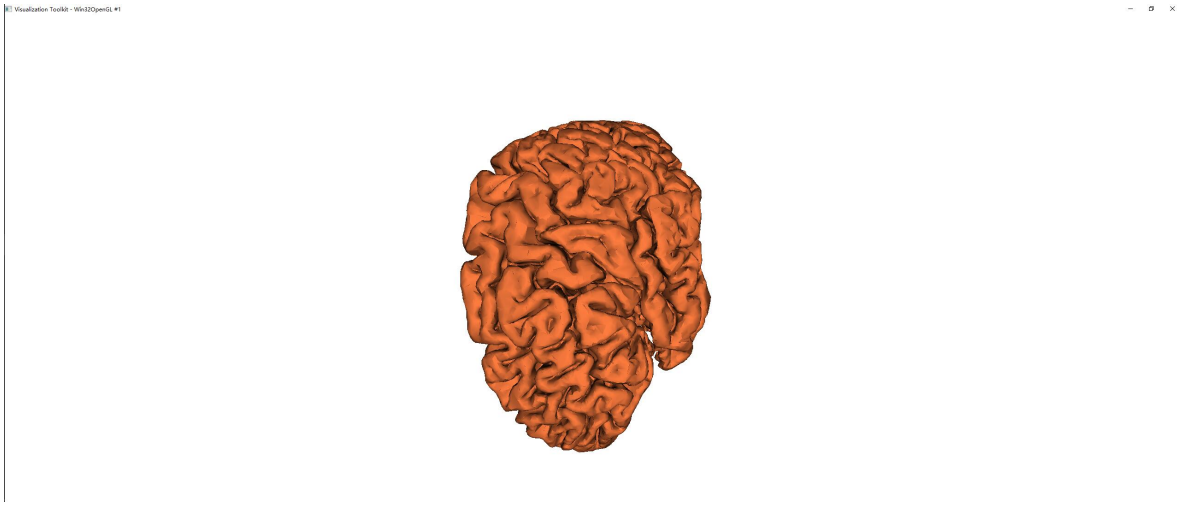
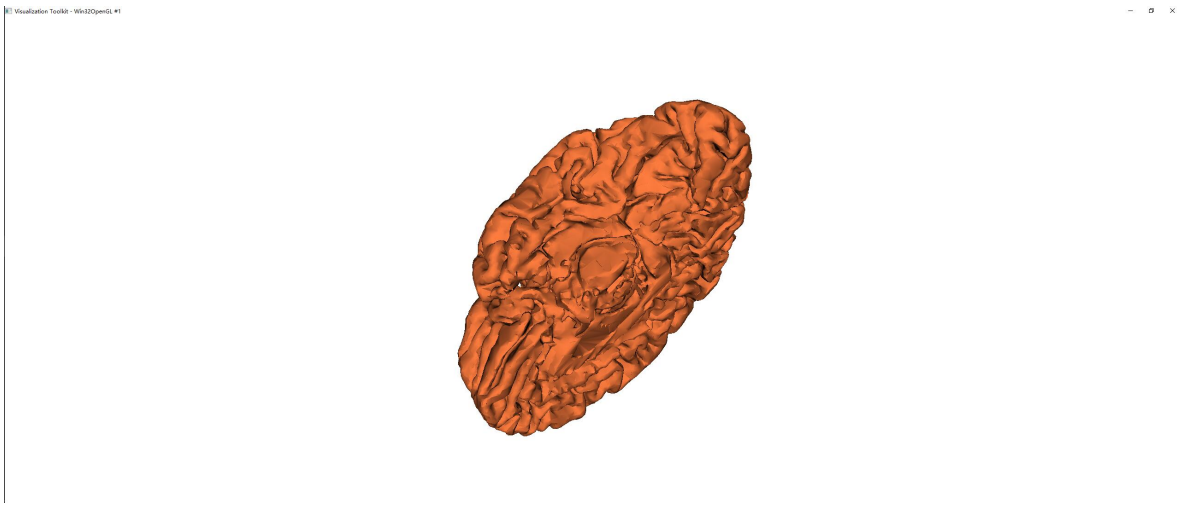
Modify path

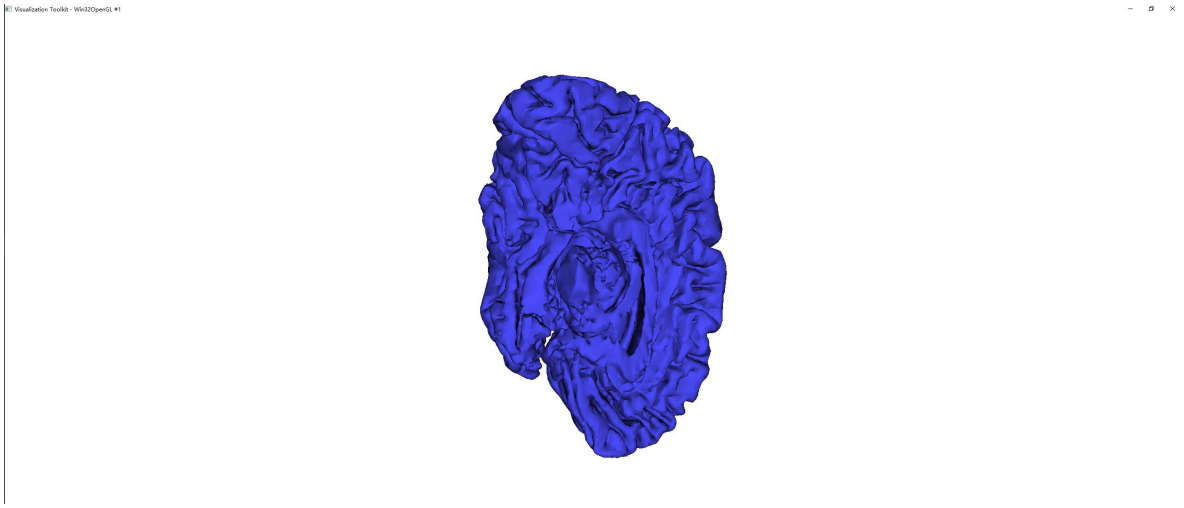
```
vtkPolyDataReader fran
    fran SetFileName "Cort_lobe_poly.vtk"
```

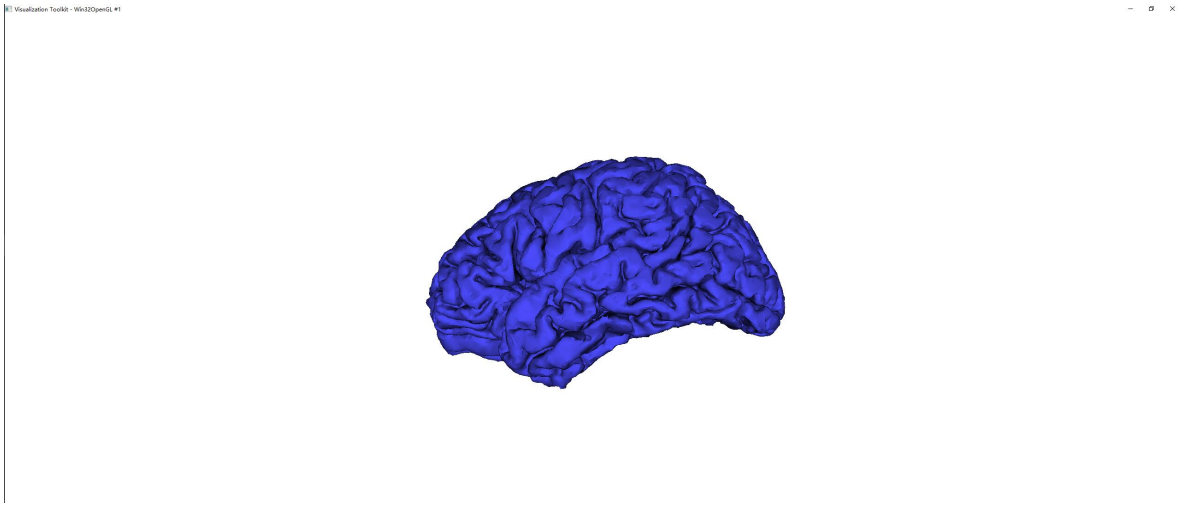




Change color

```
vtkActor franActor
    franActor SetMapper franMapper
    eval [franActor GetProperty] SetColor 0.2 0.2 1.0
```

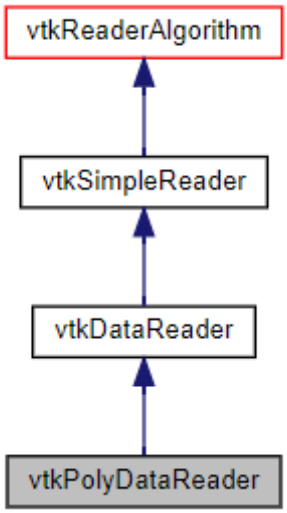




The Visualization Toolkit (VTK) is open source software for manipulating and displaying scientific data. It comes with state-of-the-art tools for 3D rendering, a suite of widgets for 3D interaction, and extensive 2D plotting capability.

```
# We start by reading some data that was originally captured from
# a Cyberware laser digitizing system.
#
vtkPolyDataReader fran
    fran SetFileName "Cort_lobe_poly.vtk"
```

vtkPolyDataReader is a source object that reads ASCII or binary polygonal data files in vtk format (see text for format details). The output of this reader is a single **vtkPolyData** data object. The superclass of this class, **vtkDataReader**, provides many methods for controlling the reading of the data file.




```
# We want to preserve topology (not let any cracks form). This may limit
# the total reduction possible, which we have specified at 90%.
#
vtkDecimatePro deci
    deci SetInput [fran GetOutput]
    deci SetTargetReduction 0.9
    deci PreserveTopologyOn
vtkPolyDataNormals normals
    normals SetInput [deci GetOutput]
    normals FlipNormalsOn
vtkPolyDataMapper franMapper
    franMapper SetInput [normals GetOutput]
vtkActor franActor
    franActor SetMapper franMapper
    eval [franActor GetProperty] SetColor 0.2 0.2 1.0
```

vtkDecimatePro is a filter to reduce the number of triangles in a triangle mesh, forming a good approximation to the original geometry. The input to **vtkDecimatePro** is a **vtkPolyData** object, and only triangles are treated. If we desire to decimate polygonal meshes, first triangulate the polygons with **vtkTriangleFilter** object.

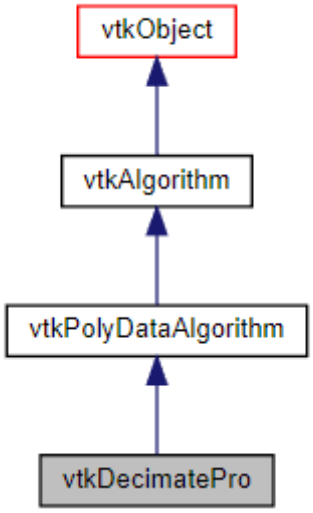
virtual void vtkDecimatePro::SetTargetReduction (double)

Specify the desired reduction in the total number of polygons (e.g., if TargetReduction is set to 0.9, this filter will try to reduce the data set to 10% of its original size).

virtual void vtkDecimatePro::PreserveTopologyOn()*

Turn on/off whether to preserve the topology of the original mesh.

If on, mesh splitting and hole elimination will not occur. This may limit the maximum reduction that may be achieved.

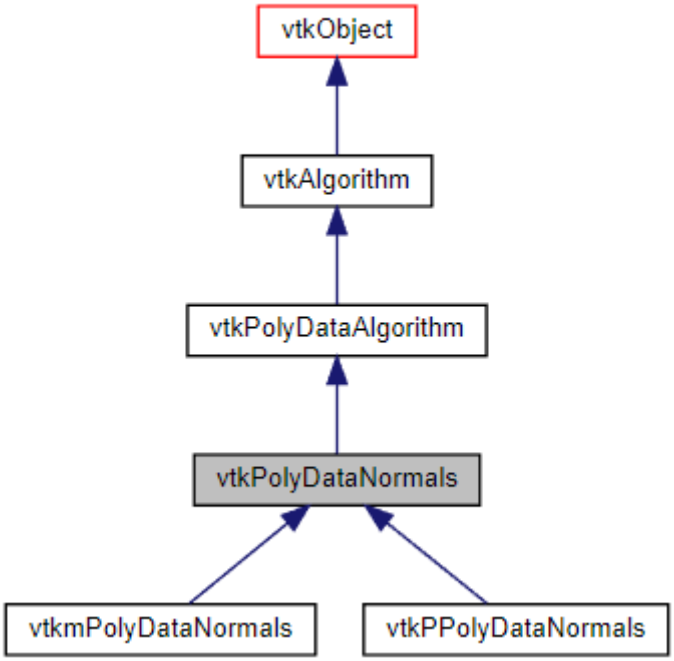


vtkPolyDataNormals is a filter that computes point and/or cell normals for a polygonal mesh. The user specifies if they would like the point and/or cell normals to be computed by setting the ComputeCellNormals and ComputePointNormals flags.

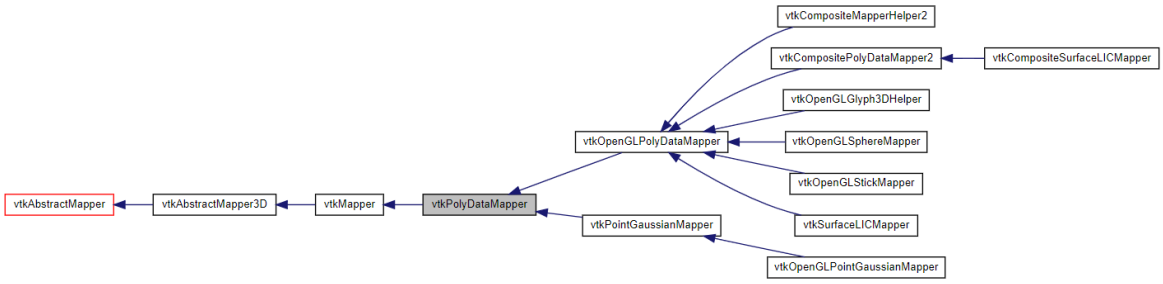
virtual void vtkPolyDataNormals::FlipNormalsOn()

Turn on/off the global flipping of normal orientation.

Flipping reverves the meaning of front and back for Frontface and Backface culling in vtkProperty. Flipping modifies both the normal direction and the order of a cell's points.



vtkPolyDataMapper is a class that maps polygonal data (i.e., **vtkPolyData**) to graphics primitives. **vtkPolyDataMapper** serves as a superclass for device-specific poly data mappers, that actually do the mapping to the rendering/graphics hardware/software.

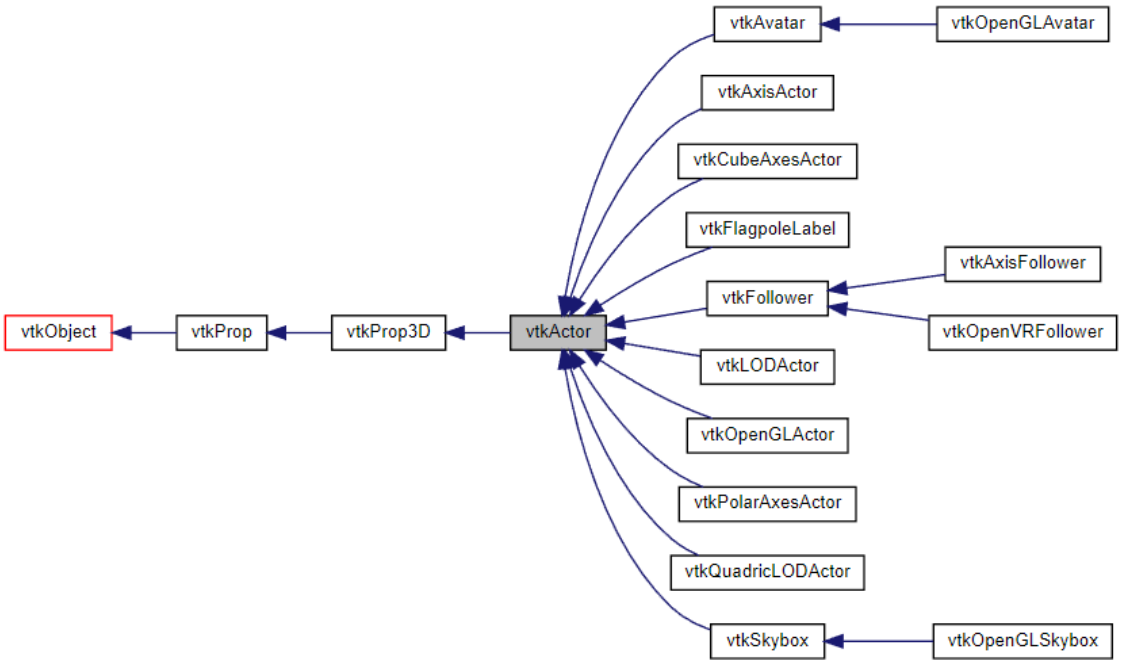


vtkActor is used to represent an entity in a rendering scene. It inherits functions related to the actors position, and orientation from **vtkProp**. The actor also has scaling and maintains a reference to the defining geometry (i.e., the mapper), rendering properties, and possibly a texture map. **vtkActor** combines these instance variables into one 4x4 transformation matrix as follows: [x y z 1] = [x y z 1] Translate(-origin) Scale(scale) Rot(y) Rot(x) Rot (z) Trans(origin) Trans(position)

virtual void vtkActor::SetMapper (vtkMapper *)

This is the method that is used to connect an actor to the end of a visualization pipeline, i.e.

the mapper. This should be a subclass of vtkMapper. Typically vtkPolyDataMapper and vtkDataSetMapper will be used.




```
# Create the RenderWindow, Renderer and both Actors
#
vtkRenderer ren1
vtkRenderWindow renWin
    renWin AddRenderer ren1
vtkRenderWindowInteractor iren
    iren SetRenderWindow renWin

# Add the actors to the renderer, set the background and size
#
ren1 AddActor franActor
ren1 SetBackground 1 1 1
renWin SetSize 250 250
```

vtkRenderer provides an abstract specification for renderers. A renderer is an object that controls the rendering process for objects. Rendering is the process of converting geometry, a specification for lights, and a camera view into an image. **vtkRenderer** also performs coordinate transformation between world coordinates, view coordinates (the computer graphics rendering coordinate system), and display coordinates (the actual screen coordinates on the display device). Certain advanced rendering features such as two-sided lighting can also be controlled.

void vtkRenderer::AddActor (vtkProp * p)

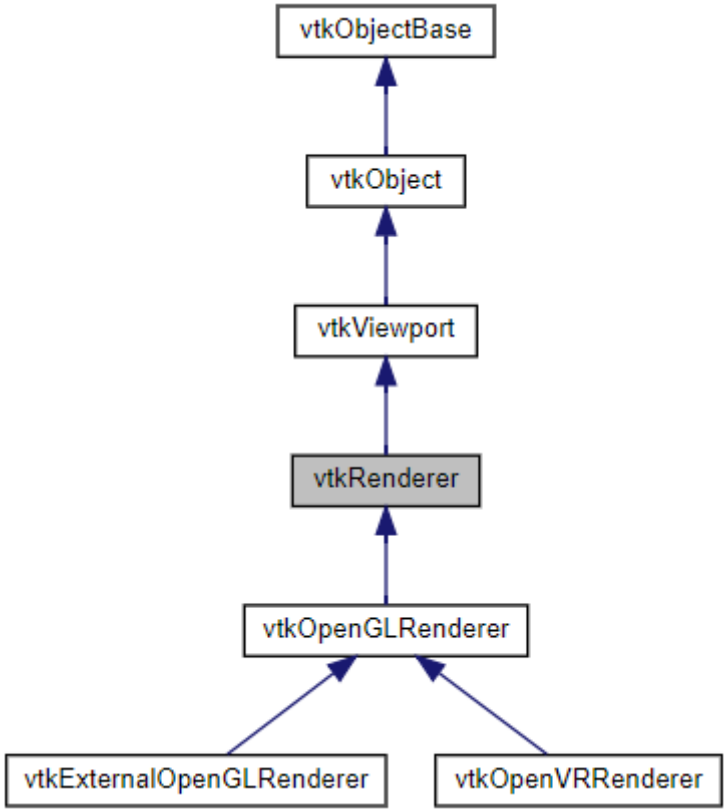
Add/Remove different types of props to the renderer.

These methods are all synonyms to AddViewProp and RemoveViewProp. They are here for convenience and backwards compatibility.

virtual void vtkRenderer::SetBackgroundTexture (vtkTexture *)

Set/Get the texture to be used for the monocular or stereo left eye background.

If set and enabled this gets the priority over the gradient background.

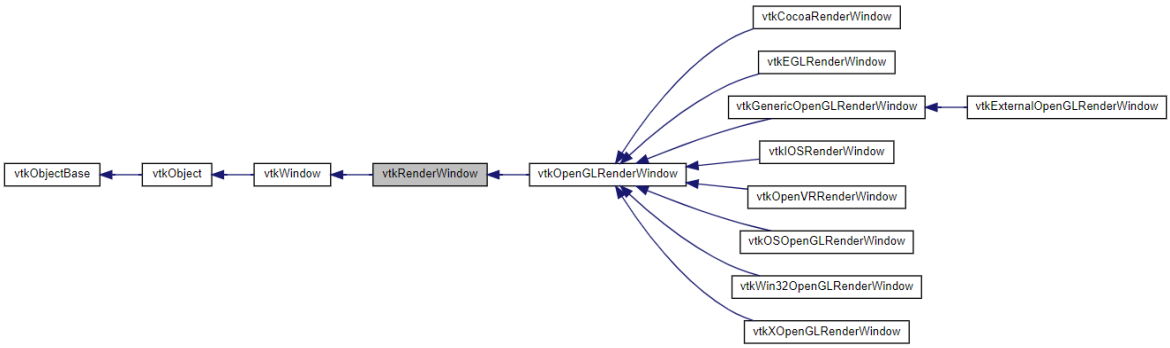


vtkRenderWindow is an abstract object to specify the behavior of a rendering window. A rendering window is a window in a graphical user interface where renderers draw their images. Methods are provided to synchronize the rendering process, set window size, and control double buffering. The window also allows rendering in stereo. The interlaced render stereo type is for output to a VRex stereo projector. All of the odd horizontal lines are from the left eye, and the even lines are from the right eye. The user has to make the render window aligned with the VRex projector, or the eye will be swapped.

virtual void vtkRenderWindow::AddRenderer (vtkRenderer *)

Add a renderer to the list of renderers.

Reimplemented in **vtkOpenVRRenderWindow**.



vtkRenderWindowInteractor provides a platform-independent interaction mechanism for mouse/key/time events. It serves as a base class for platform-dependent implementations that handle routing of mouse/key/timer messages to **vtkInteractorObserver** and its subclasses. **vtkRenderWindowInteractor** also provides controls for picking, rendering frame rate, and headlights.

void vtkRenderWindowInteractor::SetRenderWindow (vtkRenderWindow * aren)

Set/Get the rendering window being controlled by this object.