Introduction to VTK

MACbioIDi – February – March 2018

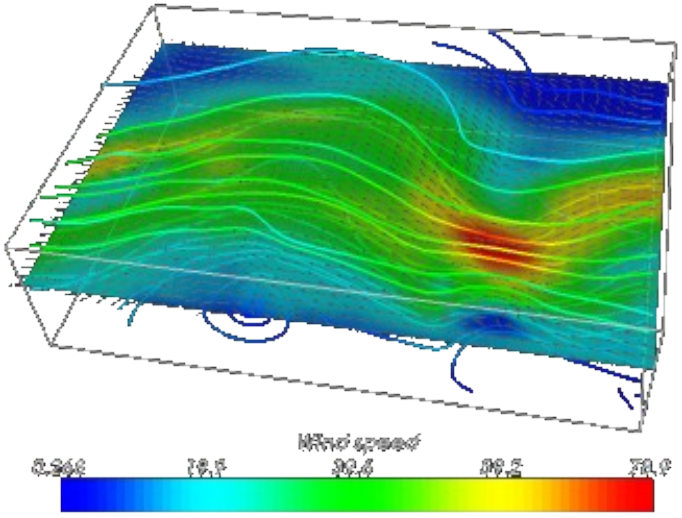
ULPGC
Tecnología Médica para el Desarrollo Sostenible
MAC 2014-2020 Cooperación Territorial
Interreg
Fondo Europeo de Desarrollo Regional
EUROPEAN UNION

Today's Topics

- Introduction
- Preparing Working Enviroment
- VTK's pipeline

Introduction

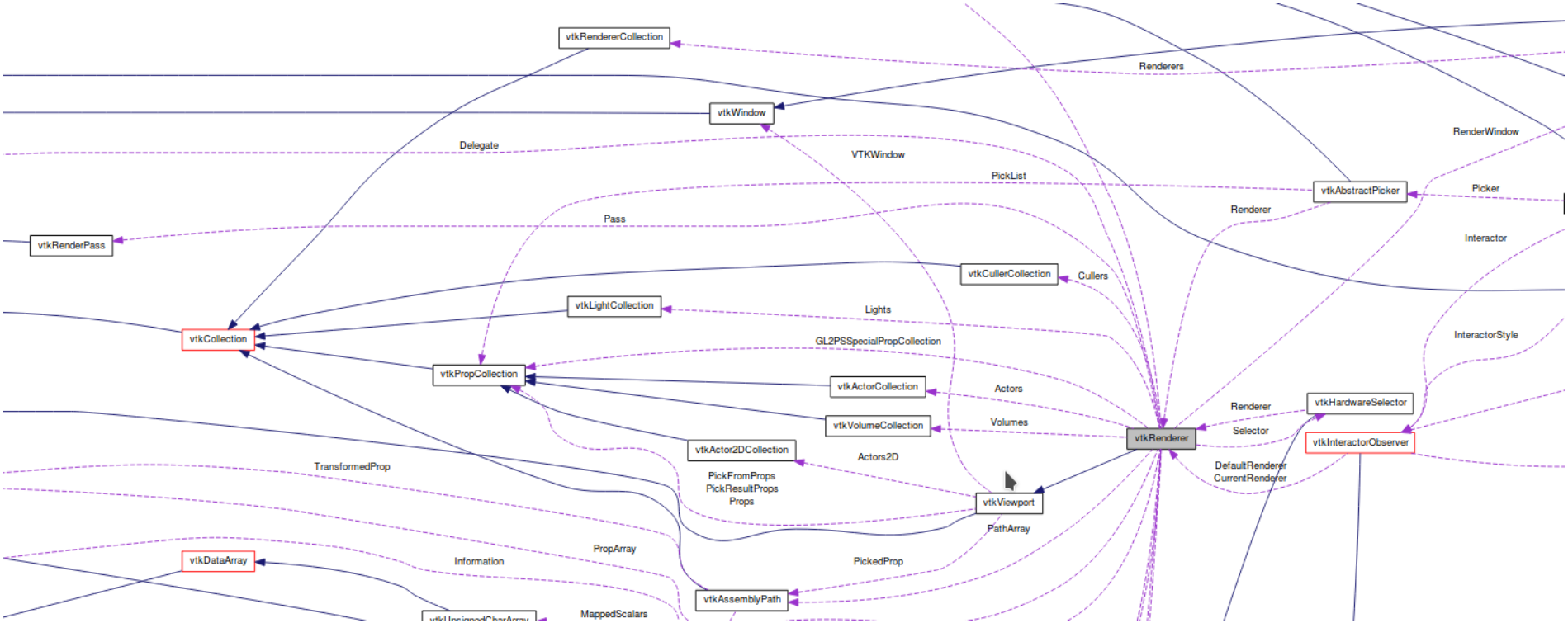
- Visualization Toolkit (**VTK**)
  - Open Sources
  - Scientific visualization
  - 3D Computer Graphics
  - Mesh and Images Processing



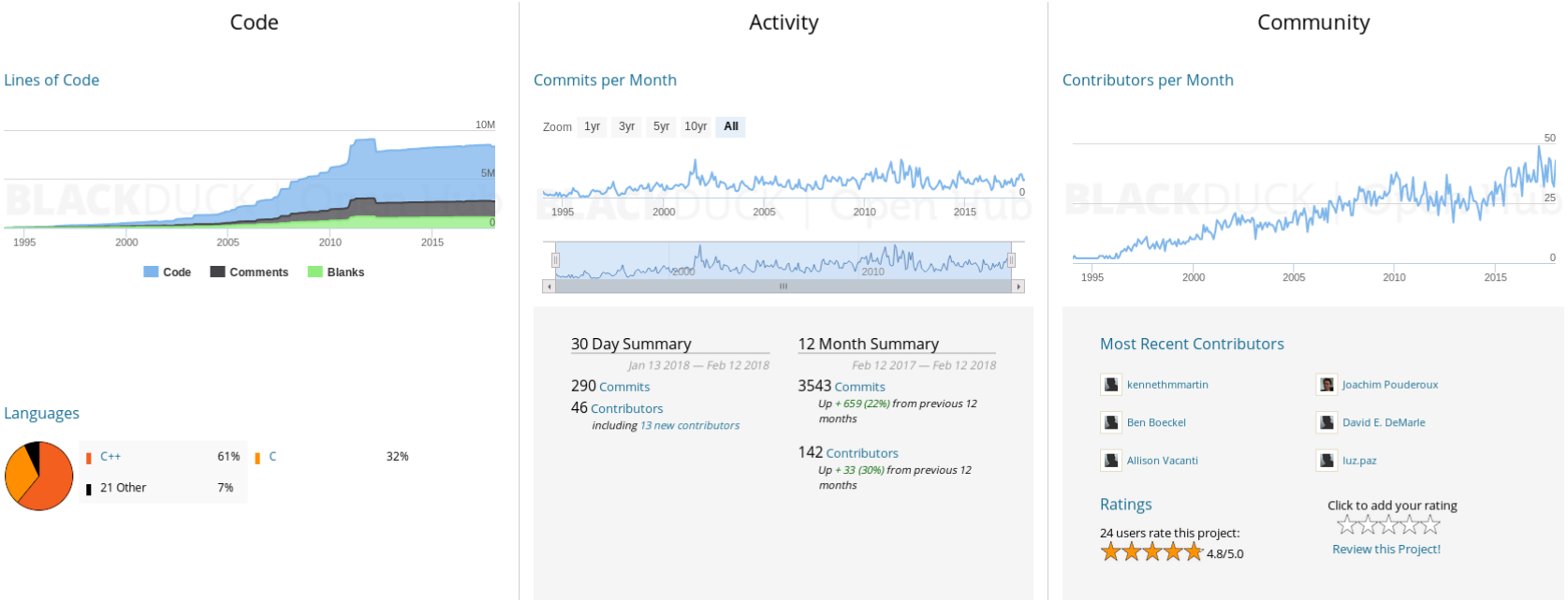
- Managed by **Kitware Inc**.

Introduction

- Source code in C++
  - Object-oriented design



Introduction



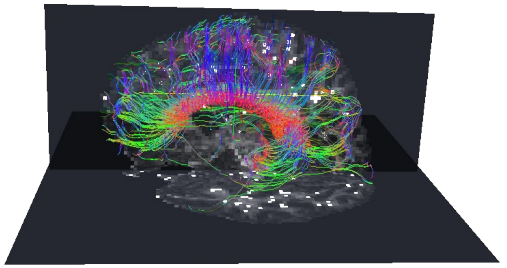
- https://www.openhub.net/p/vtk

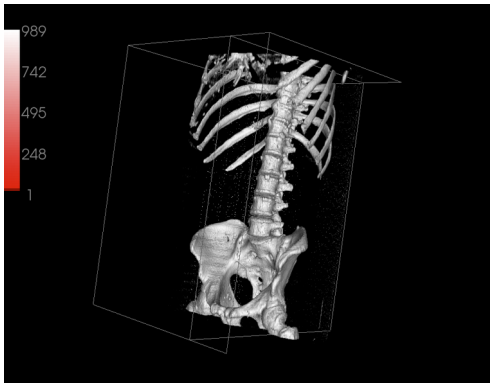
Introduction

- High level of abstraction
- Provides binding to another code languages:
  - Tcl/Tk
  - Python
  - Java
- GUI binding:
  - Qt
  - WxWidgets...

Introduction

- Examples of VTK applications:
  - Visualization
    - Scalar, vector and tensor fields
    - Volume data
  - Mesh and polygon processing
  - Image Analysis
  - Isosurface Extraction





Preparing Working Enviroment

[1] Install **Anaconda**

- Install **VTK**...

[2] Install **Eclipse IDE** and **PyDev**

[3] Prepare Project

Installing Anaconda

- **Anaconda** is like a ‘python distribution’
  - Contains the core python language and hundreds packages for data science and machine learning
- Follow the instructions in the anaconda user guide
  - In this case, we will use **Miniconda**

Installing Anaconda

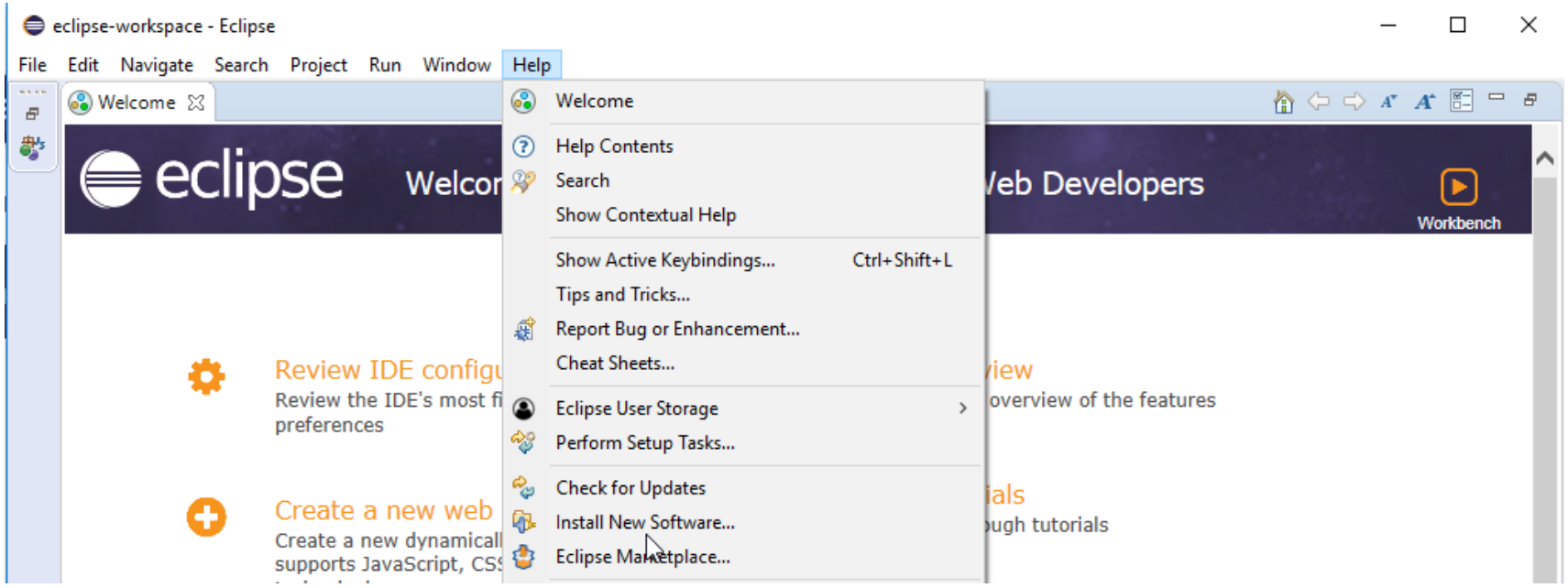
- Install the vtk package
  - Conda-forge repository contents VTK 8.1.0

- From **anaconda prompt:**

```
# conda install -c conda-forge vtk
```

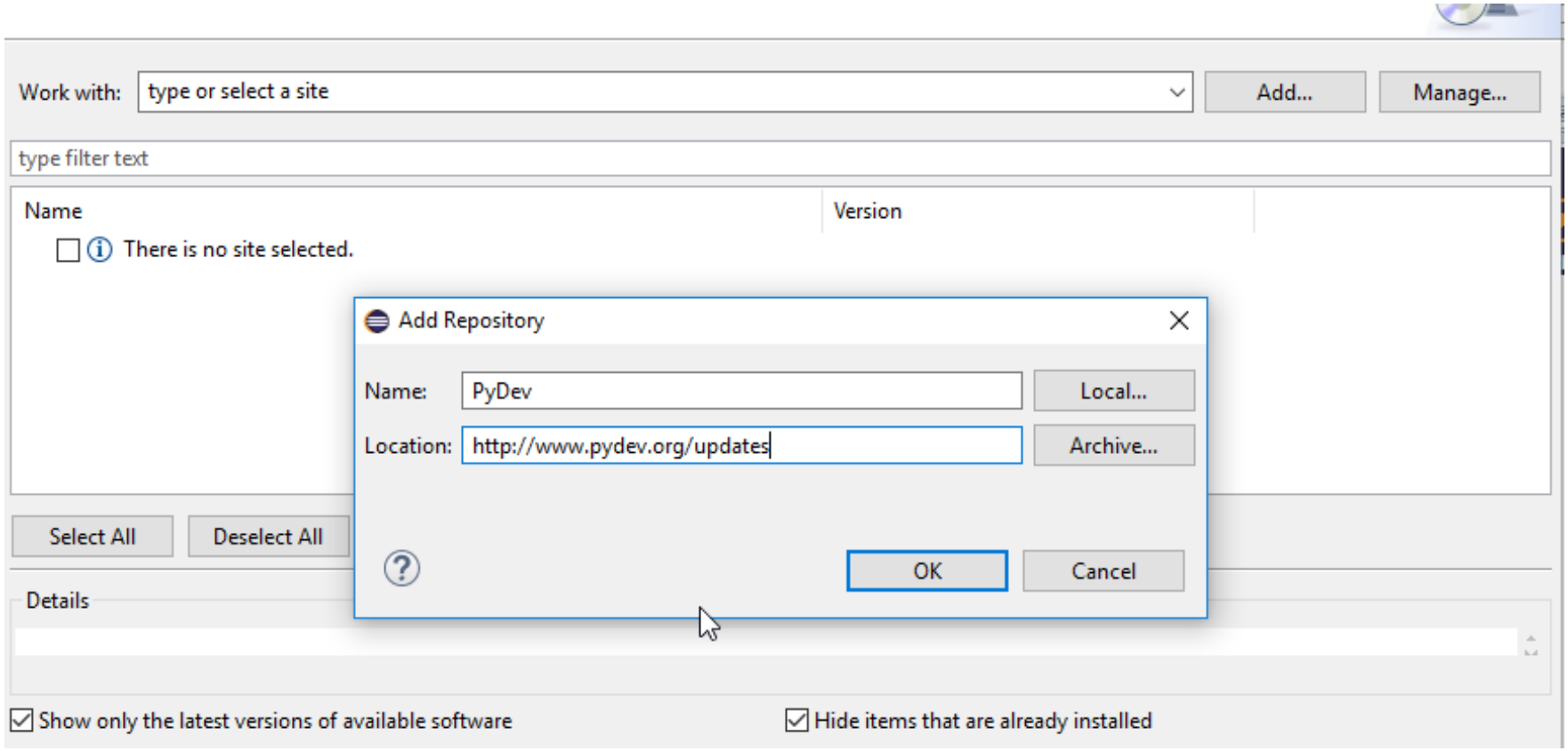
Installing IDE

- Download Eclipse IDE
  - It will ask for Java VM 1.7.0 or higher
- Install Eclipse PyDev plugin

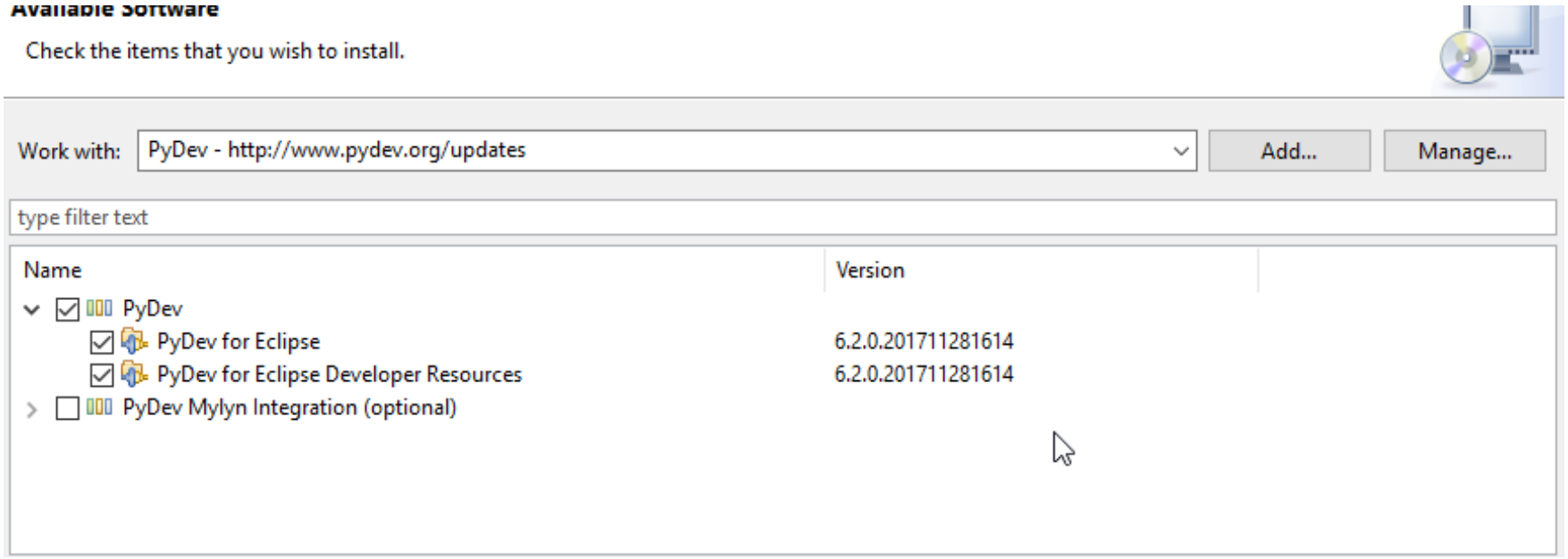


Installing PyDev

- Installing with the update site (http://www.pydev.org/updates)

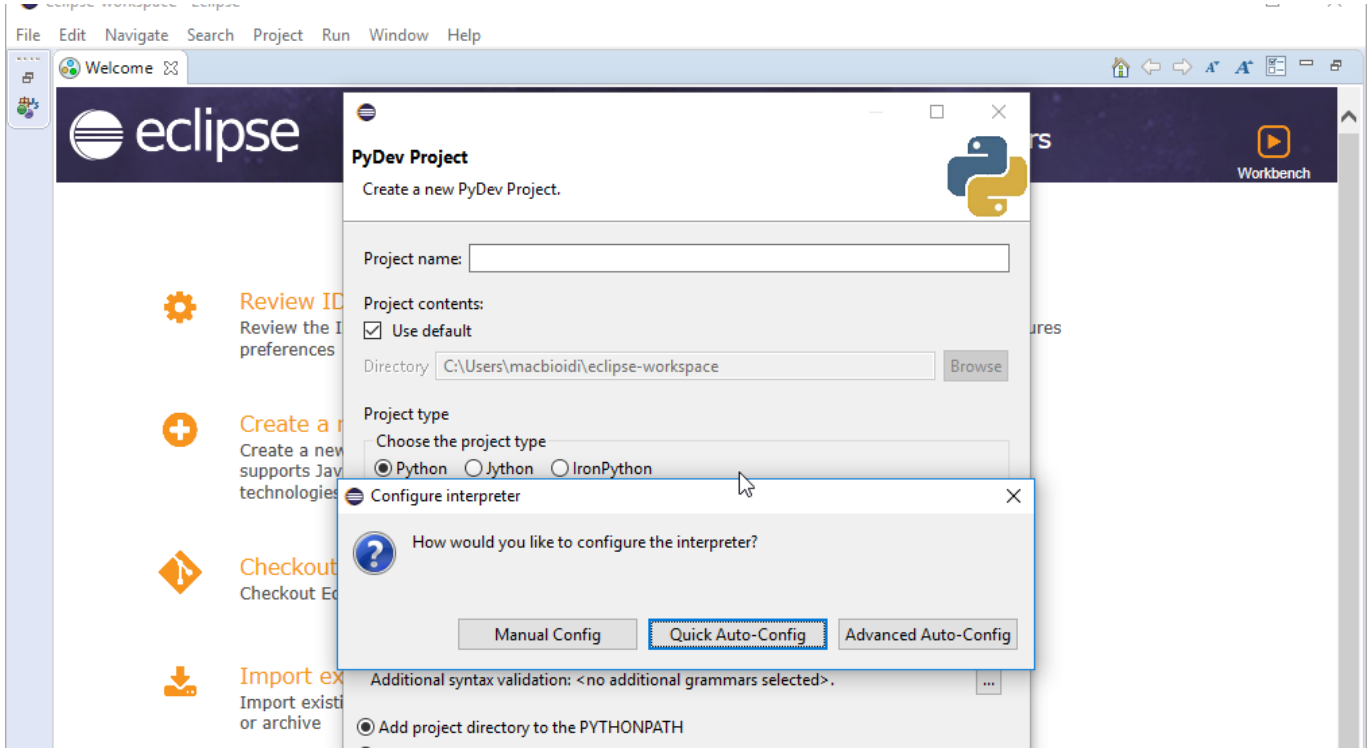


Installing PyDev



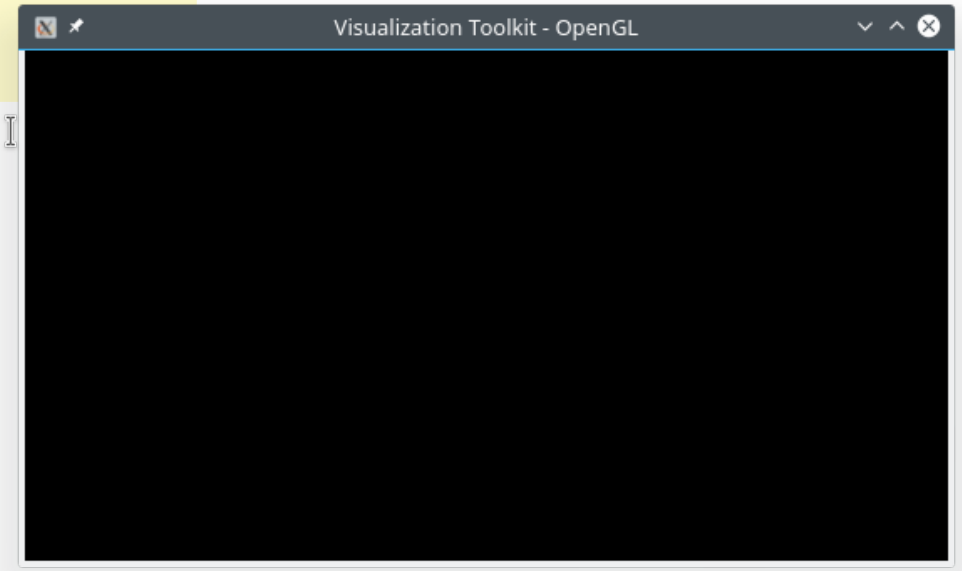
Prepare Interpreter

- Create a new PyDev Project
  - First time, you have to set the Python Interpreter (Quick Auto-config usually works)



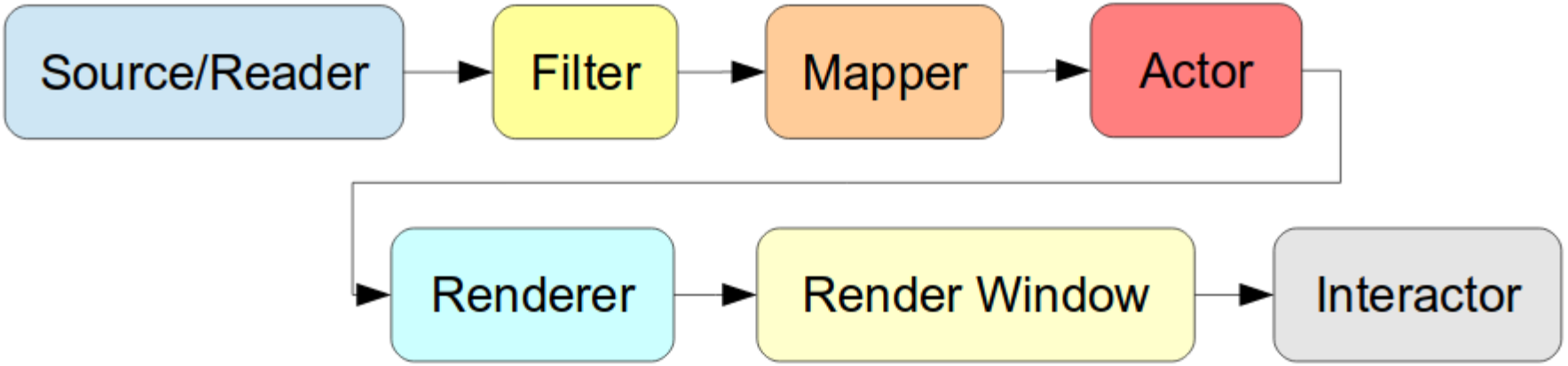
Example Code

```
import vtk

# Create an interactor
renderWindow = vtk.vtkRenderWindow()

# Create an interactor
interactor = vtk.vtkRenderWindowInteractor()
interactor.SetRenderWindow(renderWindow)

interactor.Initialize()
renderWindow.Render()
interactor.Start()
```



Visualization Pipeline

- To construct the geometric representation that is then rendered by the graphics pipeline
  - Transform informational data into graphical data

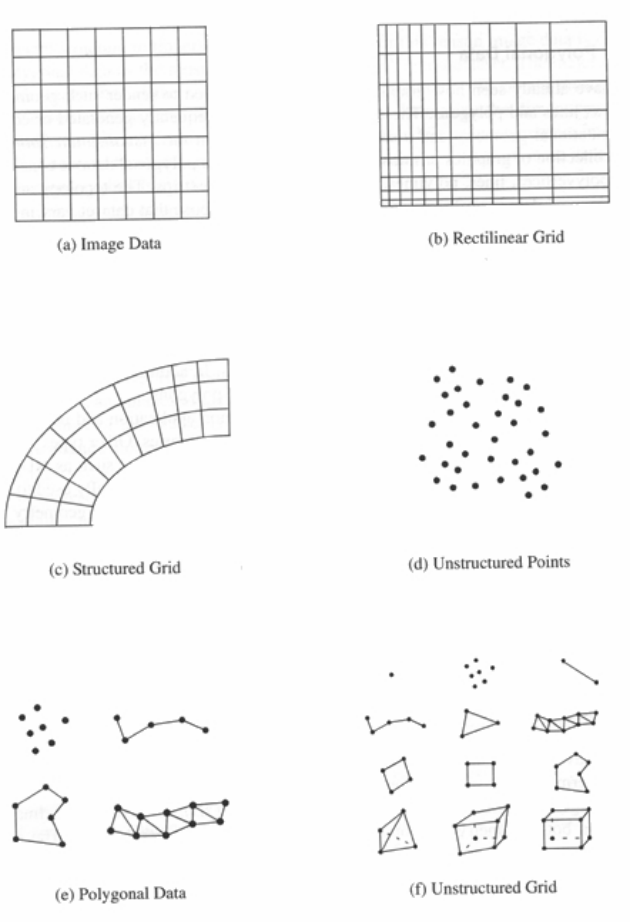


Sources/Readers

- First step where you will load the data
- You can use the various sources classes that VTK Provided
  - vtkSphereSource
  - vtkCubeSource
  - vtkConeSource
- You can read data from file too
  - Polydata Readers
  - StructuredGrid Readers...

Sources/Readers

- **VTK Data objects:**
  - vtkImageData
    - Image/Volume Data
  - vtkRectilinearGrid
    - Rectilinear Grid
  - vtkStructuredGrid
    - Structured Grid
  - vtkPolyData
    - Unstructured Points and Polygonal Data
  - vtkUnstructuredGrid
    - Unstructured Grid



Sources/Readers

- Create a cube from source class:

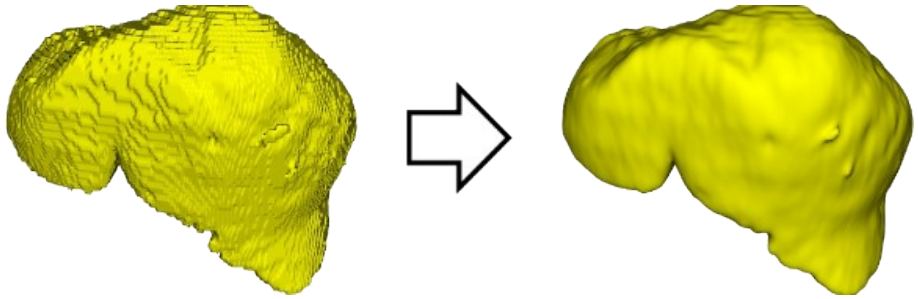
```
# Create an interactor
cube = vtk.vtkCubeSource()
```

- This class corresponds to a polygon data
  - vtkPolyData class
    - Represents a geometric structure consisting of vertices, lines, polygons, and/or triangle strips.
- To access vtkPolyData instance:

```
polyData = cube.GetOutput()
```

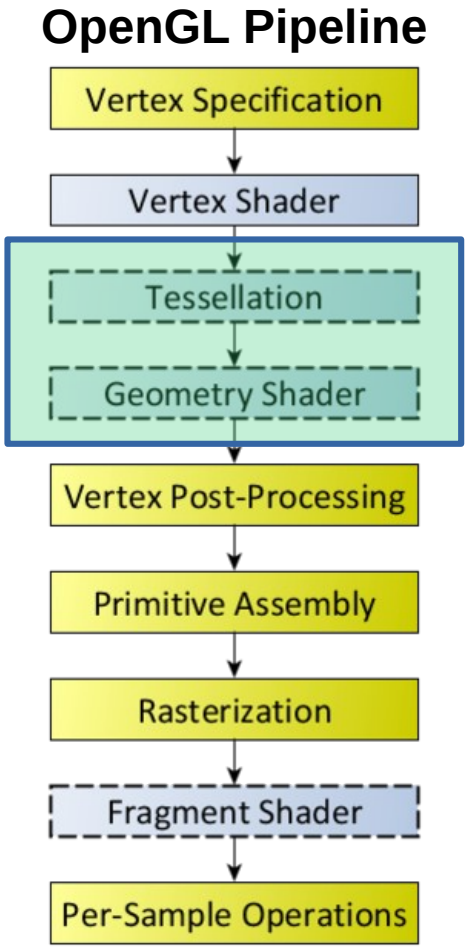
Filters

- Takes data and modify it, returning the modified data
  - Can be used to (for example):
    - Select data of a particular size, strength, intensity…
    - Process 2D/3D images or polygon meshes
    - Generate geometric objects from data



Filters

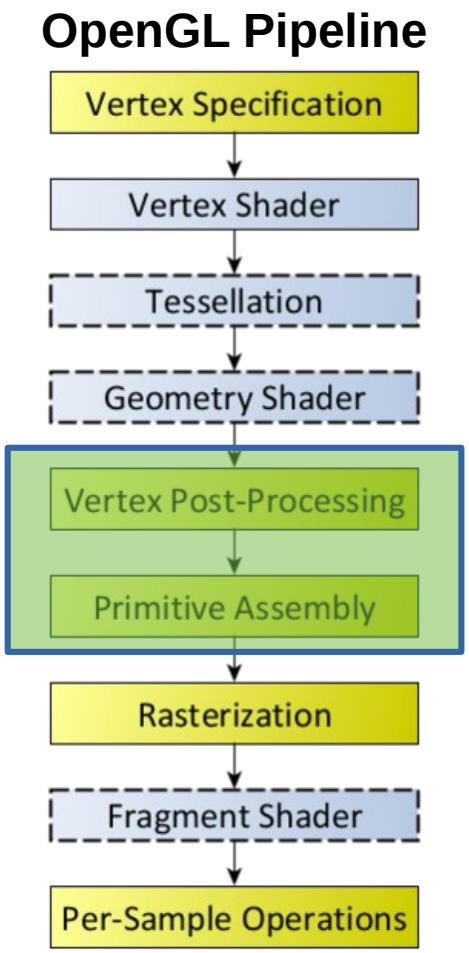
- Filters represents the first steps in the OpenGL pipeline.
- Tessellation step is useless in VTK enviroment.
  - Textures are not usual in VTK applications



Mappers

- Maps data to graphics primitives that can be displayed by the renderer
  - Interface between data and graphics primitives
- Multiple mappers may share the same input, but render it in different ways
- The mapper more usual labs is **vtkPolyDataMapper**

Mappers


OpenGL Pipeline
Vertex Specification
Vertex Shader
Tessellation
Geometry Shader
Vertex Post-Processing
Primitive Assembly
Rasterization
Fragment Shader
Per-Sample Operations

Mapper

- Create a mapper for the cube data

```
cubeMapper = vtk.vtkPolyDataMapper()
```

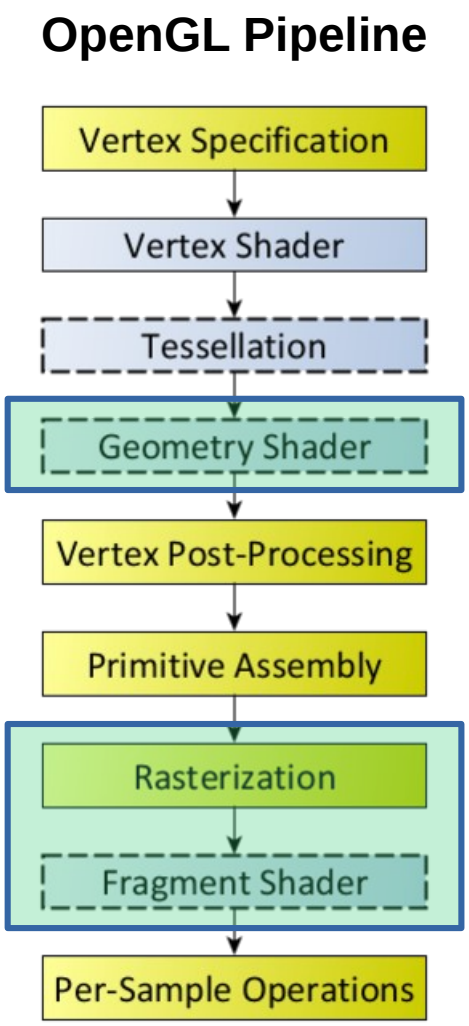
- Set connection between Sources and Mapper steps
  - In previous VTK releases, vtkPolyDataMapper::SetInput() stablished the connection
  - In current VTK release, connection is stablished using vtkPolyDataMapper::SetInputConnection()

```
cubeMapper.SetInputConnection(cube.GetOutputPort())
```

Actor

- **vtkActor** represents an object in a rendering scene
  - Uses the output of a mapper and ‘know’ how to generate the visible representation data
- Rendering produced is generated by the properties configuration:
  - Scale
  - Orientation
  - Textures...

Actor


OpenGL Pipeline
Vertex Specification
Vertex Shader
Tessellation
Geometry Shader
Vertex Post-Processing
Primitive Assembly
Rasterization
Fragment Shader
Per-Sample Operations

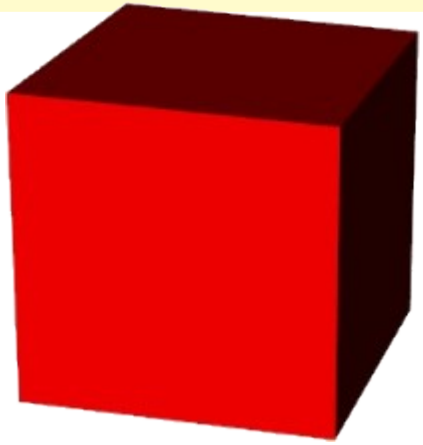
Actor

- Create a actor for the cube data mapper

```
cubeActor = vtk.vtkActor()
cubeActor.SetMapper(cubeMapper)
```

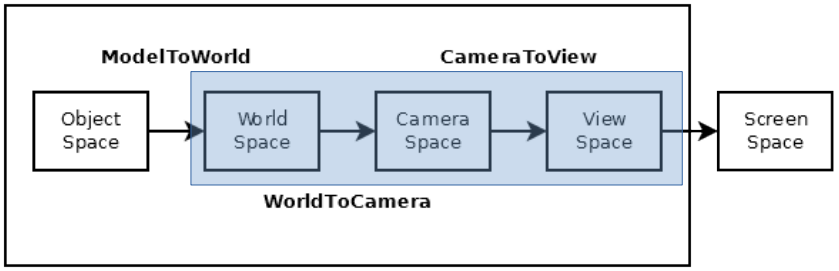
- Here, you can manipulate the differentes properties of your ‘object’:

```
# make the cube red
cubeActor.GetProperty().SetColor(1.0, 0.0, 0.0)
```



Renderer

- vtkRenderer controls the rendering process for actors and scenes
  - Converts the 3D graphics primitives into an 2D image
- Performs coordinate transformation between multiple coordinates:
  - world coordinates
  - view coordinates
  - camera coordinates



Renderer

- Create a vtkRenderer and connect it with the cube actor

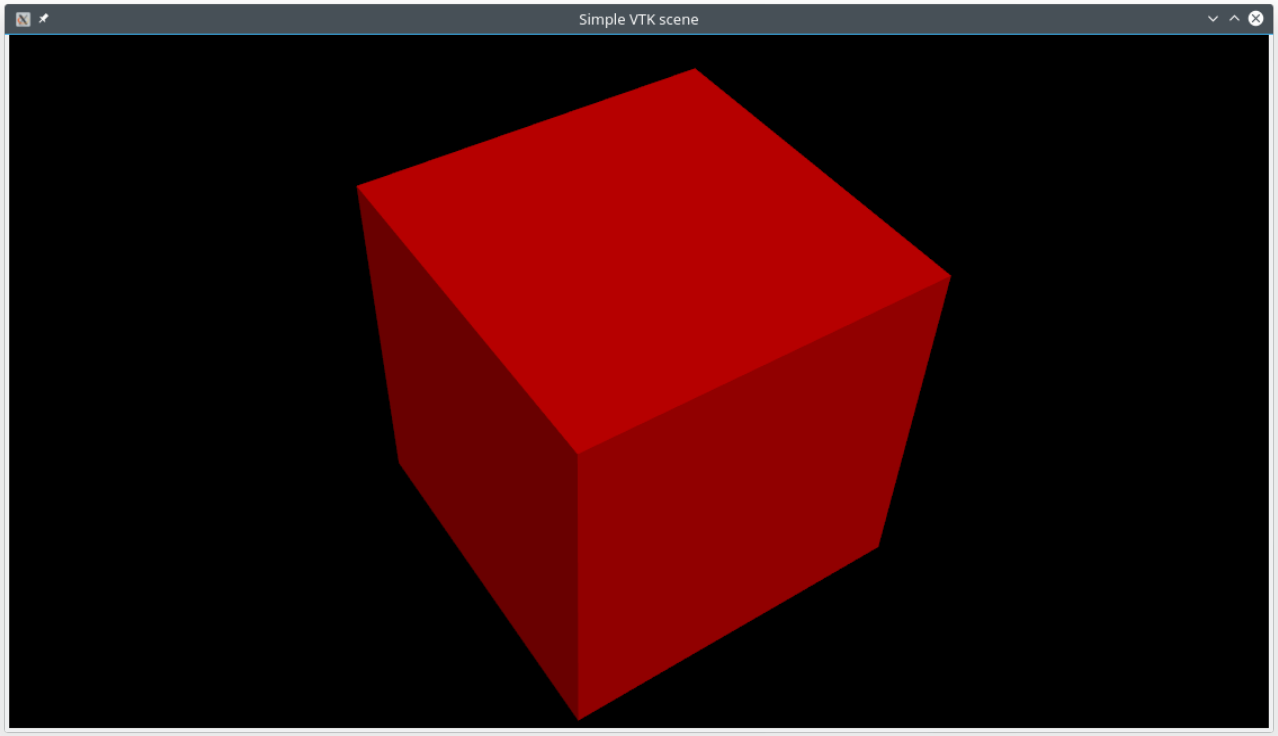
```
# Create a renderer and add the cube actor to it
renderer = vtk.vtkRenderer()
renderer.AddActor(cubeActor)
```

- Create a vtkRenderer and connect it with the cube actor

```
# make the background black
renderer.SetBackground(0.0, 0.0, 0.0)
```

Render Window

- The vtkRenderWindow class creates a window for renderers to draw into




```
renderWindow = vtk.vtkRenderWindow()
renderWindow.AddRenderer(renderer)
```

Interactors

- The vtkRenderWindowInteractor class provides platform-independent window interaction via the mouse and keyboard
- Allows you to rotate/zoom/pan the camera, select and manipulate actors, etc
- Also handles time events

Exercise

- Extract isosurfaces of the probability density of a hydrogen atom volume dataset
  - Values have to be color-mapped according to the isovalue (remember to show a colorbar)
  - Create a small keyboard interface for changing the isovalue
- Tasks:
  - Use **vtkContourFilter** in order to extract probability isosurfaces
  - Colormap the surfaces using **vtkColorTransferFunction**
  - Add colormap, **vtkScalarBarActor**
  - Add keyboard interaction

Introduction to VTK

MACbioIDi – February – March 2018

ULPGC
Tecnología Médica para el Desarrollo Sostenible
MAC 2014-2020 Cooperación Territorial
Interreg
Fondo Europeo de Desarrollo Regional
EUROPEAN UNION