TVTK的管线

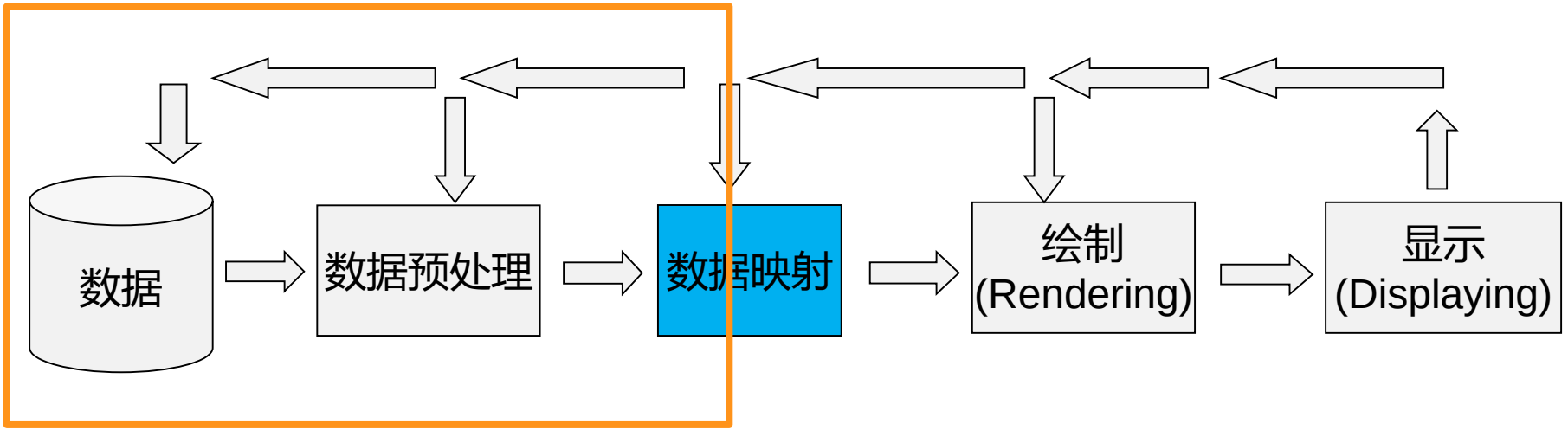
理解TVTK的管线

管线技术（Pipeline，流水线）

可视化管线(Visualization Pipeline)：将原始数据加工成图形数据的过程。

图形管线(Graphics Pipeline)：图形数据加工为我们所看到的图像的过程。

理解TVTK的管线



可视化管线

可视化管线

TVTK对象	说明
CubeSource	通过程序内部计算输出一组描述长方体的数据(PolyData)
PolyDataMapper	PolyData通过该映射器将数据映射为图形数据（mapper）

可视化管线

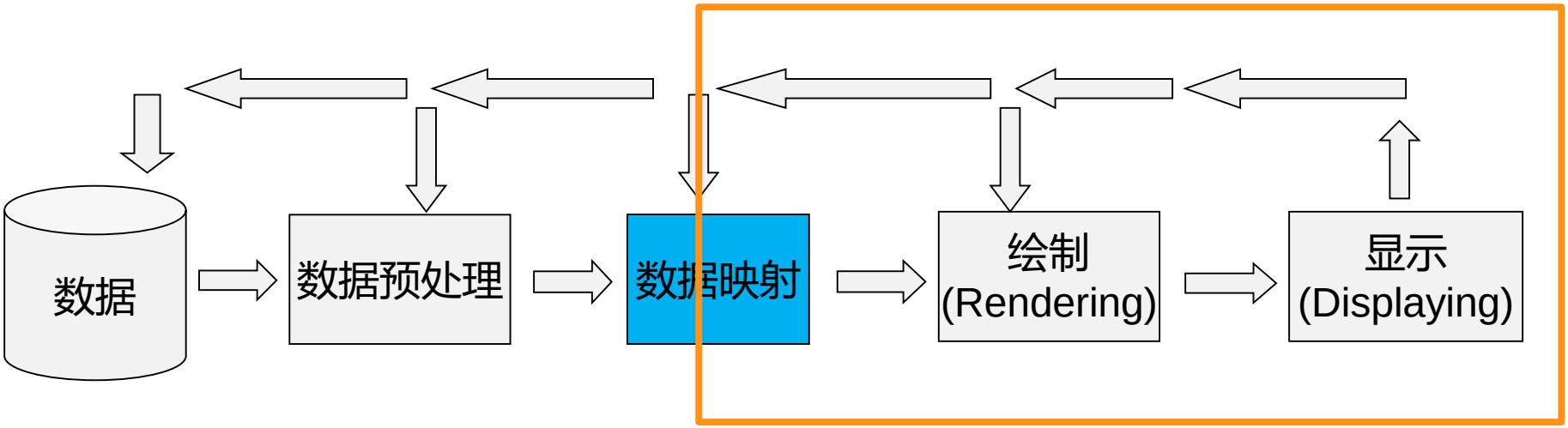
```
from tvtk.api import tvtk

# 创建一个长方体数据源，并且同时设置其长宽高
s = tvtk.CubeSource(x_length=1.0, y_length=2.0, z_length=3.0)
# 使用PolyDataMapper将数据转换为图形数据
m = tvtk.PolyDataMapper(input_connection=s.output_port)

a = tvtk.Actor(mapper=m)
r = tvtk.Renderer(background=(0, 0, 0))
r.add_actor(a)
w = tvtk.RenderWindow(size=(300,300))
w.add_renderer(r)
i = tvtk.RenderWindowInteractor(render_window=w)
i.initialize()
i.start()
```

可视化管线

```
>>> s.output_port
<tvtk.tvtk_classes.algorithm_output.AlgorithmOutput object at 0x0000020E141F0990>
>>> m.input_connection
<tvtk.tvtk_classes.algorithm_output.AlgorithmOutput object at 0x0000020E141F0990>
>>>
```

对象经由input_connection和output_port属性连接起来

理解TVTK的管线



图形管线

图形管线

TVTK对象	说明
Actor	场景中的一个实体。它包括一个图形数据(mapper)，具有描述该实体的位置、方向、大小的属性。
Renderer	渲染的场景。它包括多个需要渲染的Actor。
RenderWindow	渲染用的图形窗口，它包括一个或者多个Render。
RenderWindowInteractor	给图形窗口提供一些用户交互功能，例如平移、旋转、放大缩小。这些交互式操作并不改变Actor或者图形数据的属性，只是调整场景中的照相机(Camera)的一些设置。

图形管线

```
from tvtk.api import tvtk

# 创建一个长方体数据源，并且同时设置其长宽高
s = tvtk.CubeSource(x_length=1.0, y_length=2.0, z_length=3.0)
# 使用PolyDataMapper将数据转换为图形数据
m = tvtk.PolyDataMapper(input_connection=s.output_port)
# 创建一个Actor
a = tvtk.Actor(mapper=m)
# 创建一个Renderer，将Actor添加进去
r = tvtk.Renderer(background=(0, 0, 0))
r.add_actor(a)
# 创建一个RenderWindow(窗口)，将Renderer添加进去
w = tvtk.RenderWindow(size=(300,300))
w.add_renderer(r)
# 创建一个RenderWindowInteractor（窗口的交互工具）
i = tvtk.RenderWindowInteractor(render_window=w)
# 开启交互
i.initialize()
i.start()
```

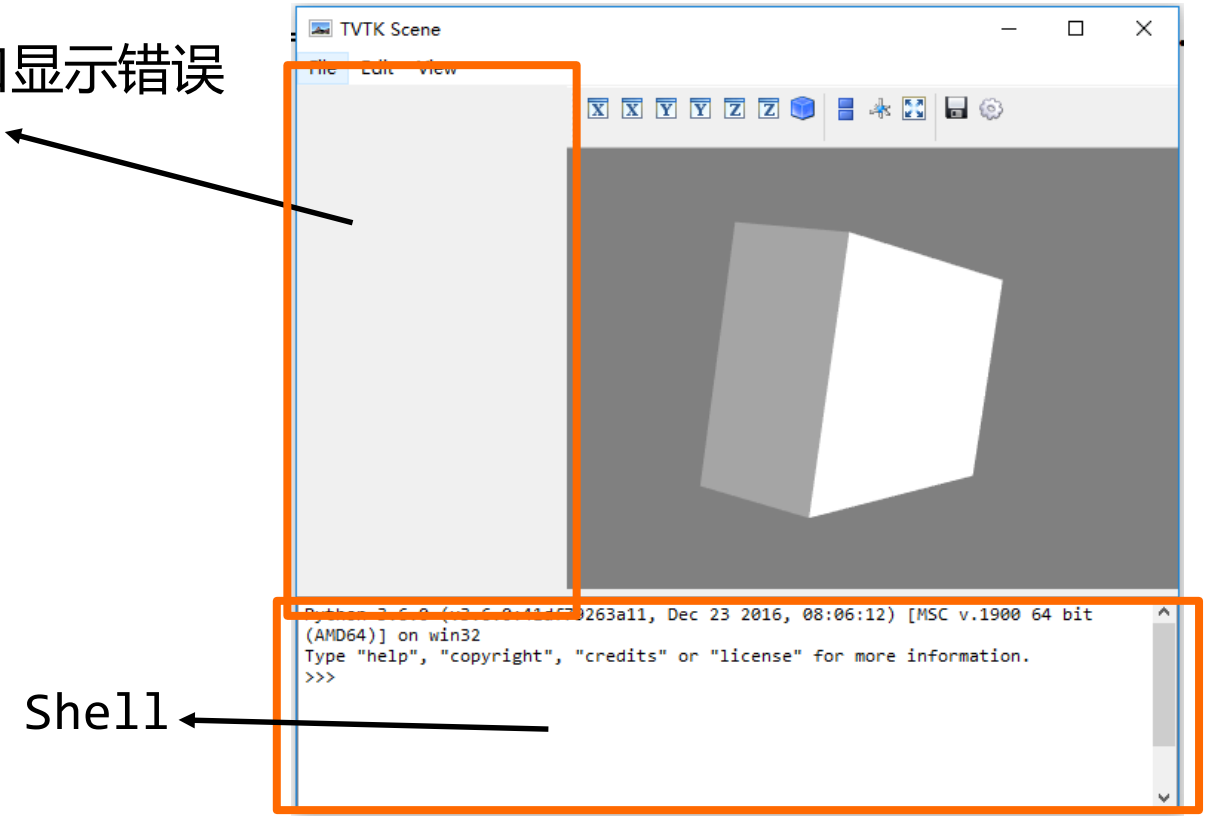
用ivtk工具观察管线

```
from tvtk.api import ivtk
```

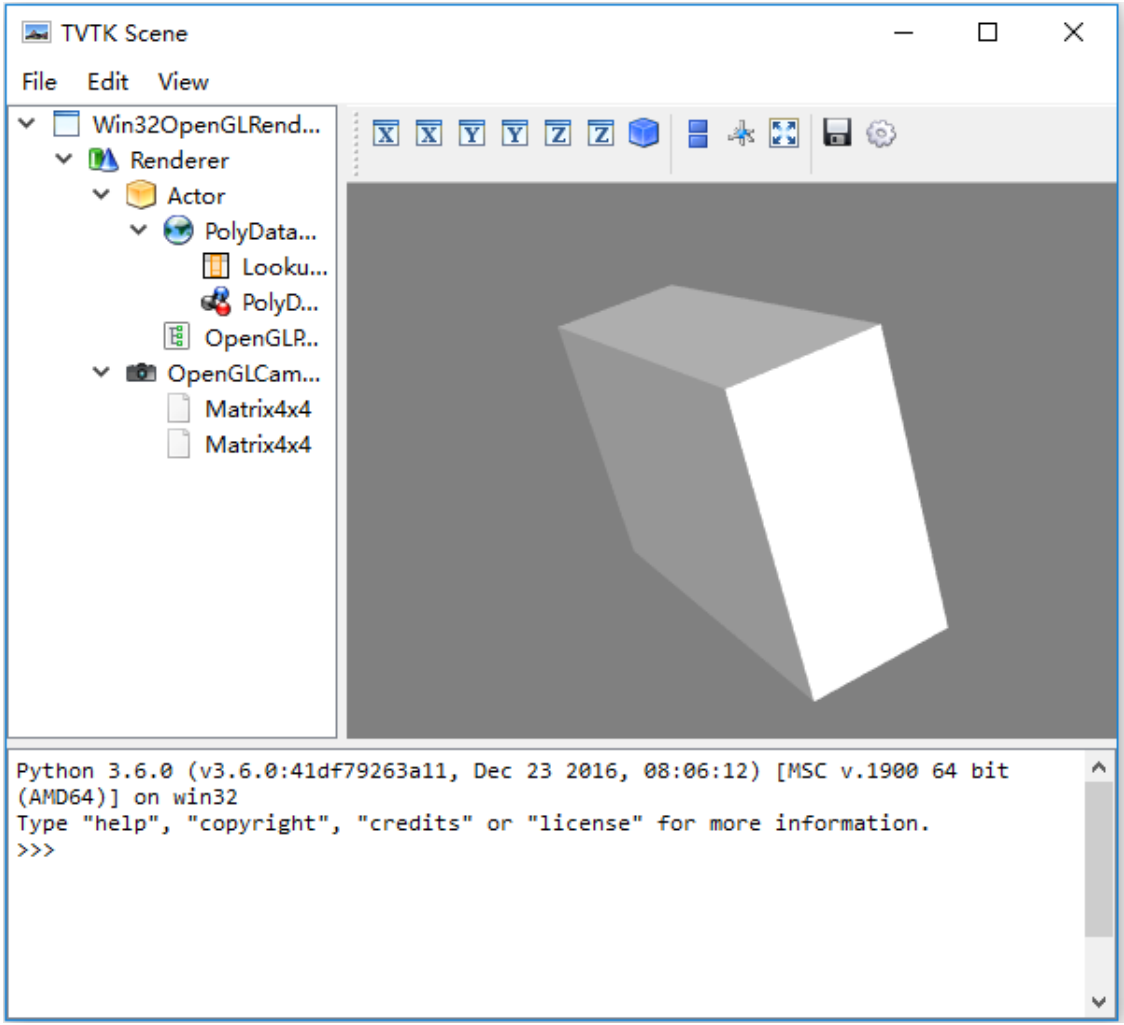
使用ivtk显示立方体的程序

```
from tvtk.api import tvtk
from tvtk.tools import ivtk
from pyface.api import GUI

s = tvtk.CubeSource(x_length=1.0, y_length=2.0, z_length=3.0)
m = tvtk.PolyDataMapper(input_connection=s.output_port)
a = tvtk.Actor(mapper=m)

#创建一个带Crust（Python Shell）的窗口
gui = GUI()
win = ivtk.IVTKWithCrustAndBrowser()
win.open()
win.scene.add_actor(a)

#开始界面消息循环
gui.start_event_loop()
```

修正窗口显示错误


TVTK Scene
File Edit View
(AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
Shell

```
from tvtk.api import tvtk
from tvtk.tools import ivtk
from pyface.api import GUI

s = tvtk.CubeSource(x_length=1.0, y_length=2.0, z_length=3.0)
m = tvtk.PolyDataMapper(input_connection=s.output_port)
a = tvtk.Actor(mapper=m)

#创建一个带Crust（Python Shell）的窗口
gui = GUI()
win = ivtk.IVTKWithCrustAndBrowser()
win.open()
win.scene.add_actor(a)

#修正错误
dialog = win.control.centralWidget().widget(0).widget(0)
from pyface.qt import QtCore
dialog.setWindowFlags(QtCore.Qt.WindowFlags(0x00000000))
dialog.show()

#开始界面消息循环
gui.start_event_loop()
```


TVTK Scene
File
Edit
View
Win32OpenGLRend...
Renderer
Actor
PolyData...
Looku...
PolyD...
OpenGLP...
OpenGLCam...
Matrix4x4
Matrix4x4
Python 3.6.0 (v3.6.0:41df79263a11, Dec 23 2016, 08:06:12) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>

```
from tvtk.api import tvtk

def ivtk_scene(actors):
    from tvtk.tools import ivtk
    #创建一个带Crust（Python Shell）的窗口
    win = ivtk.IVTKWithCrustAndBrowser()
    win.open()
    win.scene.add_actor(actors)
    #修正窗口错误
    dialog = win.control.centralWidget().widget(0).widget(0)
    from pyface.qt import QtCore
    dialog.setWindowFlags(QtCore.Qt.WindowFlags(0x00000000))
    dialog.show()
    return win

def event_loop():
    from pyface.api import GUI
    gui = GUI()
    gui.start_event_loop()

s = tvtk.CubeSource(x_length=1.0, y_length=2.0, z_length=3.0)
m = tvtk.PolyDataMapper(input_connection=s.output_port)
a = tvtk.Actor(mapper=m)
win = ivtk_scene(a)
win.scene.isometric_view()
event_loop()
```

Cube_ivtk_func.py

```python
from tvtk.api import tvtk

def ivtk_scene(actors):
    from tvtk.tools import ivtk
    #创建一个带Crust（Python Shell）的窗口
    win = ivtk.IVTKWithCrustAndBrowser()
    win.open()
    win.scene.add_actor(actors)
    #修正窗口错误
    dialog = win.control.centralWidget().widget(0).widget(0)
    from pyface.qt import QtCore
    dialog.setWindowFlags(QtCore.Qt.WindowFlags(0x00000000))
    dialog.show()
    return win

def event_loop():
    from pyface.api import GUI
    gui = GUI()
    gui.start_event_loop()

s = tvtk.CubeSource(x_length=1.0, y_length=2.0, z_length=3.0)
m = tvtk.PolyDataMapper(input_connection=s.output_port)
a = tvtk.Actor(mapper=m)
win = ivtk_scene(a)
win.scene.isometric_view()
event_loop()
```

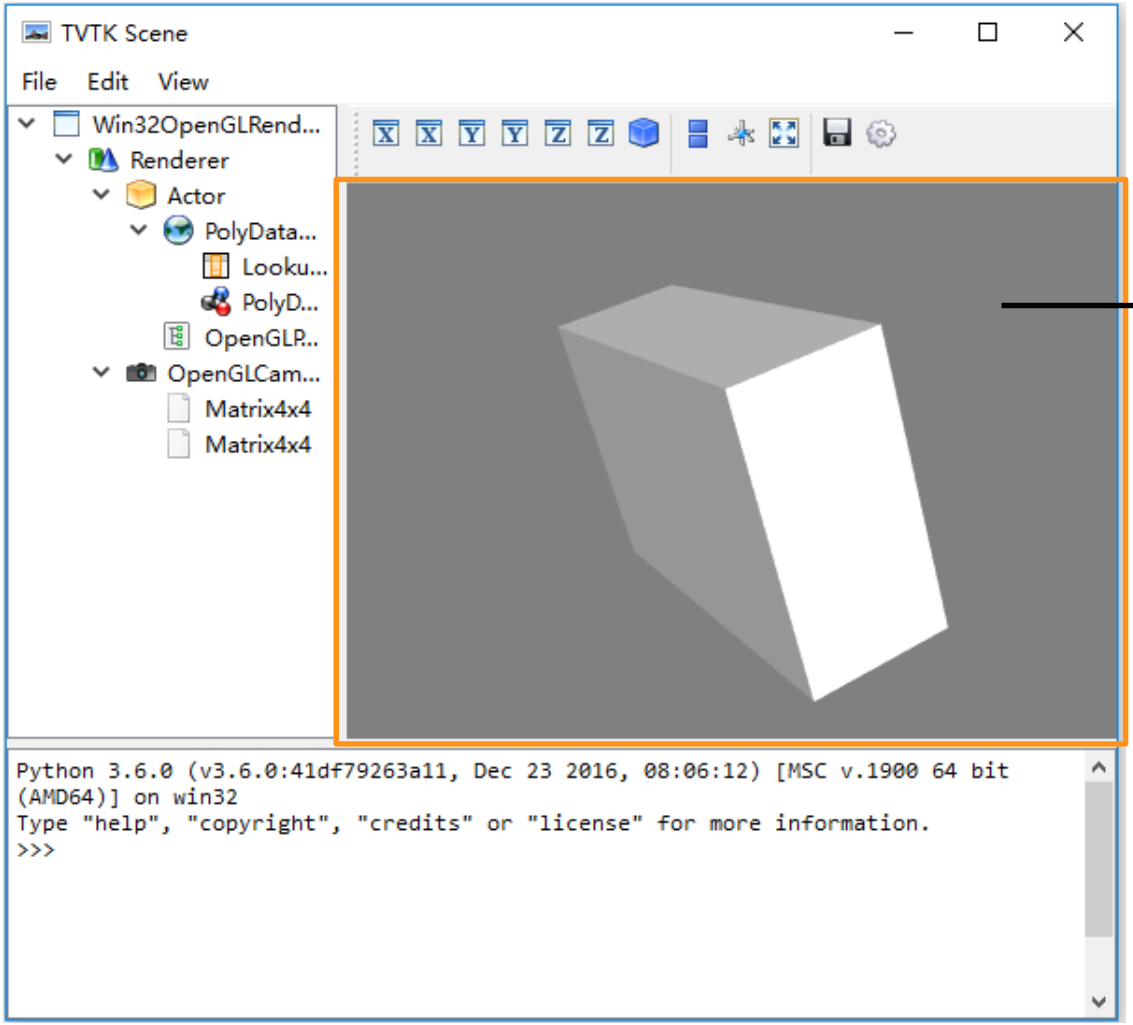
Tvtkfunc.py

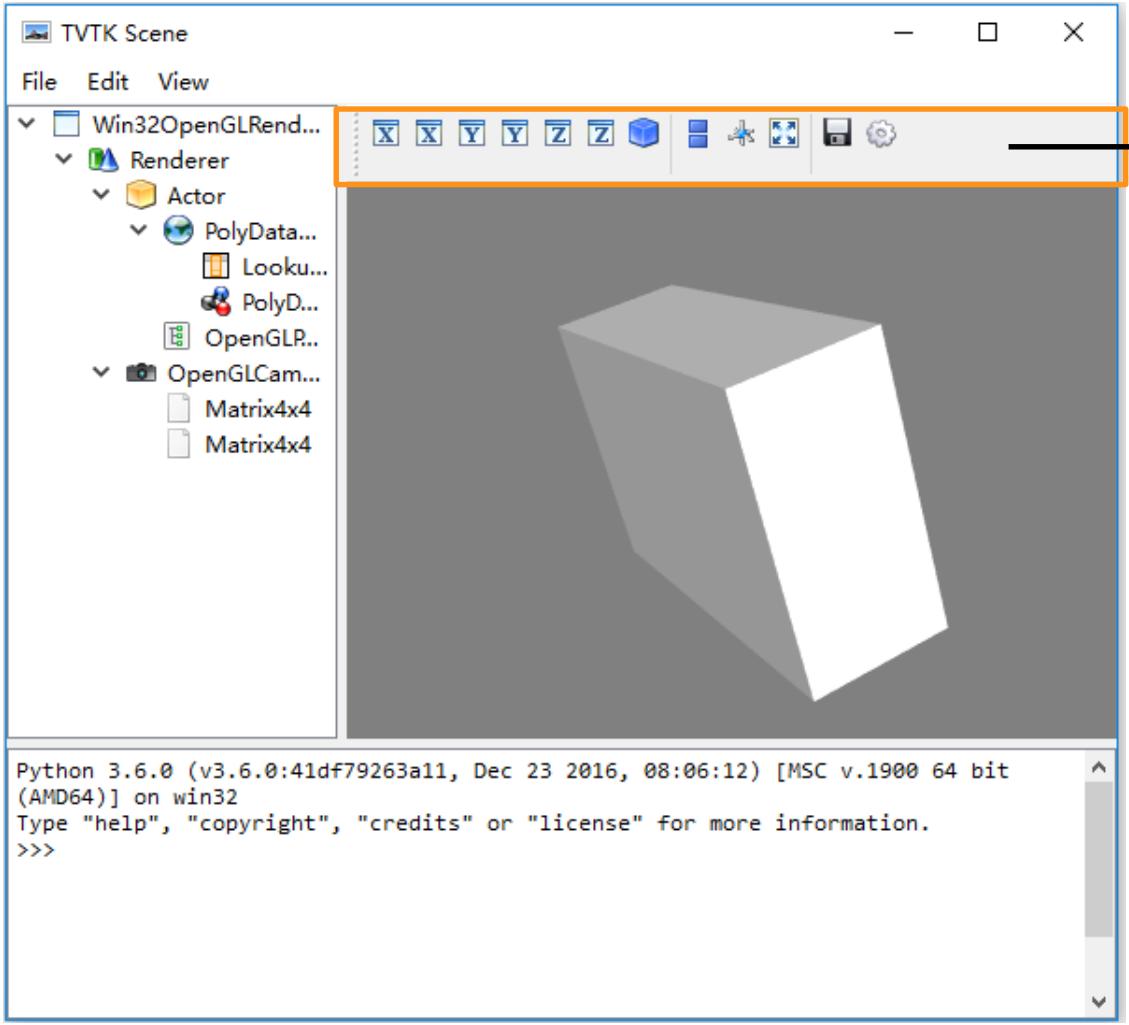
```python
def ivtk_scene(actors):
    from tvtk.tools import ivtk
    #创建一个带Crust（Python Shell）的窗口
    win = ivtk.IVTKWithCrustAndBrowser()
    win.open()
    win.scene.add_actor(actors)
    #修正窗口错误
    dialog = win.control.centralWidget().widget(0).widget(0)
    from pyface.qt import QtCore
    dialog.setWindowFlags(QtCore.Qt.WindowFlags(0x00000000))
    dialog.show()
    return win

def event_loop():
    from pyface.api import GUI
    gui = GUI()
    gui.start_event_loop()
```

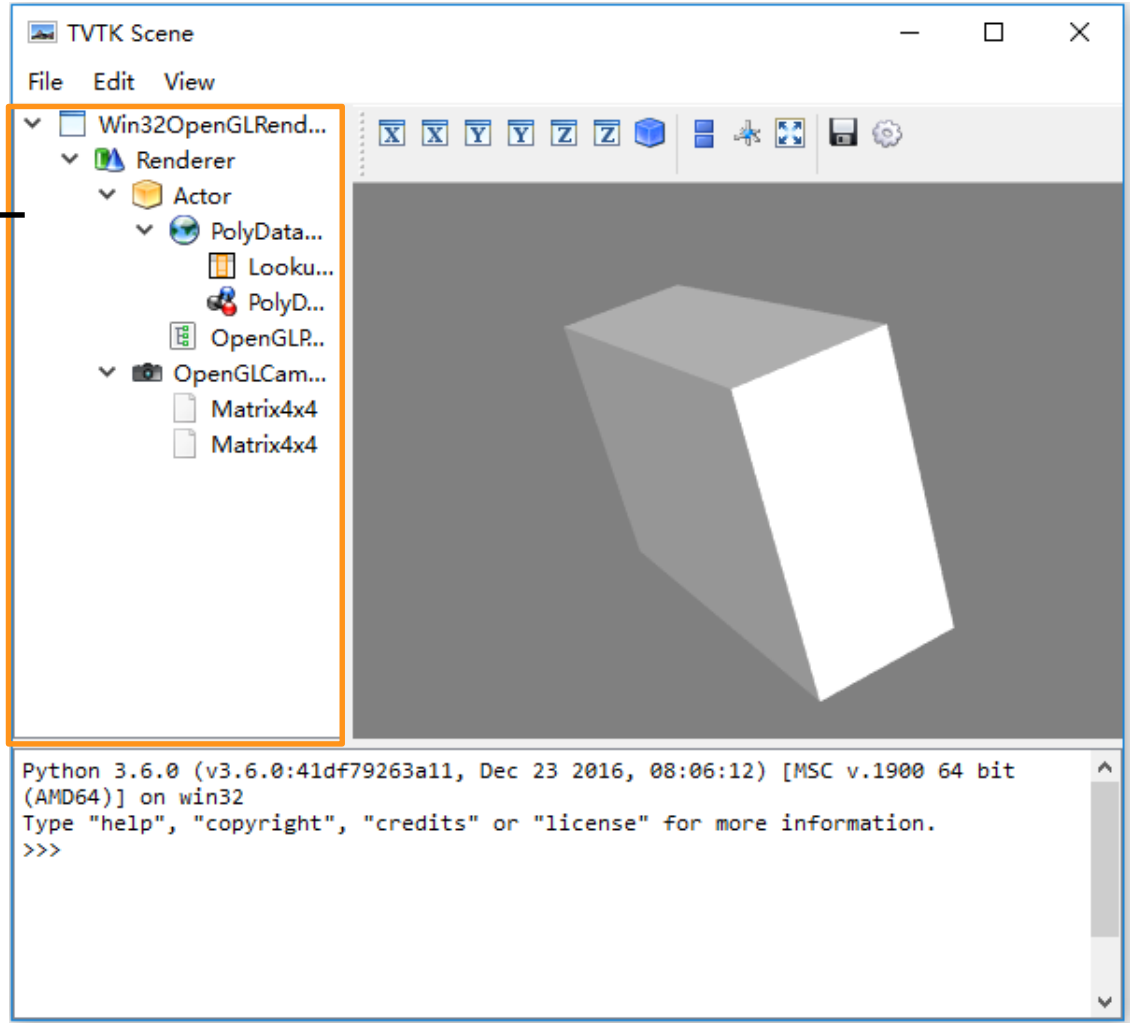
from tvtkfunc import ……

Cube_ivtk.py

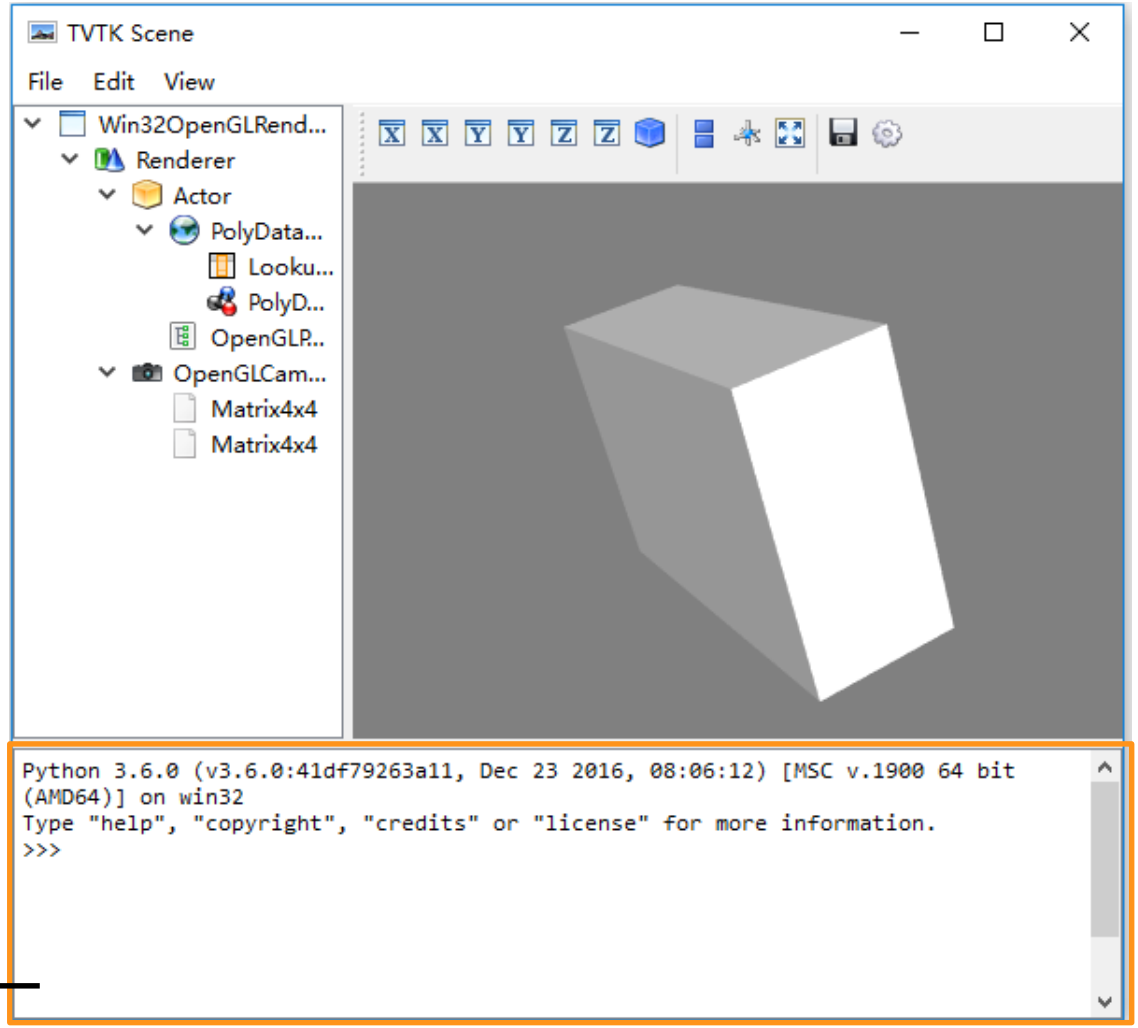
```python
from tvtk.api import tvtk
from tvtkfunc import ivtk_scene,event_loop

s = tvtk.CubeSource(x_length=1.0, y_length=2.0, z_length=3.0)
m = tvtk.PolyDataMapper(input_connection=s.output_port)
a = tvtk.Actor(mapper=m)
win = ivtk_scene(a)
win.scene.isometric_view()
event_loop()
```


TVTK Scene
File
Edit
View
Win32OpenGLRend...
Renderer
Actor
PolyData...
Looku...
PolyD...
OpenGLP...
OpenGLCam...
Matrix4x4
Matrix4x4
Python 3.6.0 (v3.6.0:41df79263a11, Dec 23 2016, 08:06:12) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
场景

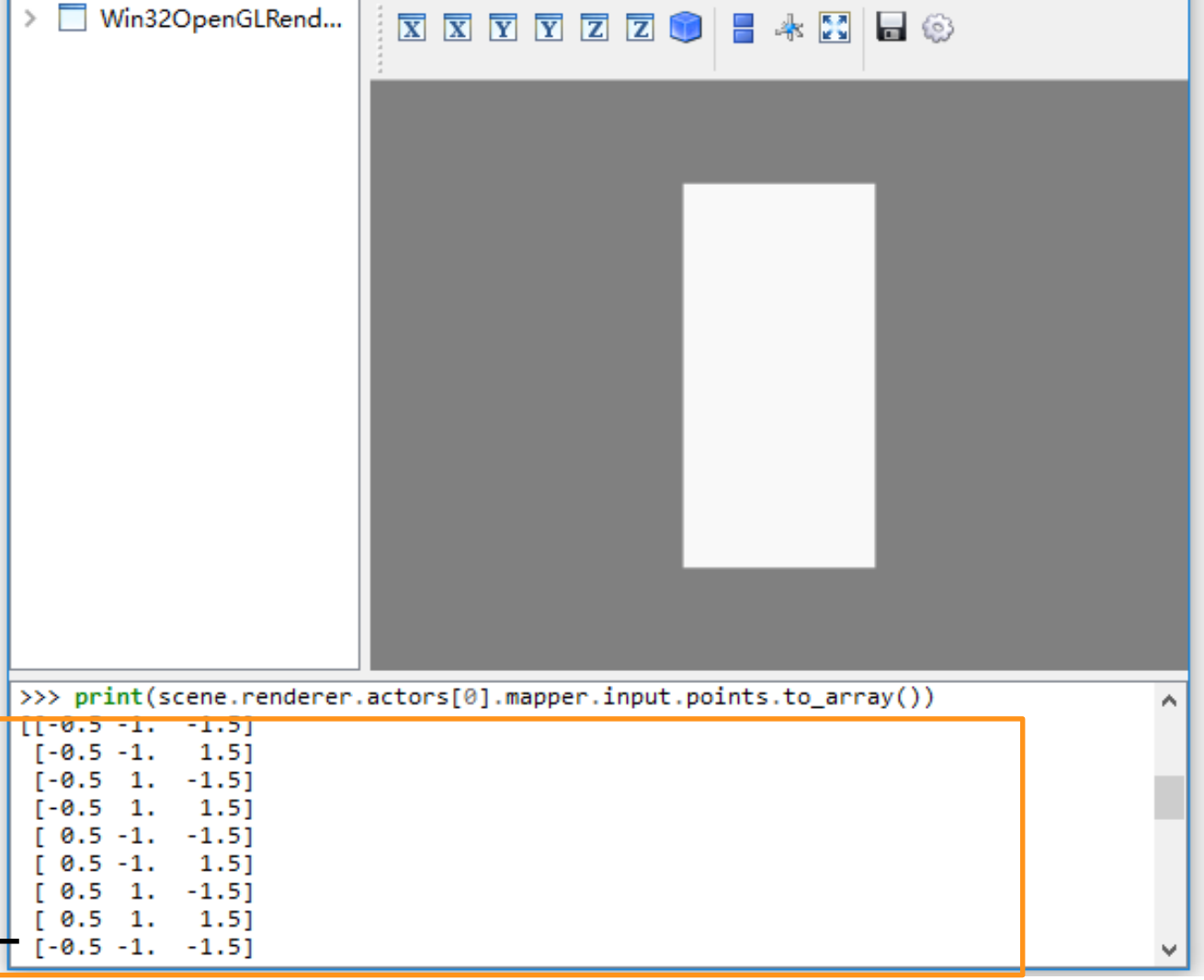

TVTK Scene
File
Edit
View
Win32OpenGLRend...
Renderer
Actor
PolyData...
Looku...
PolyD...
OpenGLP...
OpenGLCam...
Matrix4x4
Matrix4x4
Python 3.6.0 (v3.6.0:41df79263a11, Dec 23 2016, 08:06:12) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
场景工具条


TVTK Scene
File
Edit
View
Win32OpenGLRend...
Renderer
Actor
PolyData...
Looku...
PolyD...
OpenGLP...
OpenGLCam...
Matrix4x4
Matrix4x4
Python 3.6.0 (v3.6.0:41df79263a11, Dec 23 2016, 08:06:12) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>

管线浏览器

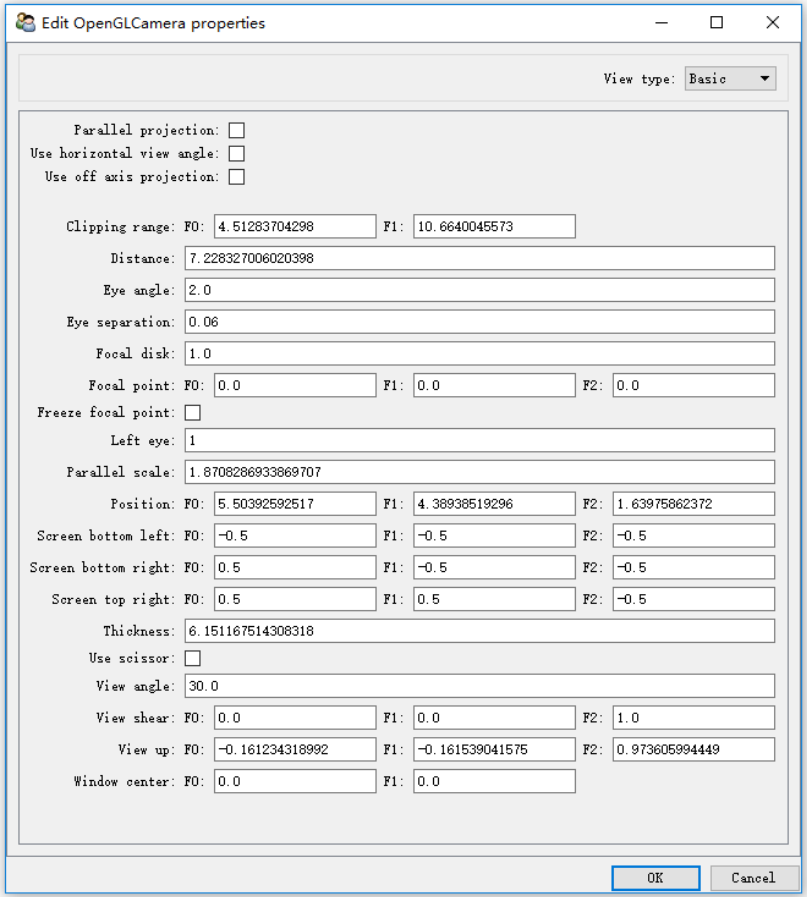

TVTK Scene
File
Edit
View
Win32OpenGLRend...
Renderer
Actor
PolyData...
Looku...
PolyD...
OpenGLP...
OpenGLCam...
Matrix4x4
Matrix4x4
Python 3.6.0 (v3.6.0:41df79263a11, Dec 23 2016, 08:06:12) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>

Python命令行



构成长方体顶点的三维坐标。

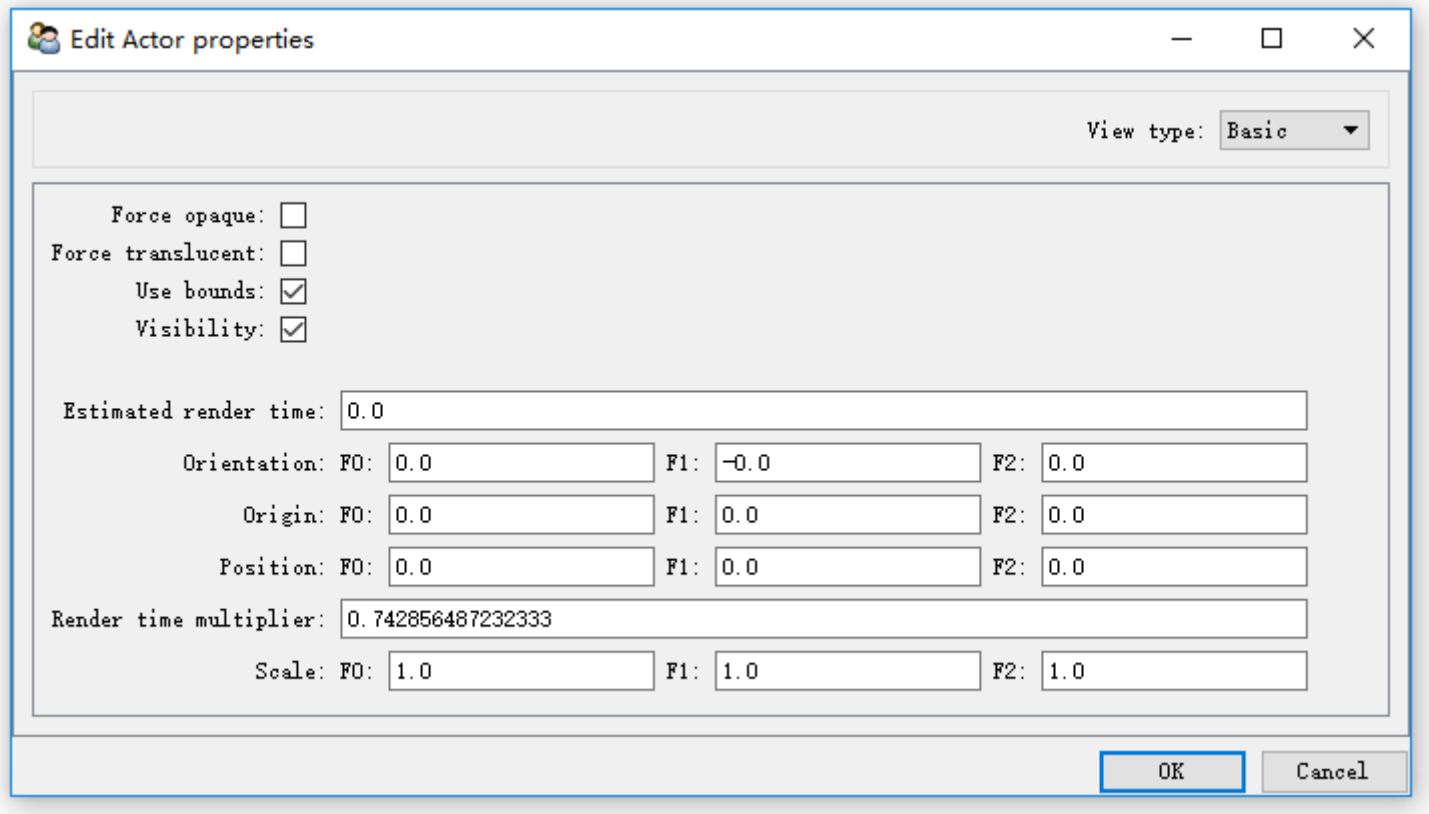
照相机


Edit OpenGLCamera properties
View type: Basic
Parallel projection:
Use horizontal view angle:
Use off axis projection:
Clipping range: F0: 4.51283704298 F1: 10.6640045573
Distance: 7.228327006020398
Eye angle: 2.0
Eye separation: 0.06
Focal disk: 1.0
Focal point: F0: 0.0 F1: 0.0 F2: 0.0
Freeze focal point:
Left eye: 1
Parallel scale: 1.8708286933869707
Position: F0: 5.50392592517 F1: 4.38938519296 F2: 1.63975862372
Screen bottom left: F0: -0.5 F1: -0.5 F2: -0.5
Screen bottom right: F0: 0.5 F1: -0.5 F2: -0.5
Screen top right: F0: 0.5 F1: 0.5 F2: -0.5
Thickness: 6.151167514308318
Use scissor:
View angle: 30.0
View shear: F0: 0.0 F1: 0.0 F2: 1.0
View up: F0: -0.161234318992 F1: -0.161539041575 F2: 0.973605994449
Window center: F0: 0.0 F1: 0.0
OK
Cancel

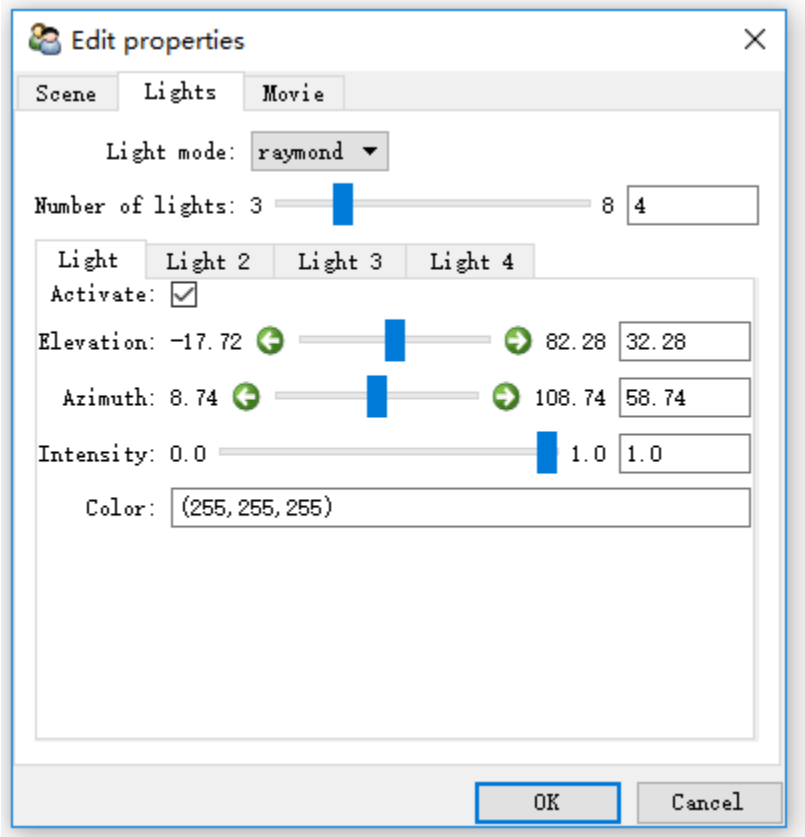
照相机属性

属性	说明
clipping_plane	它有两个元素，分别表示照相机到近、远两个裁剪平面的距离。在这两个平面范围之外将不会显示
position	照相机在三维空间中的坐标
focal_point	照相机所聚焦的焦点坐标
view_up	照相机的上方向矢量

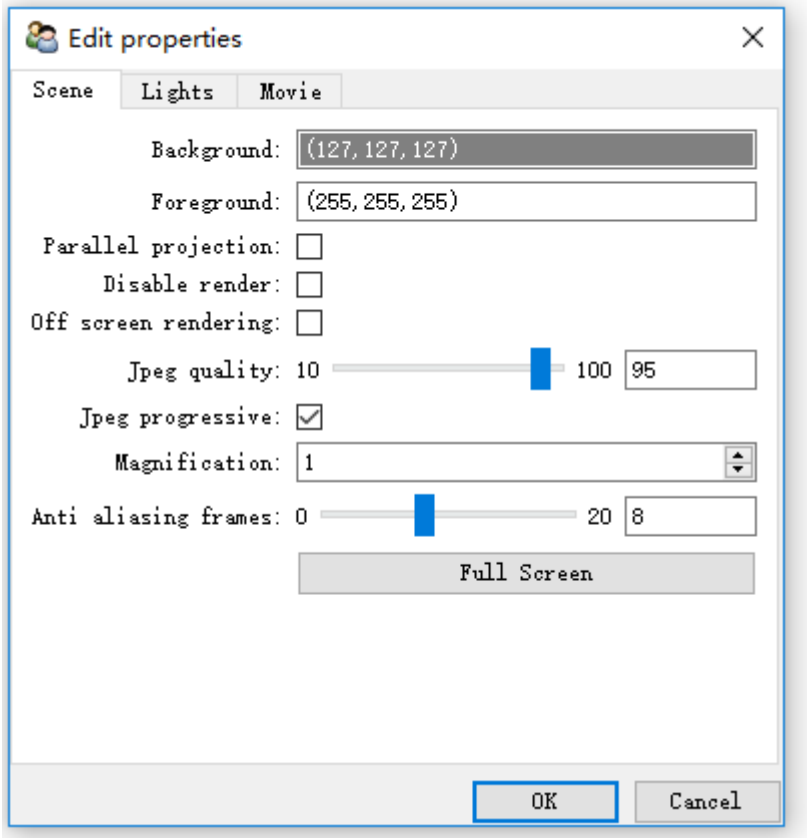
实体Actor

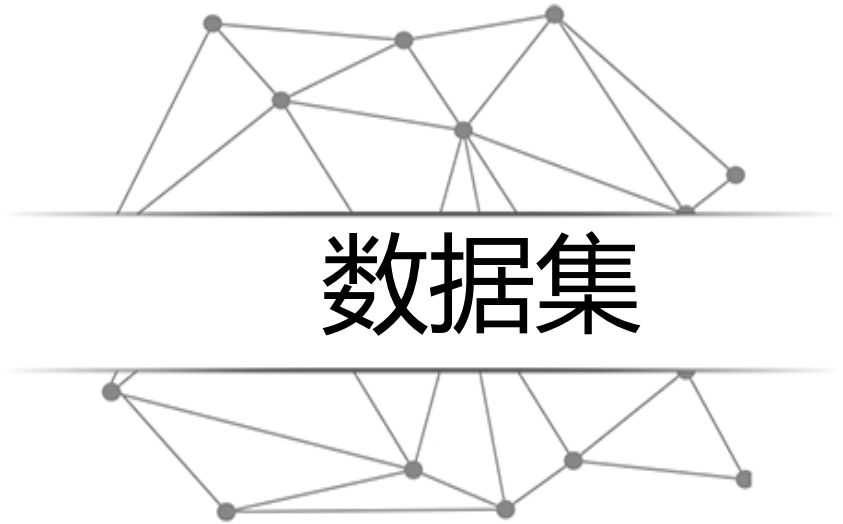

Edit Actor properties
View type: Basic
Force opaque:
Force translucent:
Use bounds:
Visibility:
Estimated render time: 0.0
Orientation: F0: 0.0 F1: -0.0 F2: 0.0
Origin: F0: 0.0 F1: 0.0 F2: 0.0
Position: F0: 0.0 F1: 0.0 F2: 0.0
Render time multiplier: 0.742856487232333
Scale: F0: 1.0 F1: 1.0 F2: 1.0
OK
Cancel

光源


Edit properties
Scene
Lights
Movie
Light mode: raymond
Number of lights: 3
8
4
Light
Light 2
Light 3
Light 4
Activate:
Elevation: -17.72
82.28
32.28
Azimuth: 8.74
108.74
58.74
Intensity: 0.0
1.0
1.0
Color: (255,255,255)
OK
Cancel

场景


Edit properties
Scene
Lights
Movie
Background:
(127, 127, 127)
Foreground:
(255, 255, 255)
Parallel projection:
Disable render:
Off screen rendering:
Jpeg quality:
10
100
95
Jpeg progressive:
Magnification:
1
Anti aliasing frames:
0
20
8
Full Screen
OK
Cancel



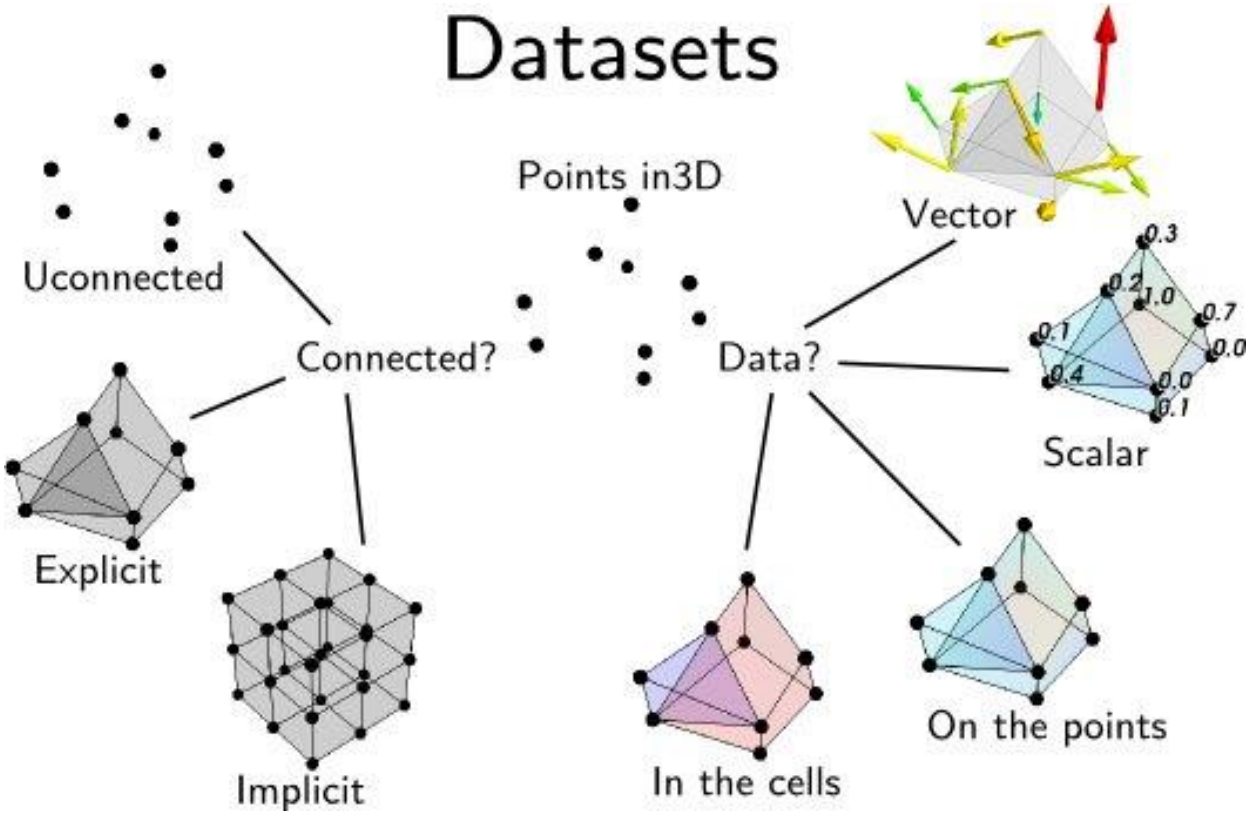
数据集

数据集

数据集（Dataset）。

- 点（Point）和数据（Data）
- 点之间：连接 vs 非连接
- 多个相关的点组成单元（Cell）
- 点的连接：显式vs隐式
- 数据：标量（Scalar）vs 矢量（Vector）

数据集

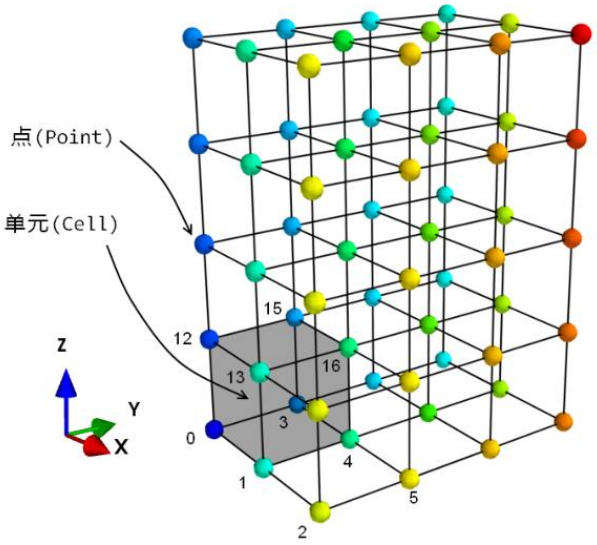

Datasets
Points in3D
Uconnected
Connected?
Explicit
Implicit
Data?
Vector
Scalar
In the cells
On the points

数据集

VTK name	Connectivity	Suitable for	Required information
ImageData	Implicit	Volumes and surfaces	3D data array and spacing along each axis
RectilinearGrid	Implicit	Volumes and surfaces	3D data array and 1D array of spacing for each axis
StructuredGrid	Implicit	Volumes and surfaces	3D data array and 3D position arrays for each axis
PolyData	Explicit	Points, lines and surfaces	x, y, z, positions of vertices and arrays of surface Cells
UnstructuredGrid	Explicit	Volumes and surfaces	x, y, z positions of vertices and arrays of volume Cells

数据集-Imagedata

ImageData表示二维或三维图像的数据结构。



数据集-Imagedata

```
from tvtk.api import tvtk
img = tvtk.ImageData(spacing=(1,1,1),origin=(1,2,3),dimensions=(3,4,5))
```

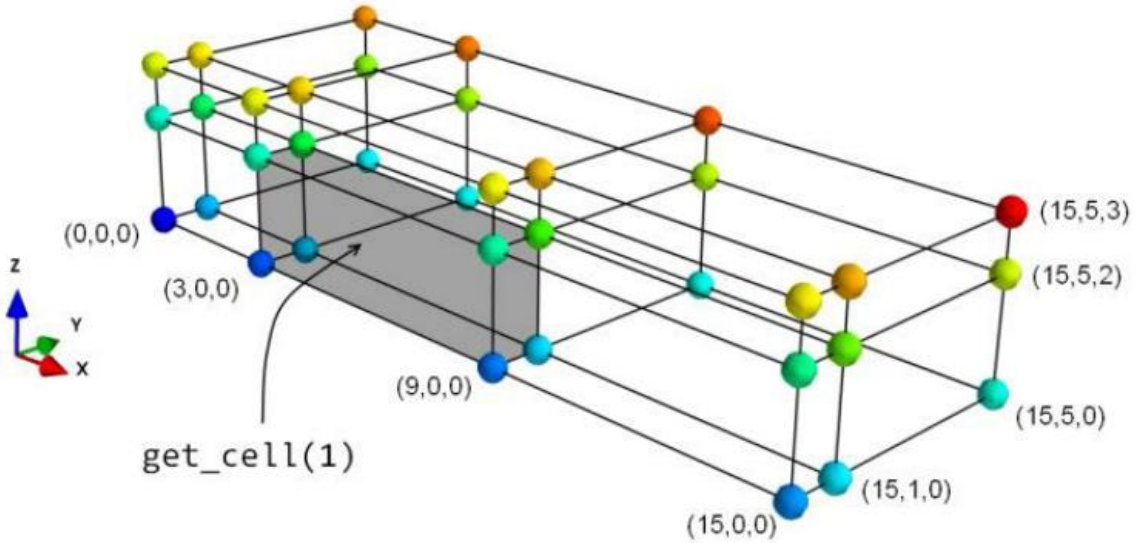
参数	说明
origin	三维网格数据的起点坐标
spacing	三维网格数据在X、Y、Z轴上的间距
dimensions	为在X、Y、Z轴上的网格数。

数据集-Imagedata

```
>>> img.get_point(0)
(1.0, 2.0, 3.0)
>>> for n in range(6):
        print("%.1f,%.1f,%.1f" % img.get_point(n))


1.0,2.0,3.0
2.0,2.0,3.0
3.0,2.0,3.0
1.0,3.0,3.0
2.0,3.0,3.0
3.0,3.0,3.0
```

数据集-RectilinearGrid

RectilinearGrid:间距不均匀的网格，所有点都在正交的网格上。



数据集-RectilinearGrid

```
from tvtk.api import tvtk
import numpy as np

x = np.array([0,3,9,15])
y = np.array([0,1,5])
z = np.array([0,2,3])
r = tvtk.RectilinearGrid()
r.x_coordinates = x
r.y_coordinates = y
r.z_coordinates = z
r.dimensions = len(x),len(y),len(z)
```

数据集-RectilinearGrid

```
>>> for n in range(6):
        print(r.get_point(n))


(0.0, 0.0, 0.0)
(3.0, 0.0, 0.0)
(9.0, 0.0, 0.0)
(15.0, 0.0, 0.0)
(0.0, 1.0, 0.0)
(3.0, 1.0, 0.0)
>>>
```

数据集-StructuredGrid

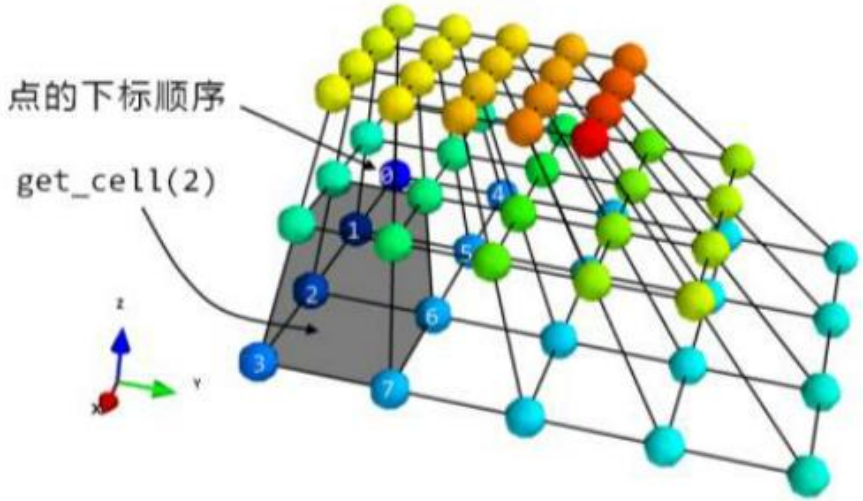
StructuredGrid:创建任意形状的网格，需要指定点的坐标。

points、

dimensions、

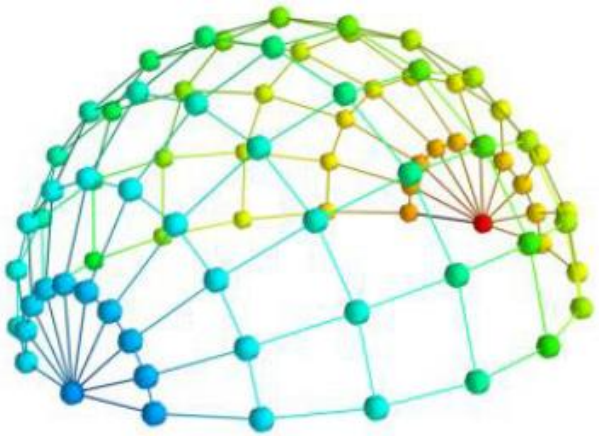
point_data.scalars

等属性进行初始化



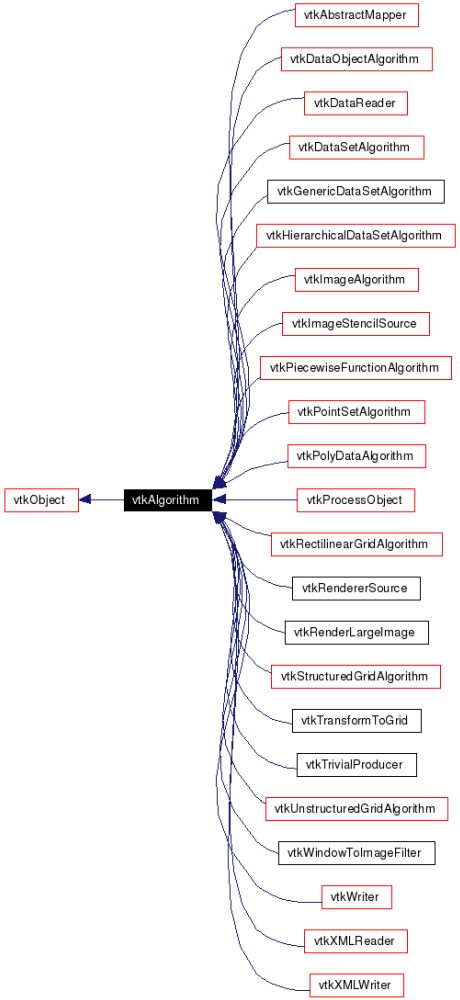
数据集-Polydata

PolyData:由一系列的点、点之间的联系以及由点构成的多边形组成。

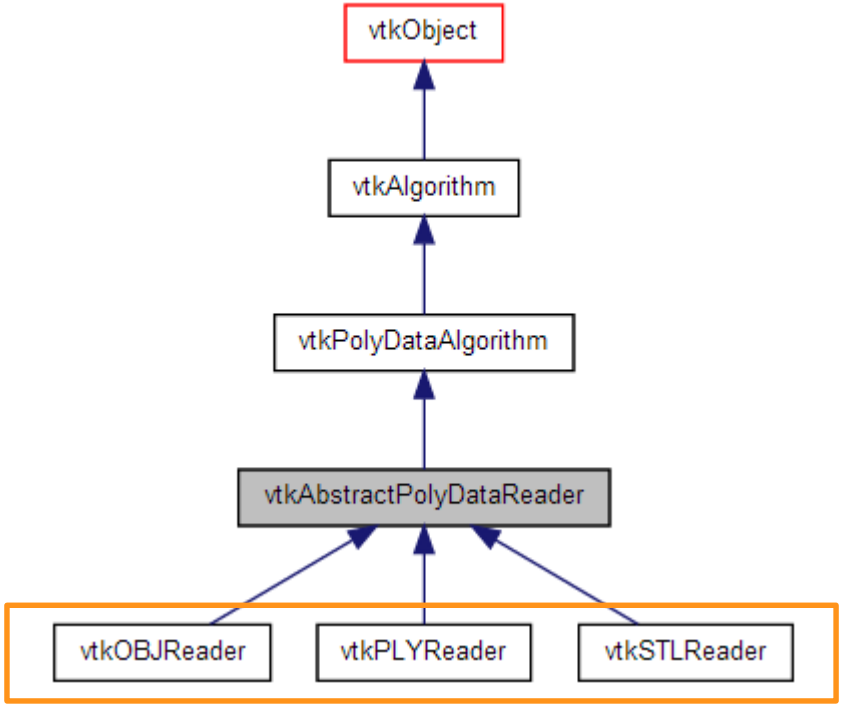


TVTK数据加载

VTK继承关系


vtkAbstractMapper
vtkDataObjectAlgorithm
vtkDataReader
vtkDataSetAlgorithm
vtkGenericDataSetAlgorithm
vtkHierarchicalDataSetAlgorithm
vtkImageAlgorithm
vtkImageStencilSource
vtkPiecewiseFunctionAlgorithm
vtkPointSetAlgorithm
vtkPolyDataAlgorithm
vtkObject
vtkAlgorithm
vtkProcessObject
vtkRectilinearGridAlgorithm
vtkRendererSource
vtkRenderLargeImage
vtkStructuredGridAlgorithm
vtkTransformToGrid
vtkTrivialProducer
vtkUnstructuredGridAlgorithm
vtkWindowToImageFilter
vtkWriter
vtkXMLReader
vtkXMLWriter

TVTK模型读取

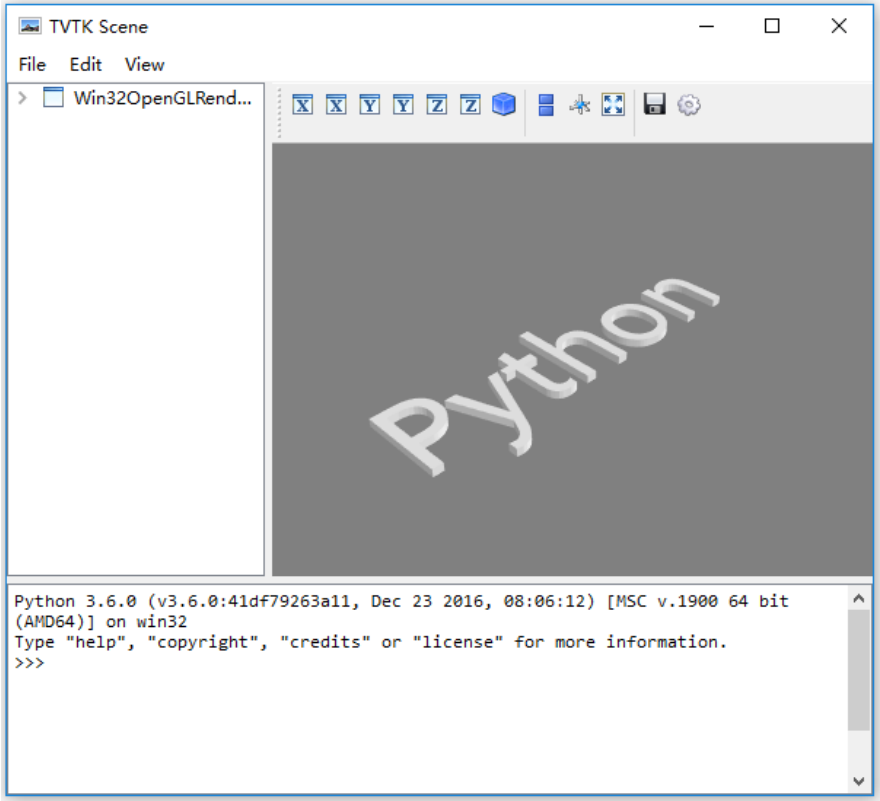

vtkObject
vtkAlgorithm
vtkPolyDataAlgorithm
vtkAbstractPolyDataReader
vtkOBJReader
vtkPLYReader
vtkSTLReader

TVTK模型读取

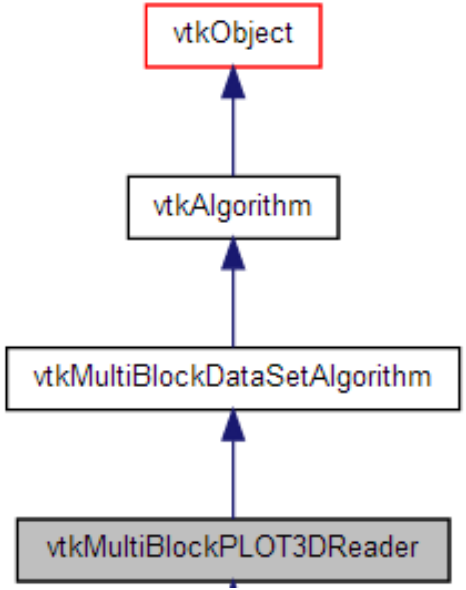
s=tvtk.STLReader(file_name = “stl文件名”)

```
from tvtk.api import tvtk
from tvtkfunc import ivtk_scene,event_loop

s = tvtk.STLReader(file_name = "python.stl")
m = tvtk.PolyDataMapper(input_connection = s.output_port)
a = tvtk.Actor(mapper = m)

win = ivtk_scene(a)
win.scene.isometric_view()
event_loop()
```

TVTK模型读取


TVTK Scene
File
Edit
View
Win32OpenGLRend...
Python
Python 3.6.0 (v3.6.0:41df79263a11, Dec 23 2016, 08:06:12) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>

TVTK MultiBlock数据读取



- Plot3D
  - 网格（XYZ 文件），
  - 空气动力学结果（Q 文件）
  - 通用结果

TVTK MultiBlock数据读取

```
Plot3d = tvtk.MultiBlockPLOT3DReader(
    xyz_file_name="combxyz.bin", #网格文件
    q_file_name="combq.bin", #空气动力学结果文件
    scalar_function_number=100, #设置标量数据数量
    vector_function_number=200 #设置矢量数据数量
)
```

TVTK MultiBlock数据读取

```
from tvtk.api import tvtk

def read_data():
    # 读入数据
    plot3d = tvtk.MultiBlockPLOT3DReader(
        xyz_file_name="combxyz.bin",#网格文件
        q_file_name="combq.bin",#空气动力学结果文件
        scalar_function_number=100, vector_function_number=200
    )
    plot3d.update()
    return plot3d

plot3d = read_data()
grid = plot3d.output.get_block(0)
```

TVTK MultiBlock数据读取

```
>>> print(type(plot3d.output))
<class 'tvtk.tvtk_classes.multi_block_data_set.MultiBlockDataSet'>
>>> print(type(plot3d.output.get_block(0)))
<class 'tvtk.tvtk_classes.structured_grid.StructuredGrid'>
```

```
>>> print(grid.dimensions)
[57 33 25]
>>> print(grid.points.to_array())
[[  2.66700006  -3.77476001  23.83292007]
 [  2.94346499  -3.74825287  23.66555977]
 [  3.21985817  -3.72175312  23.49823952]
 ...,
 [ 15.84669018   5.66214085  35.7493782 ]
 [ 16.17829895   5.66214085  35.7493782 ]
 [ 16.51000023   5.66214085  35.7493782 ]]
>>> print(grid.cell_data.number_of_arrays)
0
>>> print(grid.point_data.number_of_arrays)
4
>>> print(grid.point_data.scalars.name)
Density
>>> print(grid.point_data.vectors.name)
Velocity
>>>
```