11th Summer School of SCIENTIFIC VISUALIZATION

June 11 - 15, 2012

Introduction to Visualization ToolKit

Silvano Imboden – s.imboden@cineca.it

Paolo Quadrani – p.quadrani@cineca.it

CINECA

Index

- Characteristic
- Data Types
- Pipeline
- Demo vtkGUI

Characteristics

VKT is a C++ library

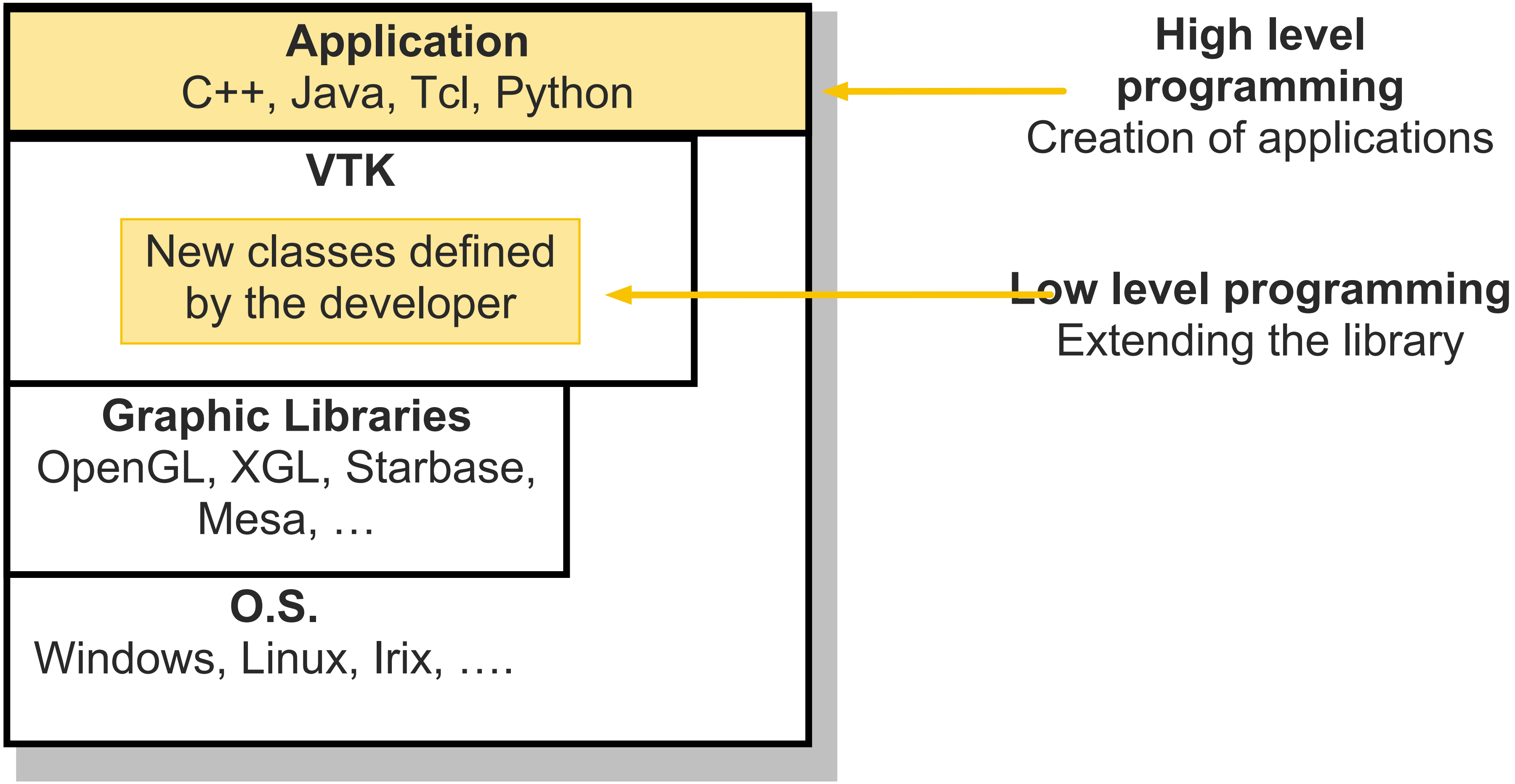
- FREE
- Open Source
- Cross Platform
- Extensible
- More than 600 classes
- Documented
- Dash Board

Characteristics (2)

- What is NOT
  - Isn’t a navigation environment
  - Isn’t a modeler

- Limits
  - Don’t support time varying data
  - Transparencies

Programming (with) VTK



Application
C++, Java, Tcl, Python

VTK

New classes defined by the developer

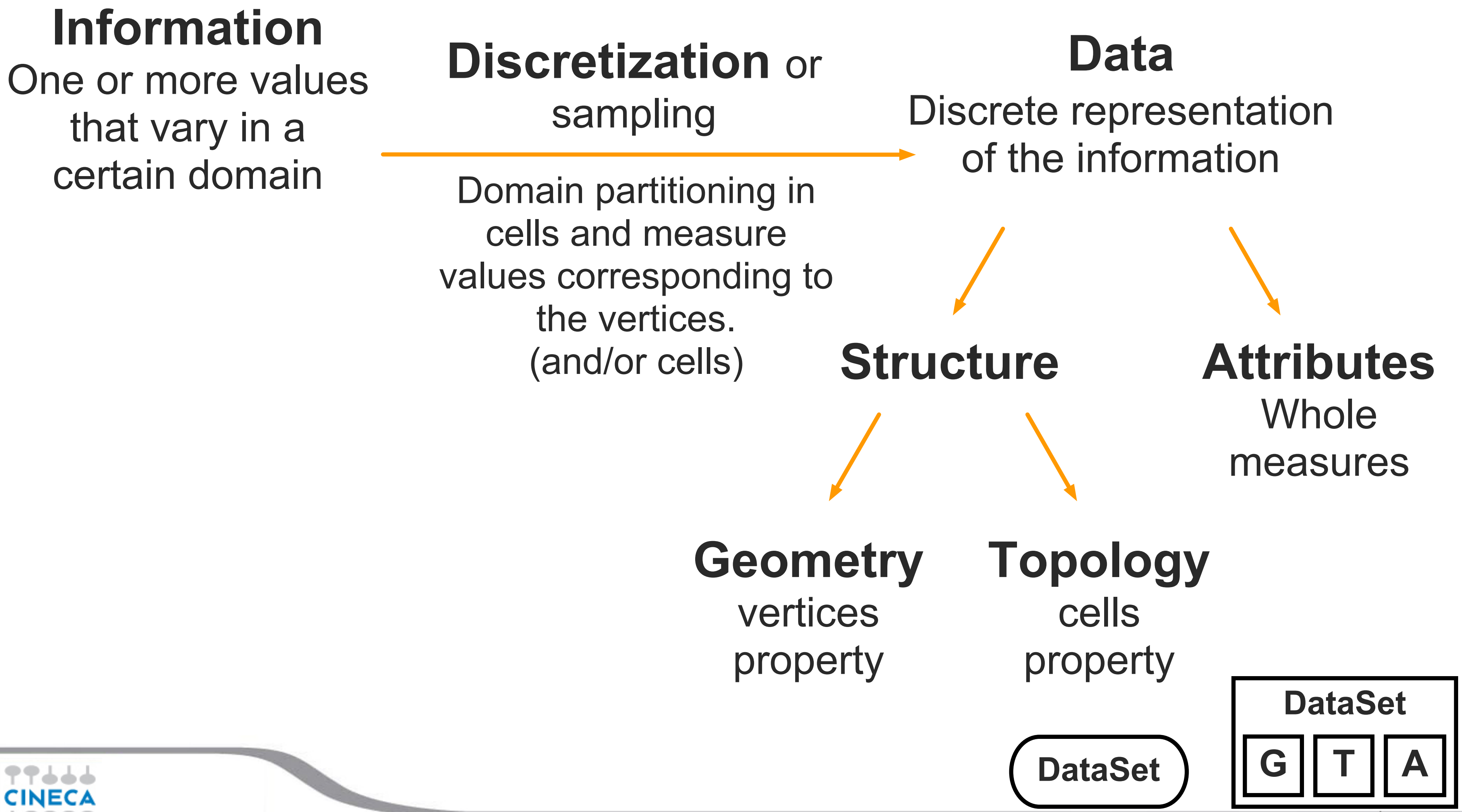
Graphic Libraries
OpenGL, XGL, Starbase, Mesa, …

O.S.
Windows, Linux, Irix, ….

High level programming
Creation of applications

Low level programming
Extending the library

Data



Information
One or more values that vary in a certain domain

Discretization or sampling

Domain partitioning in cells and measure values corresponding to the vertices.
(and/or cells)

Data
Discrete representation of the information

Structure

Attributes
Whole measures

Geometry
vertices property

Topology
cells property

DataSet

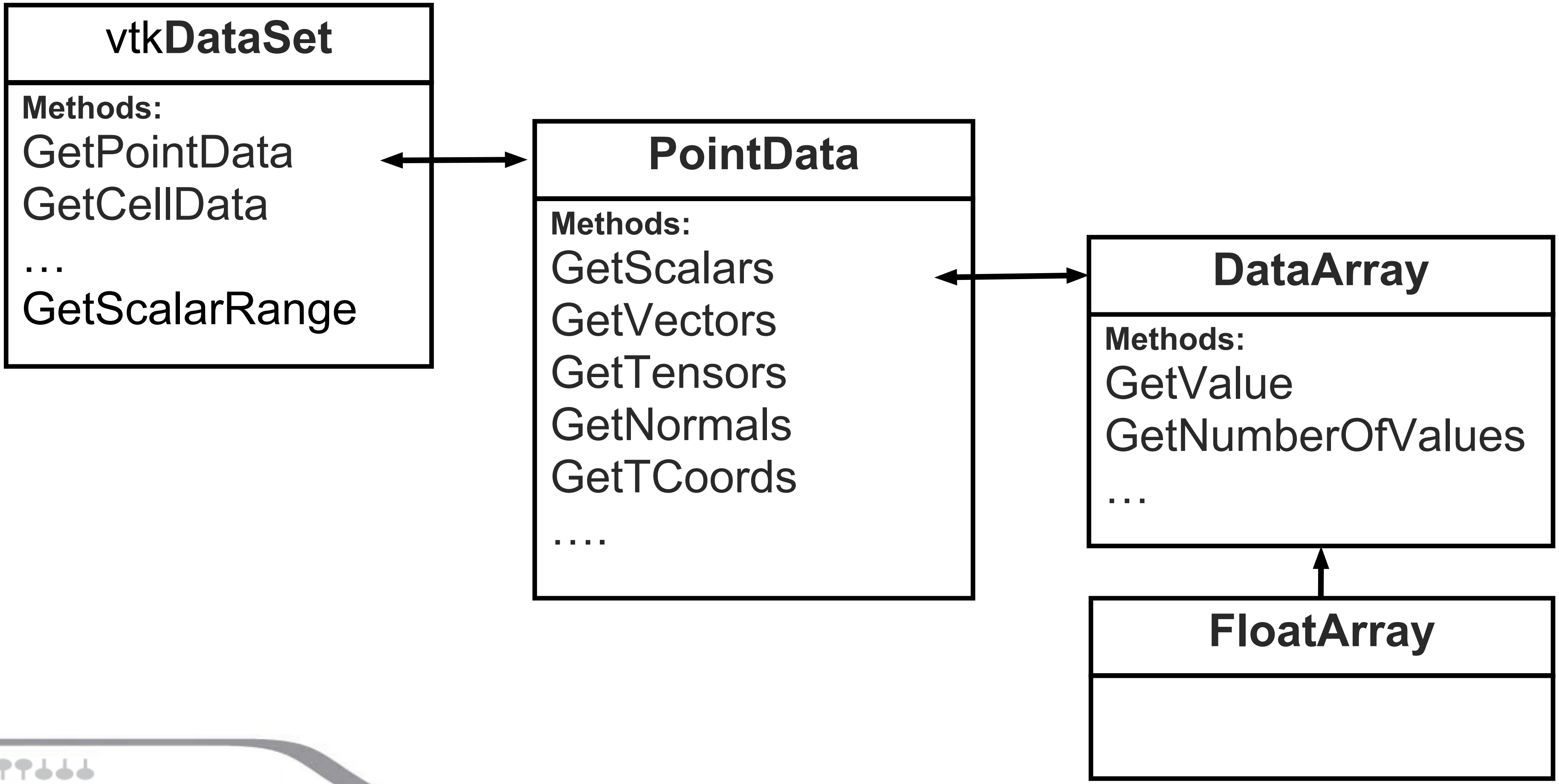
DataSet
G **T** **A**

Attributes

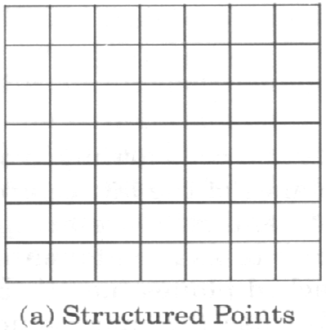
- Association
  - Points attributes
  - Cells attributes
- Type
  - Scalars (max 4 components)
  - Vectors (3 components)
  - Tensors rank 3 (9 components)
  - Normal (3 components)
  - Texture Coordinates (max 3 components)
  - Fields (n*m components)
- Representation
  - char …. double

Attributes

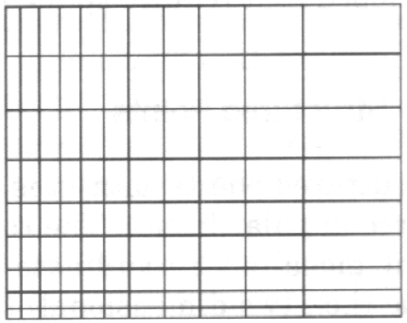
```
Dato->GetPointData()->GetScalars()->GetValue(1);
```



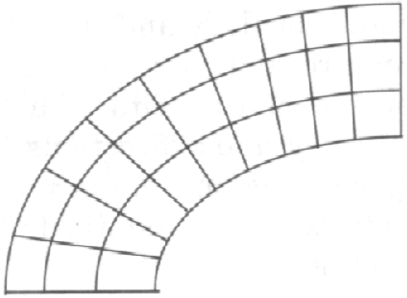
Data types



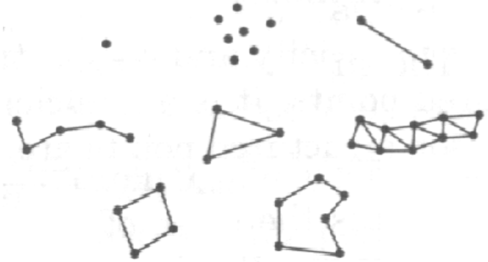
StructuredPoints (vtkImageData)



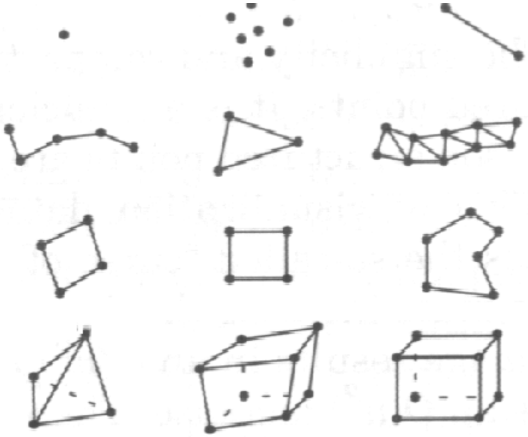
RectilinearGrid



StructuredGrid



PolyData

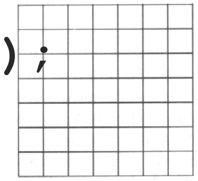


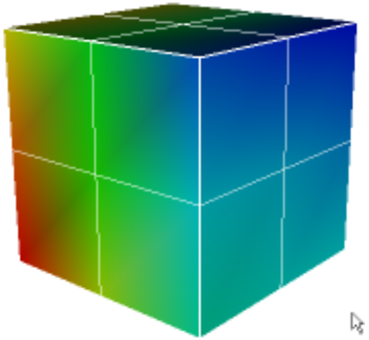
UnstructuredGrid

vtkStructuredPoints

Geometry and Topology (voxel) are both implicit and are determined using Origin, Dimensions, and Spacing.
Sample C++ code that creates a StructuredPoints

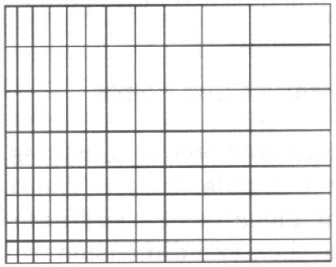
```
vtkStructuredPoints *SP = vtkStructuredPoints::New();
SP->SetOrigin     (0,0,0);
SP->SetDimensions(3,3,3);
SP->SetSpacing    (1,1,1);

vtkFloatArray *FA = vtkFloatArray::New();
for(i=0; i<27; i++)
   FA->InsertValue( i, i );
SP->GetPointData()->SetScalars( FA );
```


(a) Structured Points



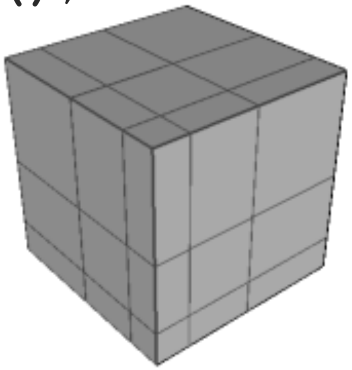
vtkRectilinearGrid

- Implicit Topology (hexahedron)
- Geometry obtained combining values of X,Y,Z coordinates specified using three arrays.



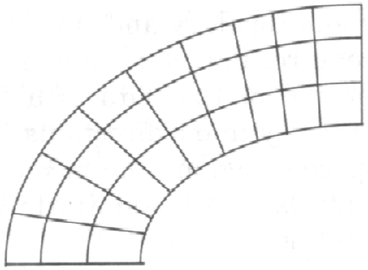
(b) Rectilinear Grid

```
vtkFloatArray *FA = vtkFloatArray::New();
FA->InsertValue( 0, 0 );
FA->InsertValue( 1, 1 );
FA->InsertValue( 2, 3 );
FA->InsertValue( 3, 6 );

vtkRectilinearGrid *RG = vtkRectilinearGrid::New();
RG->SetDimensions (4,4,4);
RG->SetXCoordinates(FA);
RG->SetYCoordinates(FA);
RG->SetZCoordinates(FA);
```



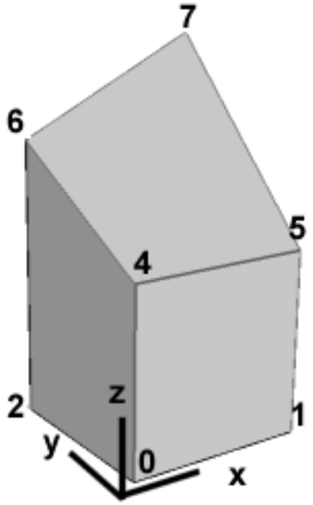
vtkStructuredGrid

- Implicit Topology – (hexahedron)
- Explicit Geometry



(c) Structured Grid

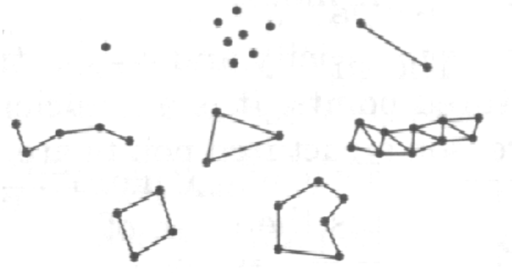
```
vtkPoints *P = vtkPoints::New();
P->InsertNextPoint( 0,0,0 );
P->InsertNextPoint( 1,0,0 );
P->InsertNextPoint( 0,1,0 );
P->InsertNextPoint( 1,1,0 );
P->InsertNextPoint( 0,0,1 );
P->InsertNextPoint( 1,0,1 );
P->InsertNextPoint( 0,1,1.5 );
P->InsertNextPoint( 1,1,2   );

vtkStructuredGrid *SG = vtkStructuredGrid::New();
SG->SetDimensions (2,2,2);
SG->SetPoints(P);
```



vtkPolyData

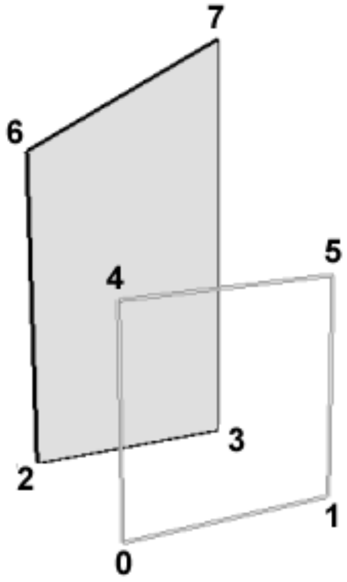
- Geometry and Topology both explicit
- Cells are subdivided in four classes:

Verts, Lines, Polys, Strip



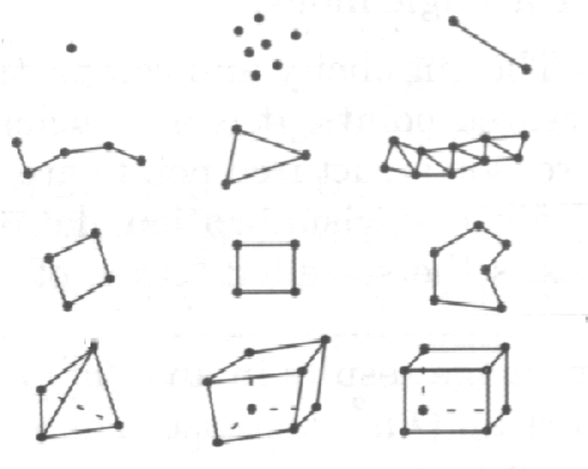
(e) Polygonal Data

```
vtkCellArray *CA = vtkCellArray::New();
CA->InsertNextCell( 4 );
CA->InsertCellPoint( 3 );
CA->InsertCellPoint( 2 );
CA->InsertCellPoint( 6 );
CA->InsertCellPoint( 7 );
// in the same way, create CA2
//inserting indexes 0,1,5,4,0

vtkPolyData *PD = vtkPolyData::New();
PD->SetPoints( P  );
PD->SetPolys ( CA );
SG->SetLines ( CA2);
```



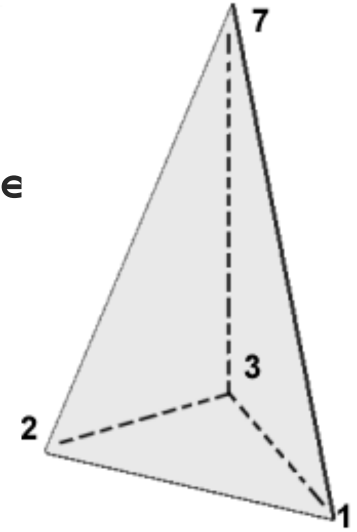
vtkUnstructuredGrid

- Geometry and Topology both explicit
- Celle can be 0,1,2 or 3D



(f) Unstructured Grid

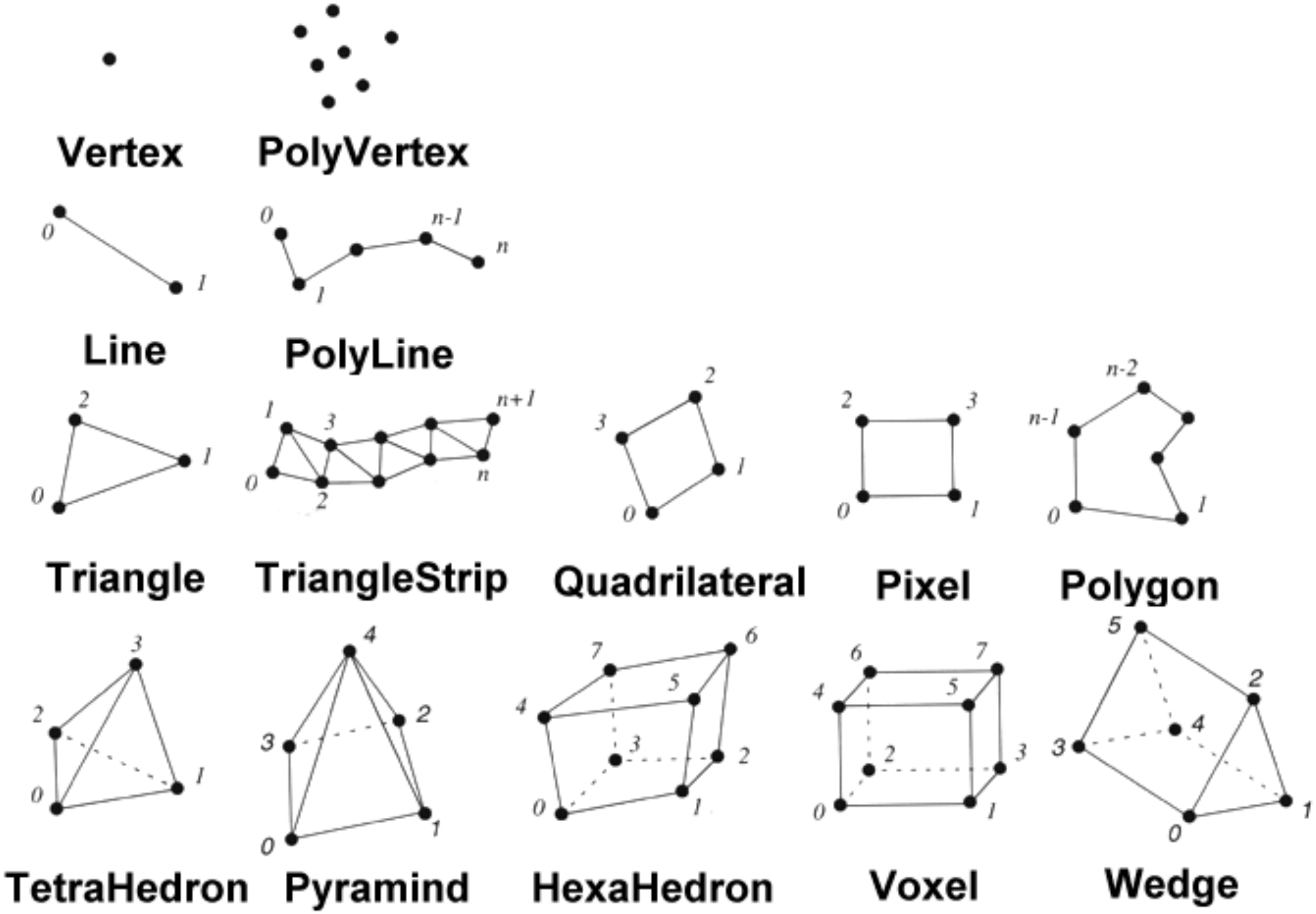
```
vtkIdList *IL = vtkIdList::New();
IL->InsertNextId( 1 );
IL->InsertNextId( 2 );
IL->InsertNextId( 3 );
IL->InsertNextId( 7 );

vtkUnstructuredGrid *UG = vtkUnstructuredGrid::Ne
UG->SetPoints( P );
UG->InsertNextCell( VTK_TETRA, IL );
```



see vtkCellType.h

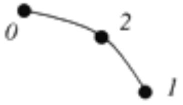
Cell types

- 0d
- 1d
- 2d
- 3d

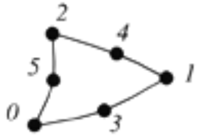


Cell types

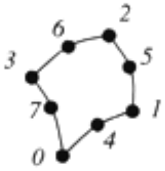
- Non linear Cells



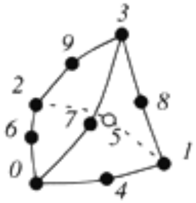
Quadratic Edge



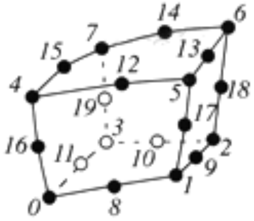
Quadratic Triangle



Quadratic Quadrilateral



Quadratic Tetrahedron



Quadratic Hexahedron

Data querying

- Geometry
  - GetNumberOfPoints, GetPoint, FindPoint
  - GetCenter, GetBounds, GetLenght,
- Topology
  - GetNumberOfCells, GetCell, FindCell, IntersectWithLine
  - GetPointCells, GetCellPoints, GetCellNeighbors
- Attributes
  - GetScalarRange
  - GetScalar,GetVector ….
  - EvaluatePosition

Supported formats

- Reader/Writer – works only on one data
  - Native VTK format (ASCII, Binary, XML)
  - Images: BMP, JPEG, TIFF, PNG, PNM, RAW (also 3D), DEM, GESigna
  - Surfaces: STL, MCubes, PLY
  - Volumes: Plot3D, SLC, UGFacet
  - Other: Particles

- Importer/Exporter – works only on the scene
  - Import : 3DS, VRML
  - Export : IVO, OBJ, OOGLE, RIB, VRML

Data Import

Strategies:

- “ASCII ART”
  - The VTK ASCII format is really simple, in some cases you have only to add a header to the data and transform it in VTK.
- Create VTK data programmatically
  - If you are able to write a program that is able to read the data to be imported, can be created a VTK data type as seen in the previous slides (Programmable Source)
- Build a Reader
  - In case of frequent usage, building a reader is the best way to proceed, but also the more expensive. At the end it can be donated to the community.

Pipeline

data-flow paradigm

- Create a visualization using VTK means:
  - Find out in the VTK libraries the necessary filters
  - Link them together (this is called **pipeline**)

In simple cases the pipeline will be a linear chain, while in more complex cases it can be a graph.

- The pipeline ends with a Window object

Showing this window, we will see the first result of the elaboration; you can then pass to the interactive phase that allows you:

  - Change the object's properties or how they are linked
  - Evaluate the obtained result eventually go back to the previous value.

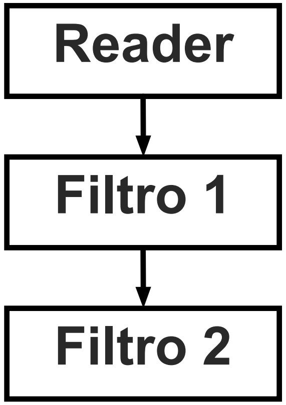
- No more code is strictly required.

(execution demand driven)

Filters

- A **Filter** is an object that can elaborate a data, in particular receive a data from its **input**, elaborate it considering its **parameters**, gives the result using its **output**.

In some cases, filters don't have inputs (Readers, Source) or don't have the output (Writer, Mapper)



Reader

Filtro 1

Filtro 2

Read a data from a file

Apply a certain transformation

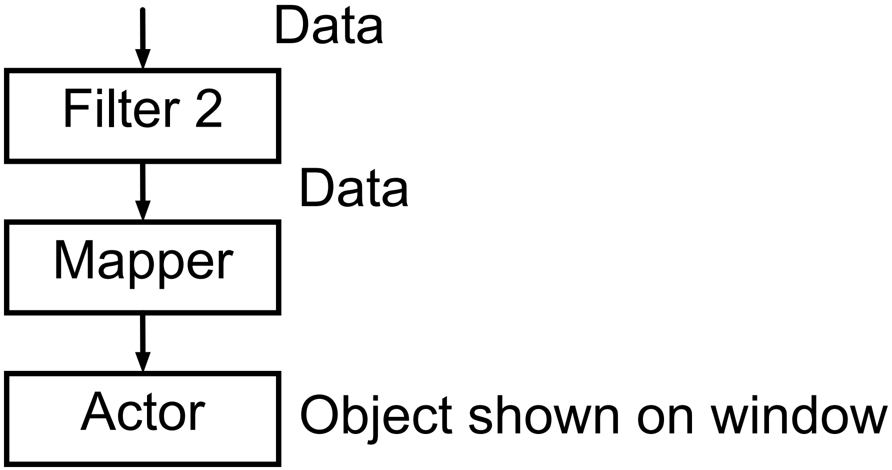
Apply a second transformation

- Multiple Input / Output
- Multiple Fan-Out

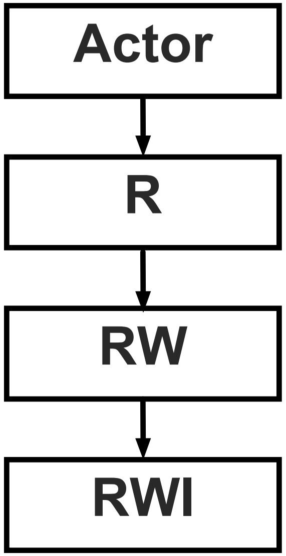
- Developer doesn't create data

Mapper and Actors

- In general a chain of filters end with two objects: the Mapper and the Actor.
- The Mapper specify the interface between data and graphics primitives
- The Actor represents one of the objects shown into the window. The Actor is always linked to a Mapper.



R,RW,RWI



Actor

R

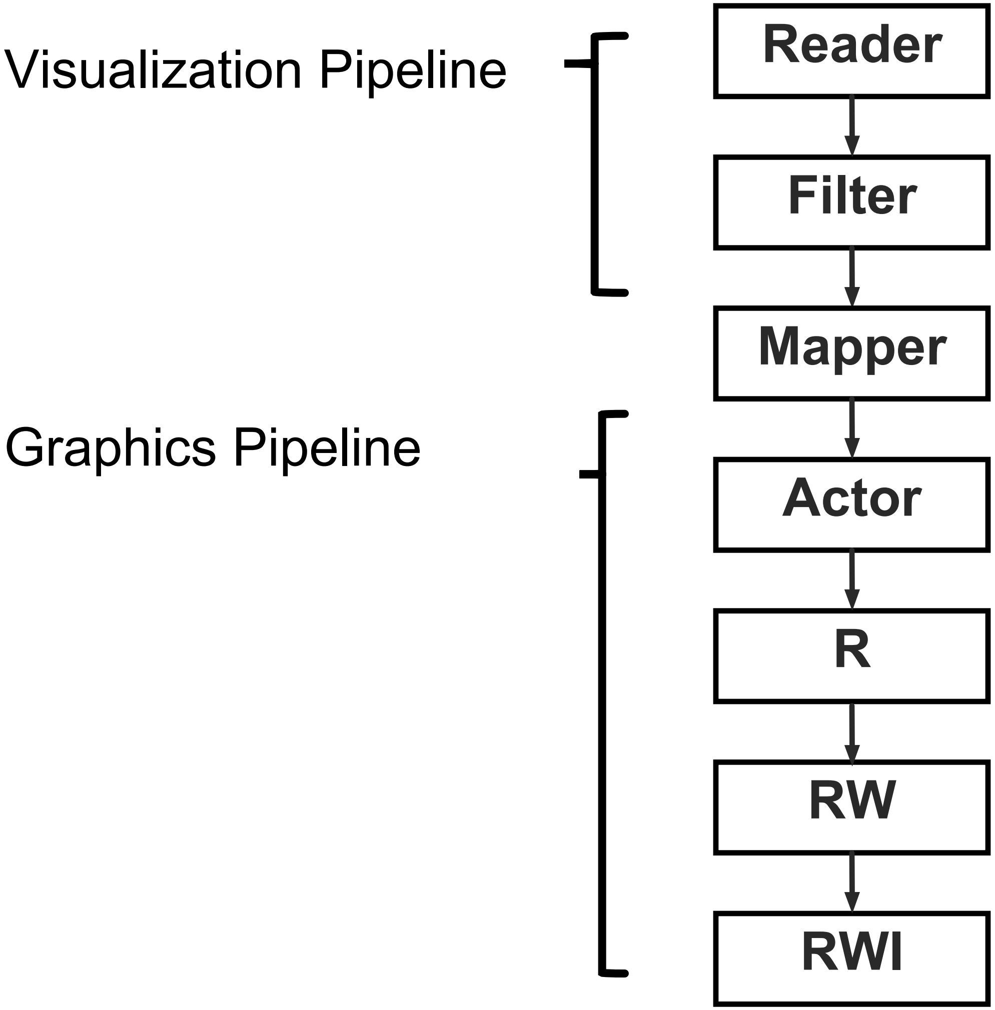
RW

RWI

The pipeline visualization happen using the following objects:

- The **Renderer** receives one or more actors and represents “the visualized scene”.
- The **RenderWindow** represents the window that you see on the screen and contains the scene.
- The **RenderWindowInteractor** add the interactivity, the possibility to manage the Mouse events. By default the interactor allows you to change the scene point of view.

Complete Pipeline



Visualization Pipeline

Graphics Pipeline

Reader

Filter

Mapper

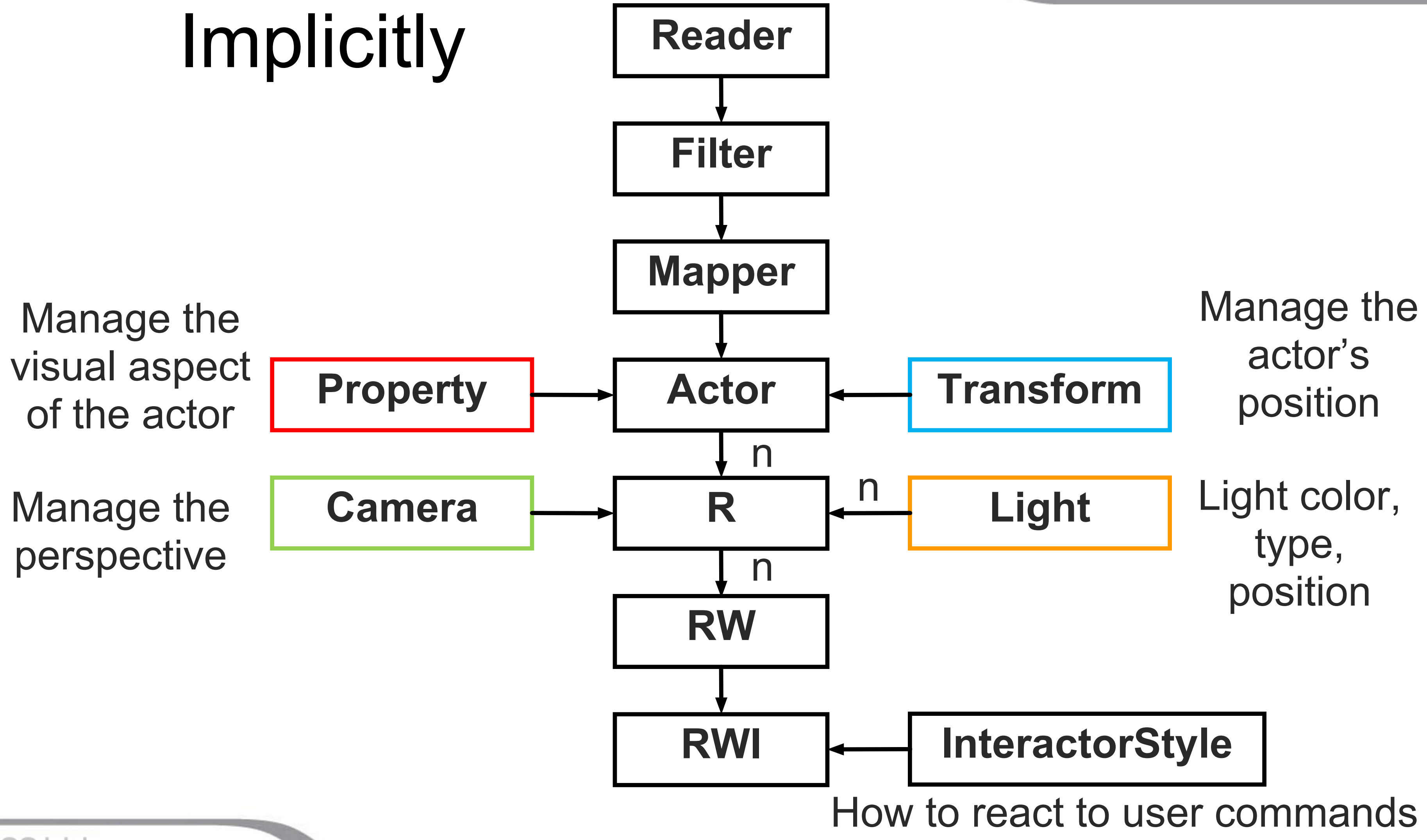
Actor

R

RW

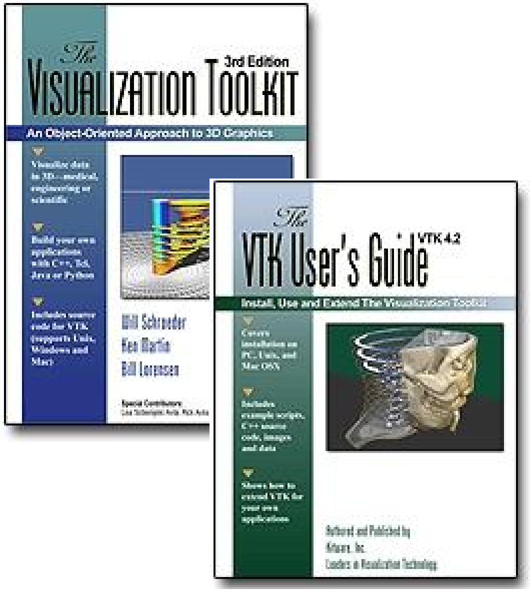
RWI

Objects Created Implicitly


Reader
Filter
Mapper
Manage the visual aspect of the actor
Property
Actor
Transform
Manage the actor's position
n
Manage the perspective
Camera
R
n
Light
Light color, type, position
n
RW
RWI
InteractorStyle
How to react to user commands

Tools

- User Guide
- Examples (http://www.vtk.org/Wiki/VTK/Examples)
- Help
- Sources
- Wiki
- Mailing List
- Git / DashBoard / BugList


The Visualization Toolkit
3rd Edition
An Object-Oriented Approach to 3D Graphics
Will Schroeder
Ken Martin
Bill Lorensen
The VTK User's Guide
VTK 4.2
Install, Use and Extend The Visualization Toolkit
Authored and Published by
Kitware, Inc.

Thank you ☺