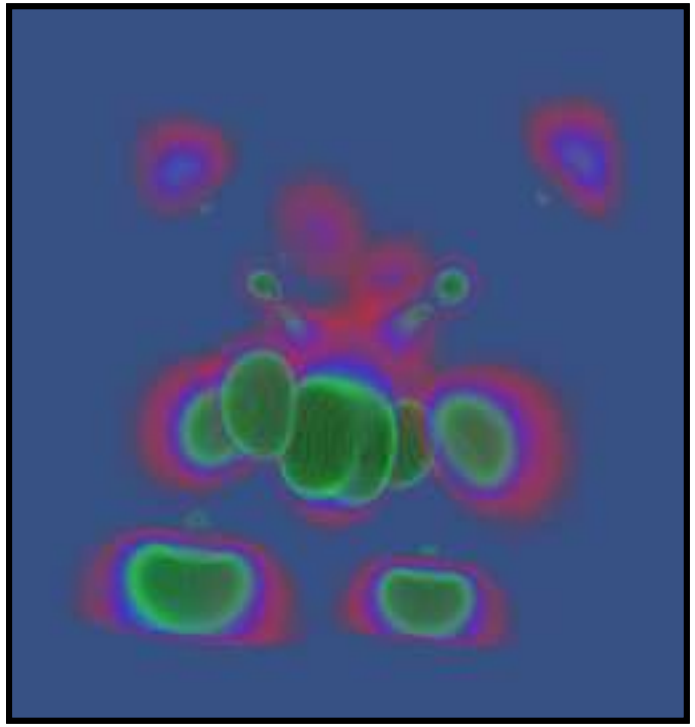
Outline



LOD 1



LOD 2



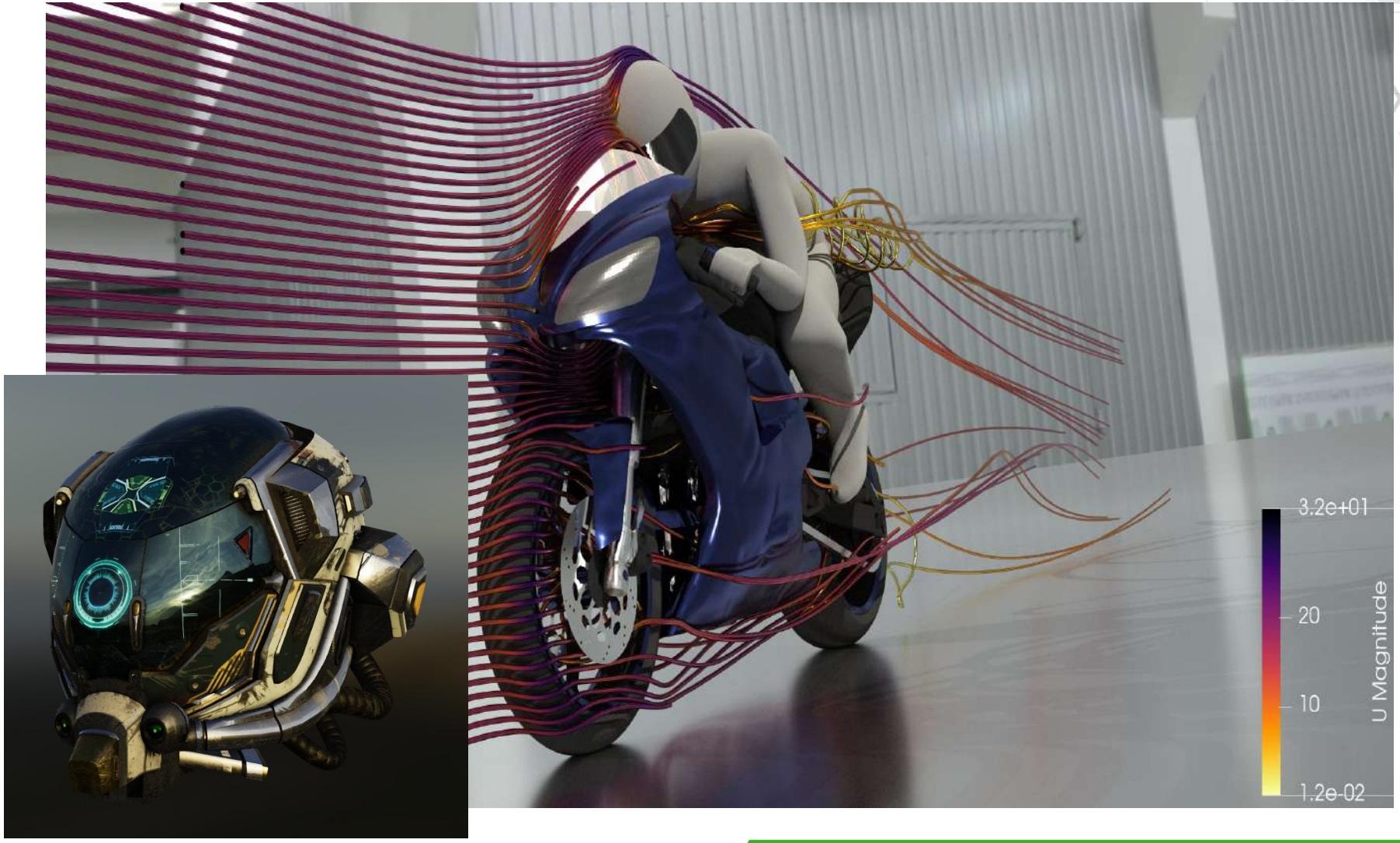
kitware

vtkAssembly

- Regroupe des acteurs entre eux
- Le résultat est similaire à un acteur

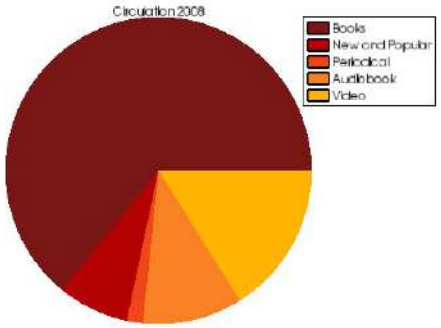
VTK – Rendu réaliste

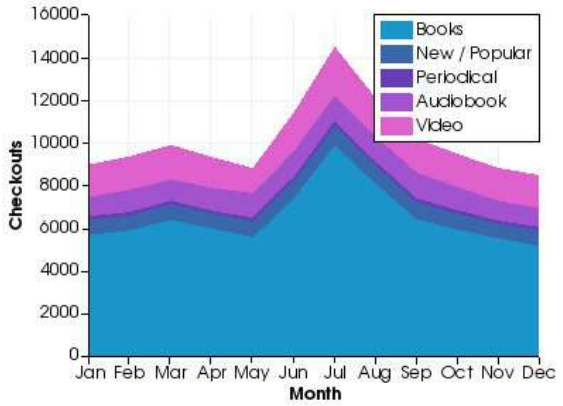
- **OSPray:**
  - Bibliothèque CPU d'Intel
- **Physically Based Rendering (PBR)**
  - GPU

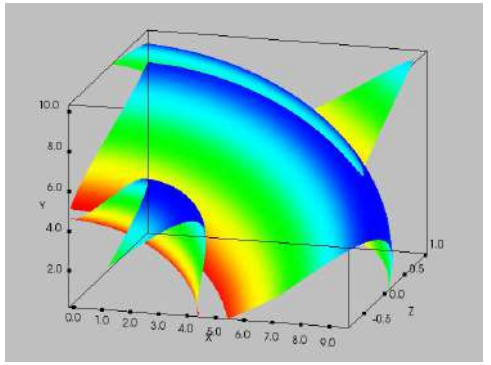


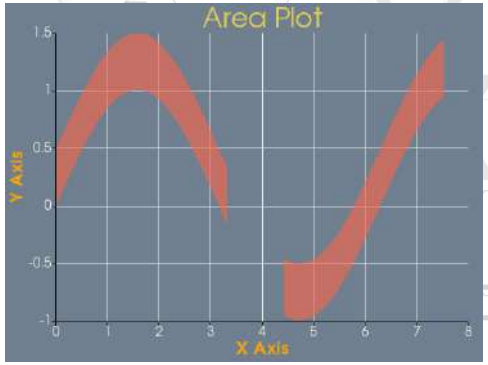
kitware

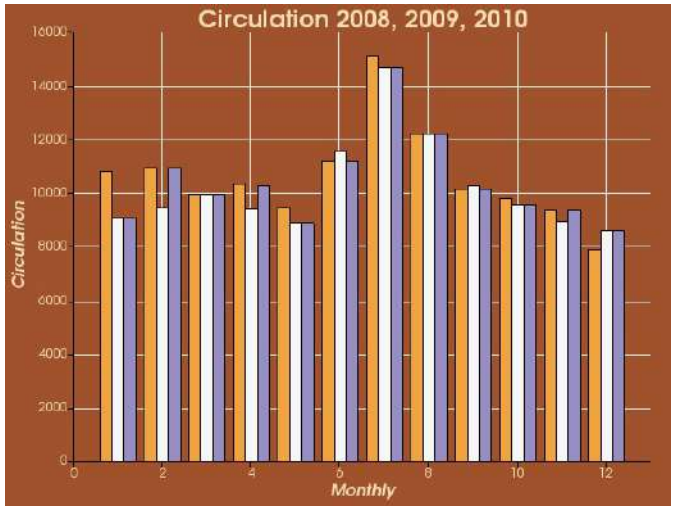
VTK 2D

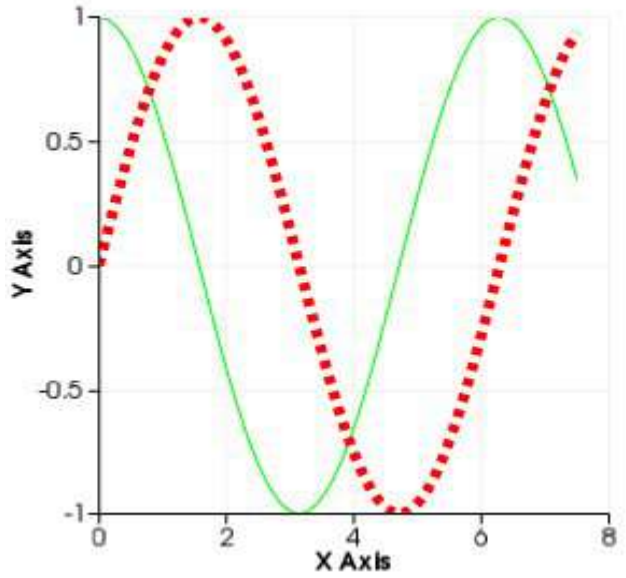


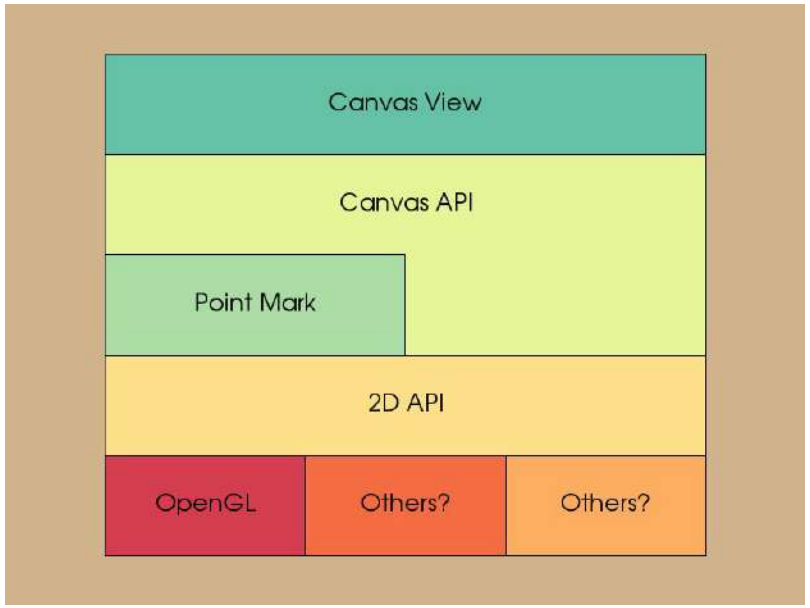












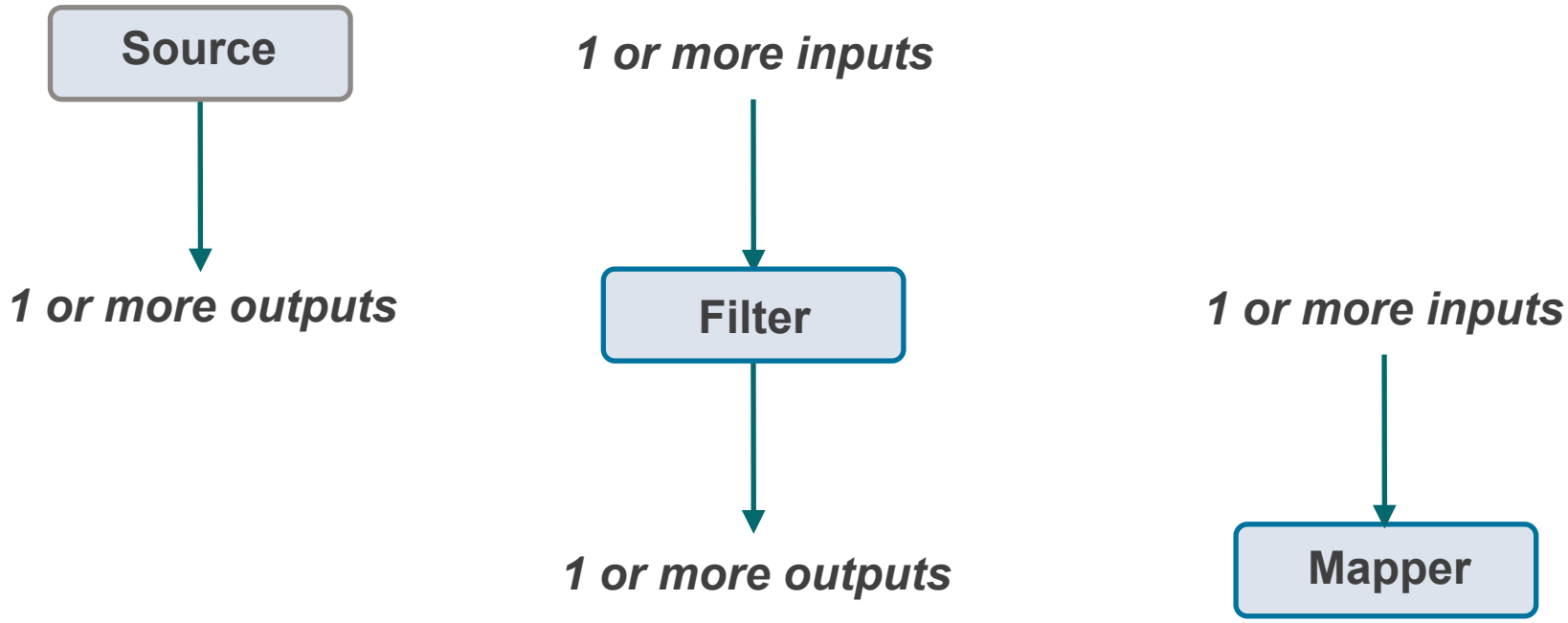
kitware

Pipeline de Traitement

kitware

Algorithmes

- Les algorithmes (vtkAlgorithm) opèrent sur des données

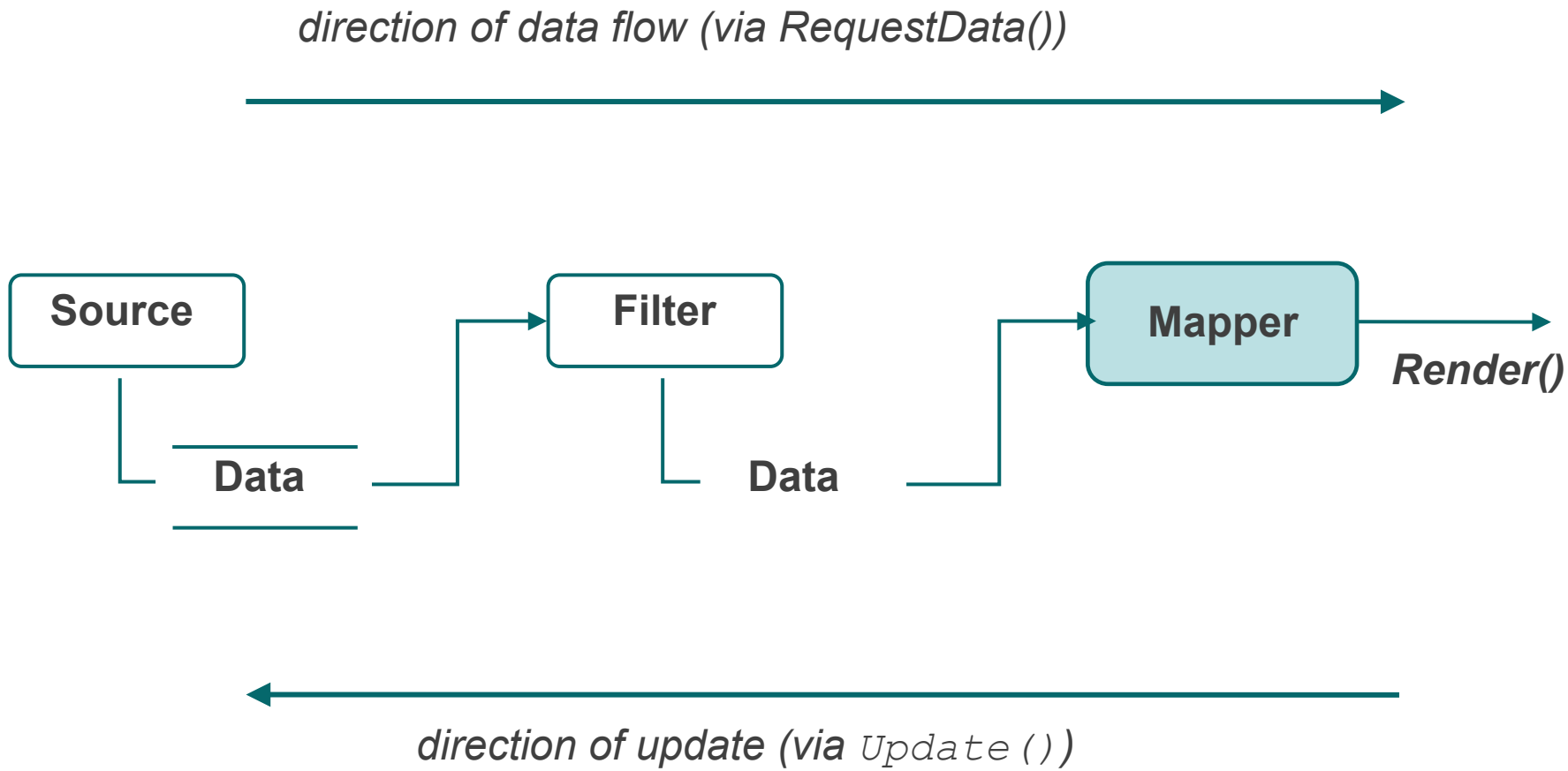


kitware

Exemples d'Algorithmes

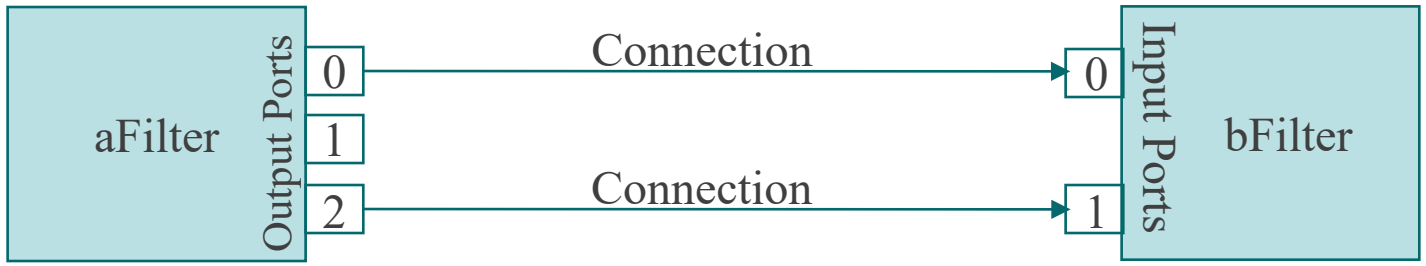
- **Readers: parse files and produce VTK data structures**
  - vtkDataSetReader, vtkPNGReader, vtkPLOT3DReader, vtkParticleReader, vtkExodusReader, vtkOpenFOAMReader
- **Contour: extract isocontours within data**
  - vtkContourFilter
- **Probing: sample attributes at set of locations**
  - vtkProbeFilter
- **Warping: bend geometry by value**
  - vtkWarpScalar, vtkWarpVector
- **Glyph: place and orient markers within data**
  - vtkGlyph2D, vtkGlyph3D
- **Streamline: advect along data to display or compute flow characteristics**
  - vtkStreamline

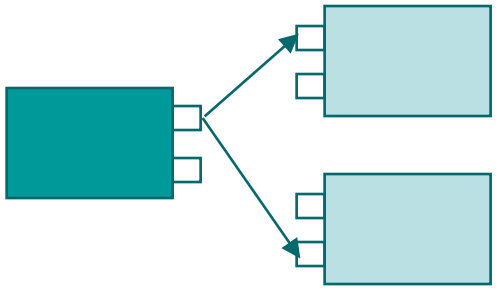
Execution du Pipeline



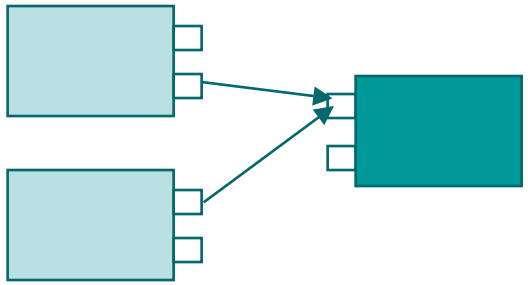
kitware

Topologie du pipeline





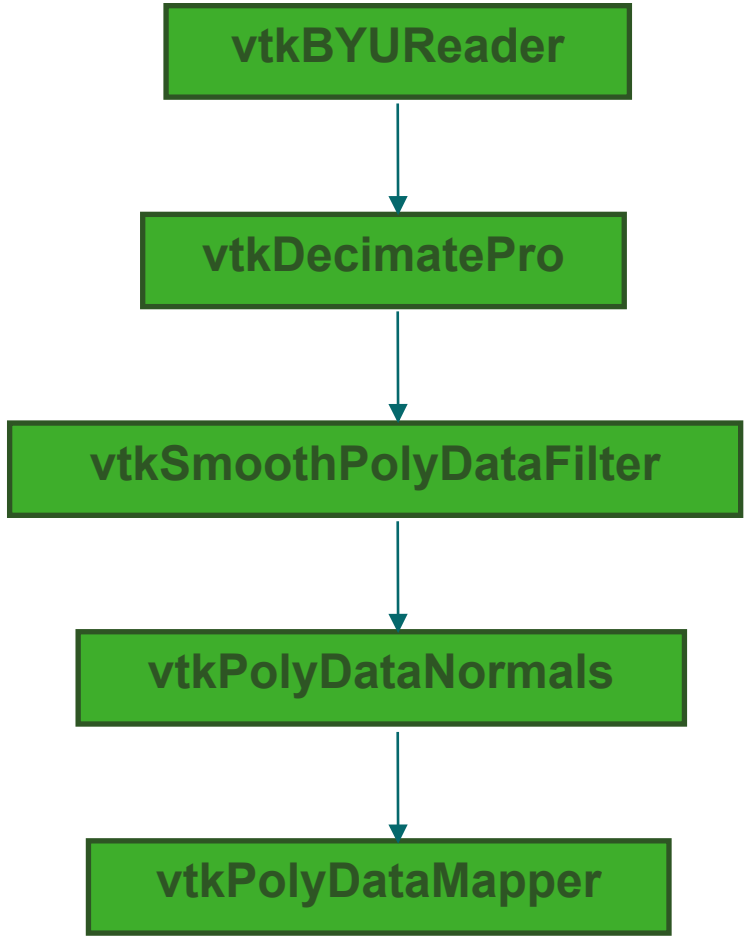
Reuse an output port: OK



Several connections on an input port: `AddInputConnection()` if allowed by the filter (ex: vtkAppendFilter)

kitware

Exemple de Pipeline



kitware

Exemple de Pipeline

```
vtkNew<vtkBYUReader> byu;
  byu->SetGeometryFileName("fran_cut.g”);

vtkNew<vtkDecimatePro> deci;
  deci->SetInputConnection(byu->GetOutputPort());
  deci->SetTargetReduction(0.9);
  deci->PreserveTopologyOn();
  deci->SetMaximumError(0.0002);
```

kitware

Exemple de Pipeline

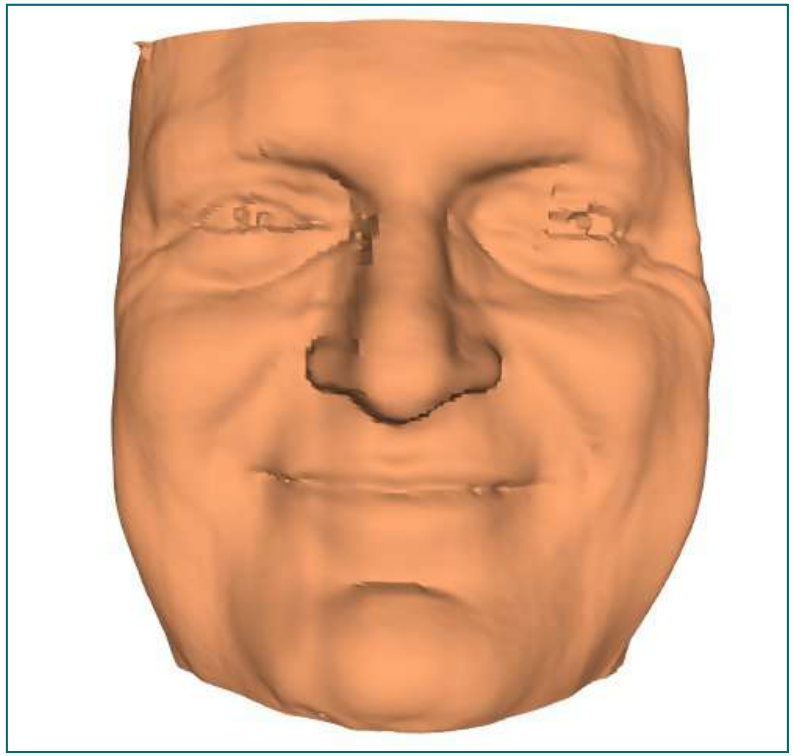
```
vtkNew<vtkSmoothPolyDataFilter> smooth;
  smooth->SetInputConnection(deci->GetOutputPort());
  smooth->SetNumberOfIterations(20);
  smooth->SetRelaxationFactor(0.05);

vtkNew<vtkPolyDataNormals> normals;
  normals->SetInputConnection(smooth->GetOutputPort());

vtkNew<vtkPolyDataMapper> cyberMapper;
  cyberMapper->SetInputConnection(normals->GetOutputPort());

vtkNew<vtkActor> cyberActor;
  cyberActor->SetMapper(cyberMapper);
  cyberActor->GetProperty()->SetColor(1.0, 0.49, 0.25);
  cyberActor->GetProperty()->SetRepresentationToWireframe();
```

Exemple de Pipeline

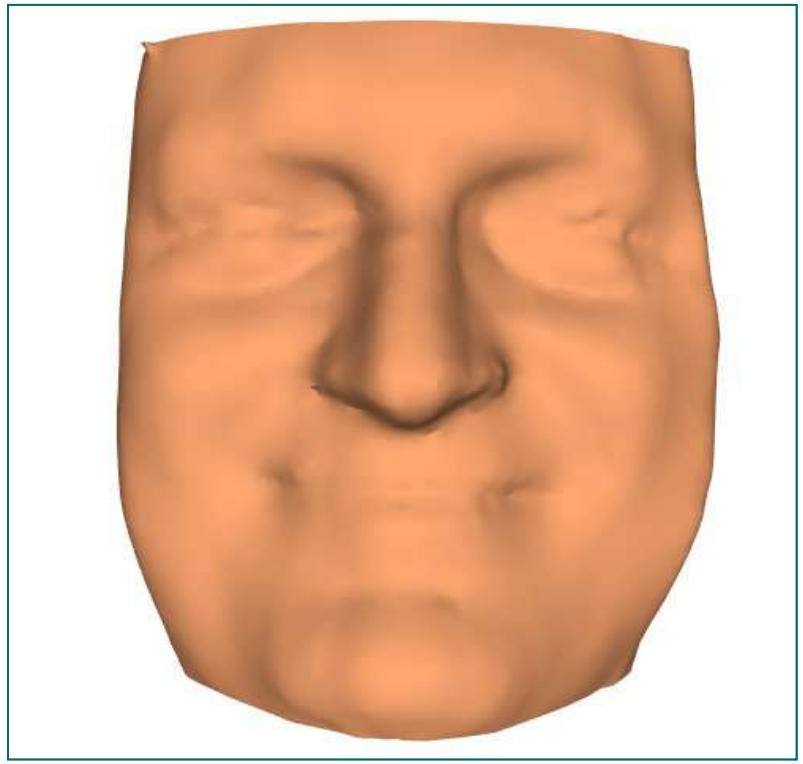
```
vtkNew<vtkRenderer> ren1;
  ren1->AddViewProp(cyberActor);
  ren1->SetBackground(1, 1, 1);

vtkNew<vtkRenderWindow> renWin;
  renWin->AddRenderer(ren1);
  renWin->SetSize(500, 500);

vtkNew<vtkRenderWindowInteractor> iren;
  iren->SetRenderWindow( renWin );
  iren->Start();
```

kitware

Resultat



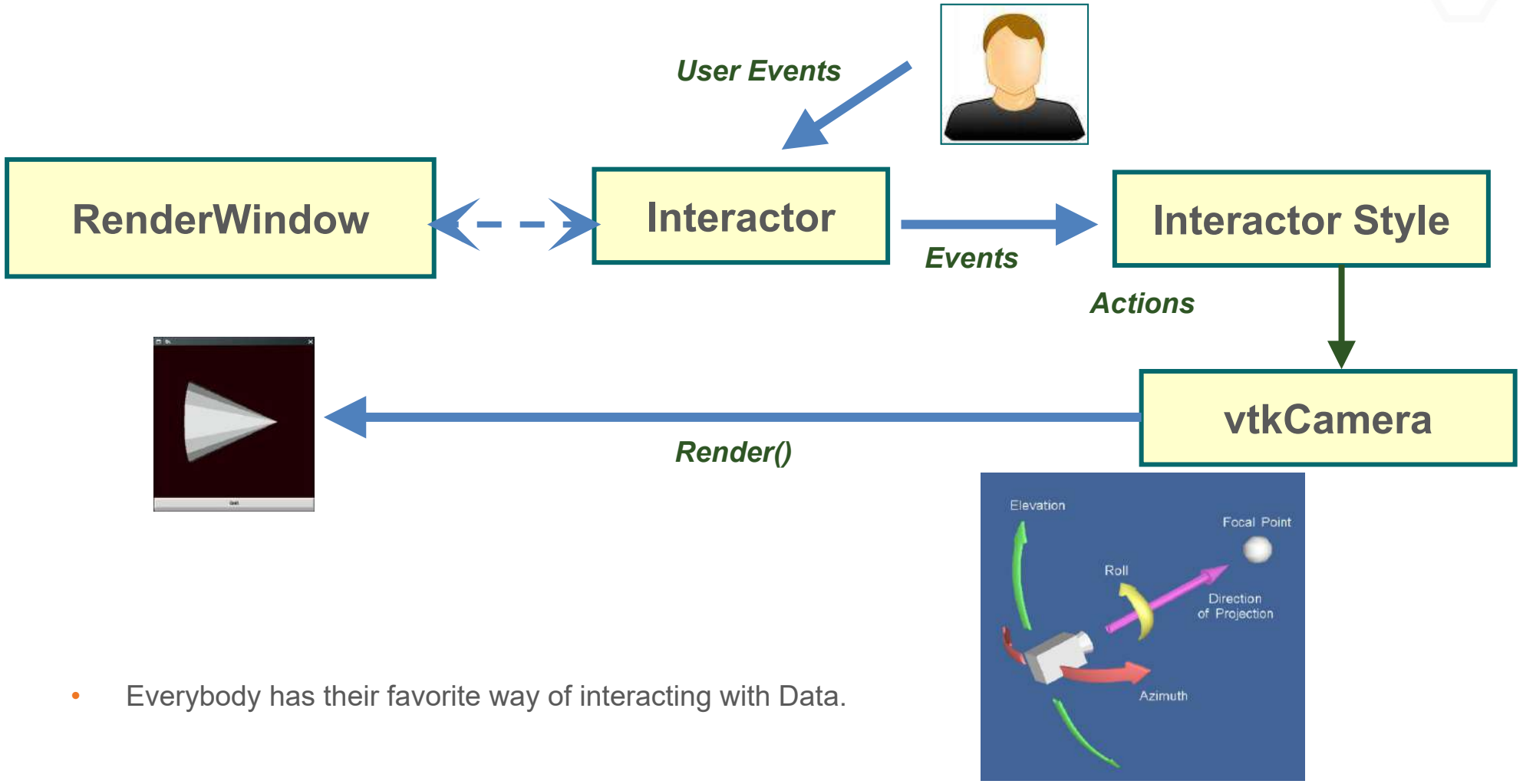
Before
(52,260 triangles)



After Decimation
and Smoothing
(7,477 triangles)

kitware

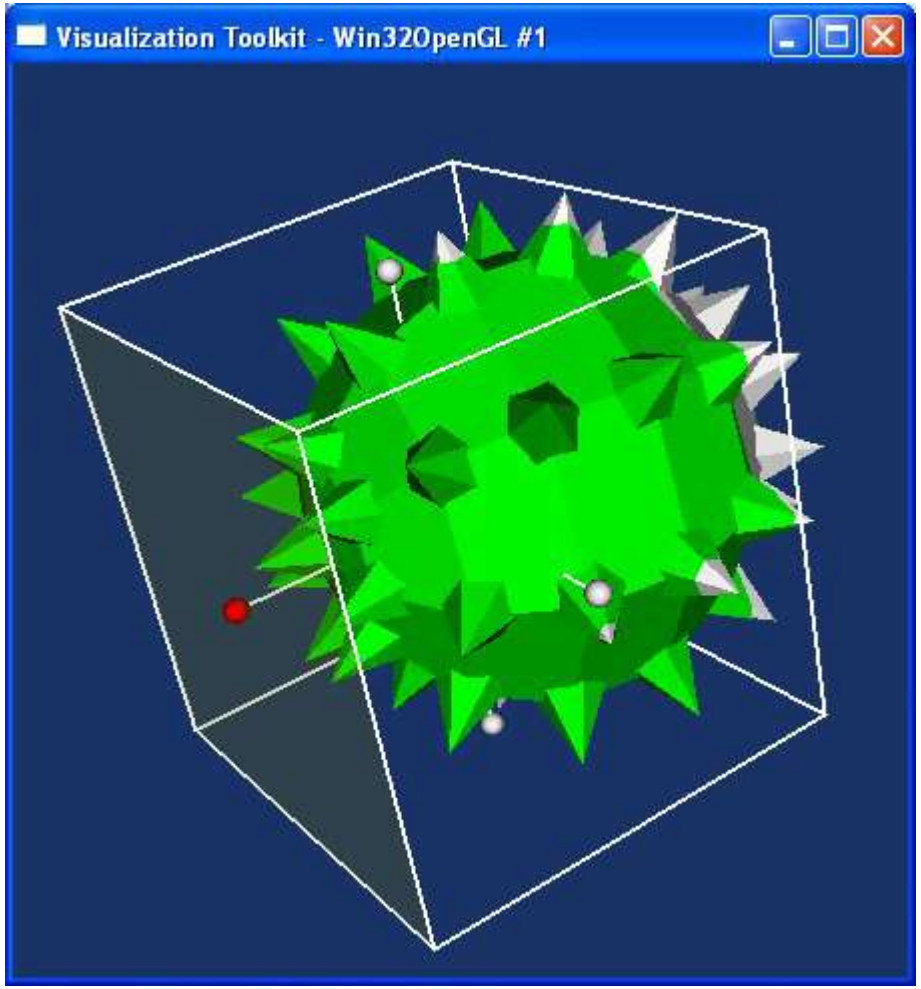
Interaction et Widgets



- Everybody has their favorite way of interacting with Data.

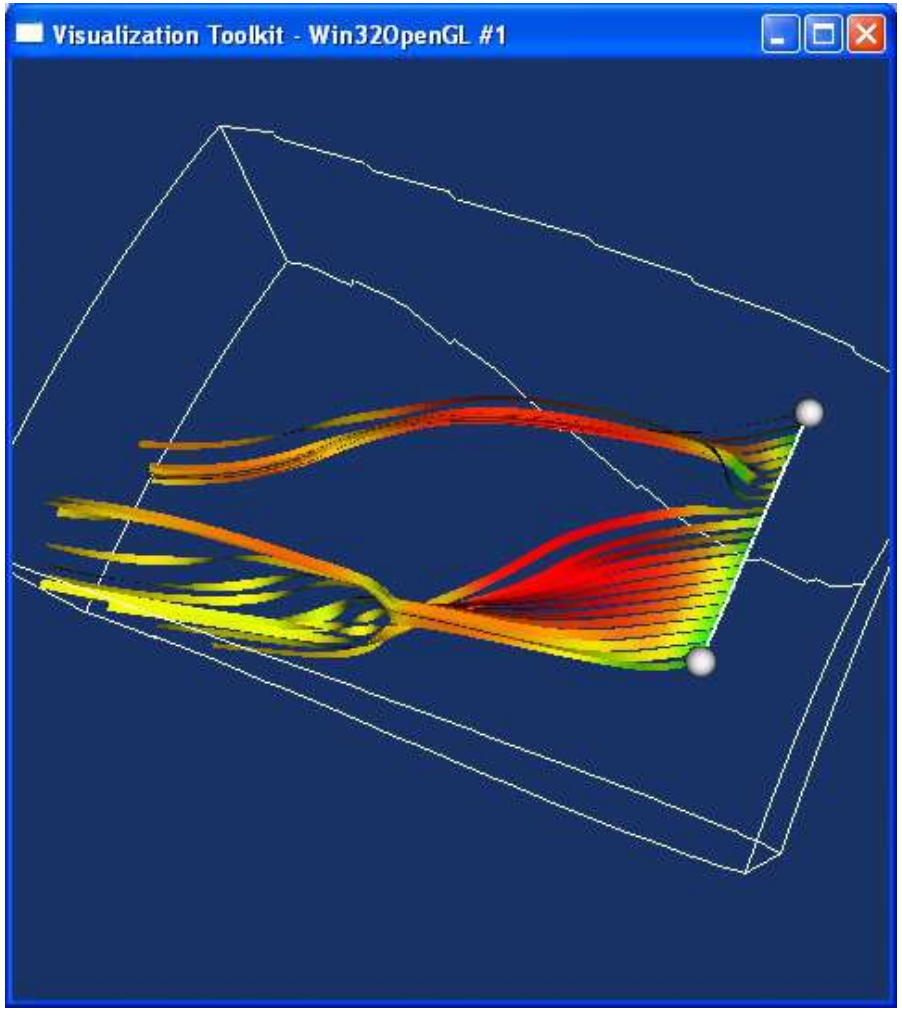
kitware

vtkBoxWidget



kitware

vtkLineWidget



kitware

TP

- En Python
- Pipeline de visualisation
- Lien avec ITK