科学可视化基础

SV01

黄天羽

www.python123.org

概念的提出

“可视化”概念的提出

Visualization ← Visual

视觉的
形象的

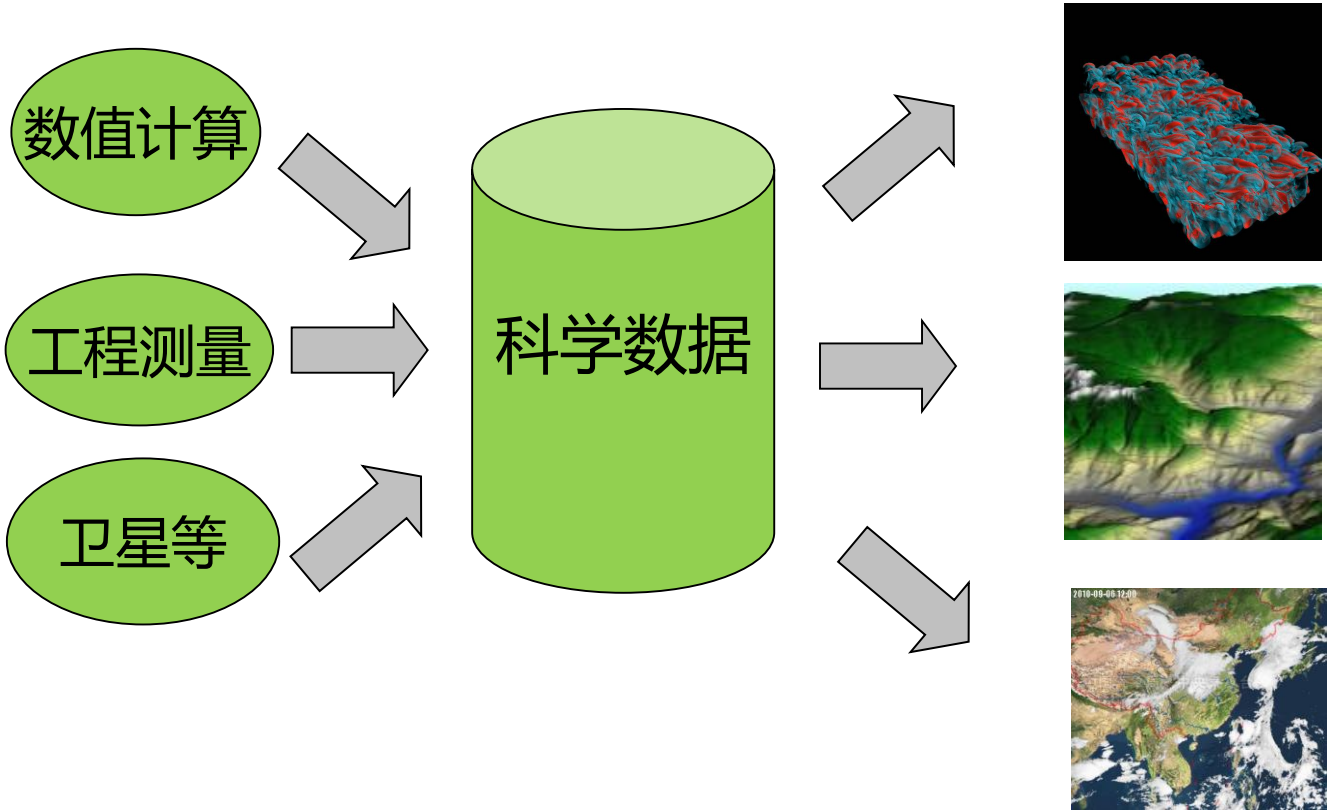
1987年2月, 美国国家科学基金会(NSF)

抽象的事务、过程 ← 图形、图像

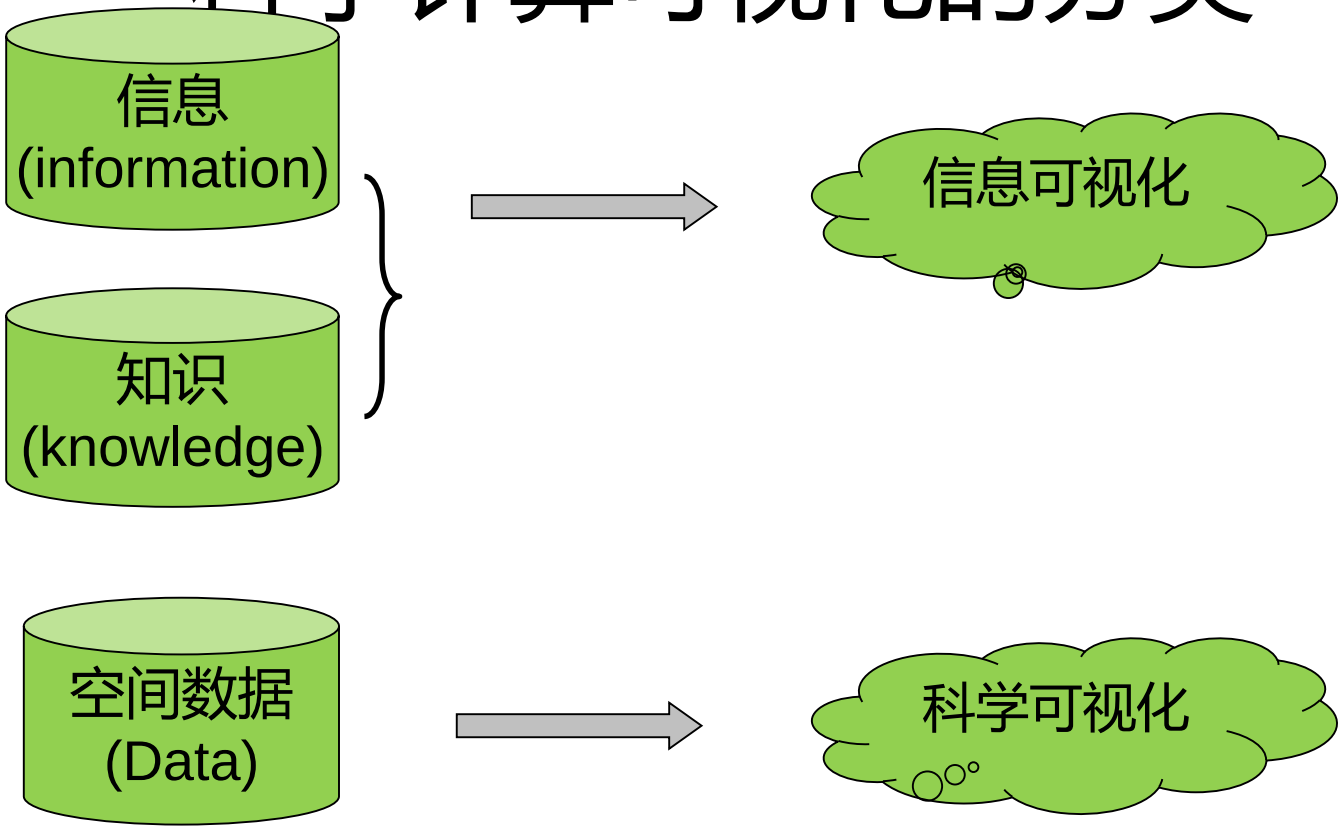
可视化

可视化界面(图形界面)、可视化编程等

科学计算可视化的含义



科学计算可视化的分类



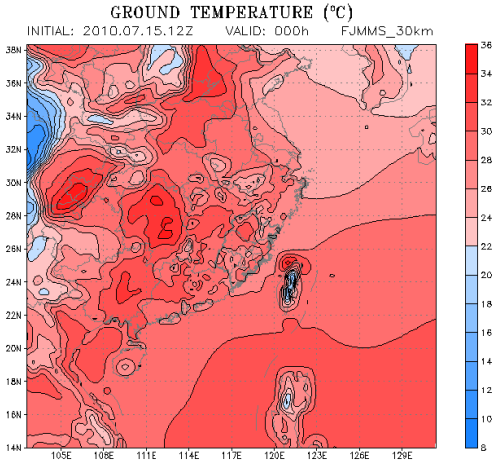
科学计算的可视化方法

科学计算可视化的主要方法

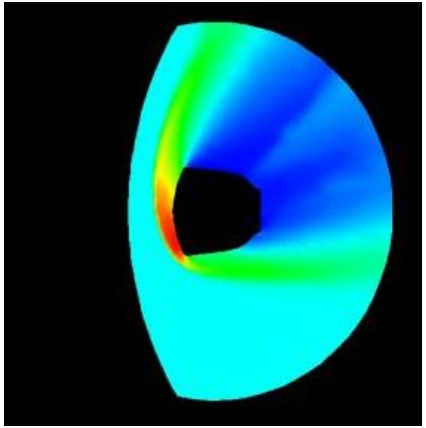
- 1 二维标量数据场
  - 1.1 颜色映射方法
  - 1.2 等值线方法
  - 1.3 立体图法和层次分割法
- 2 三维标量数据场
  - 2.1 面绘制方法(surface rendering)
  - 2.2 体绘制方法(volume rendering)
- 3 矢量数据场
  - 3.1 直接法
  - 3.2 流线法 (stream line)

颜色映射法

将颜色与数据之间建立映射关系



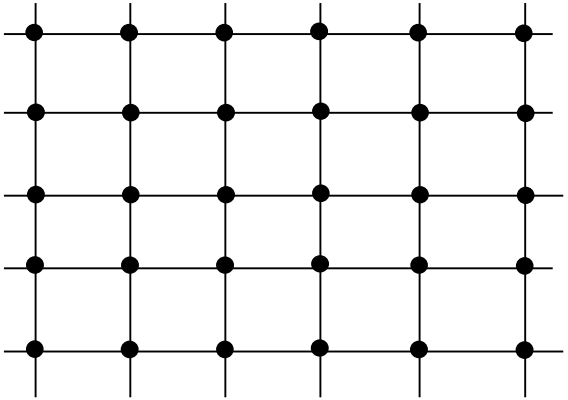
中国部分区域温度度分布图



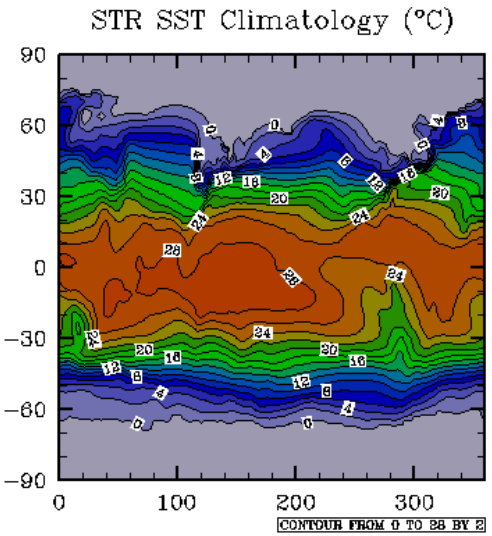
某宇宙飞船周围空气密度分布图

等值线方法

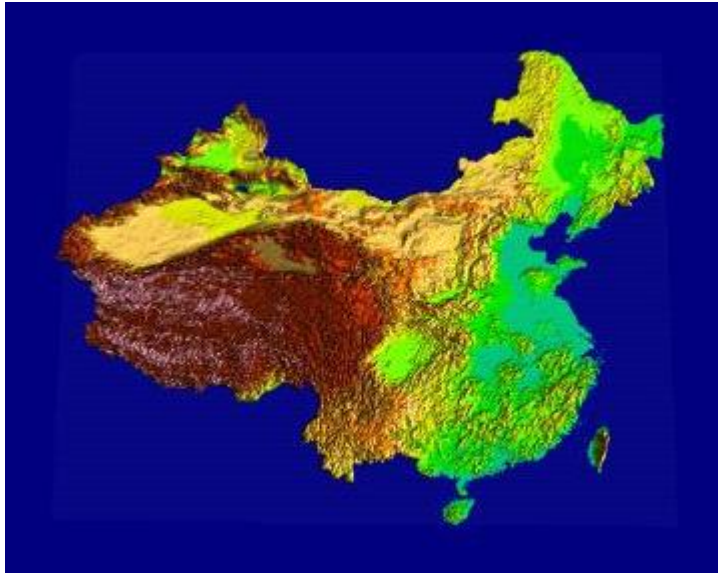
F(xi,yi)=f (f为给定的值)



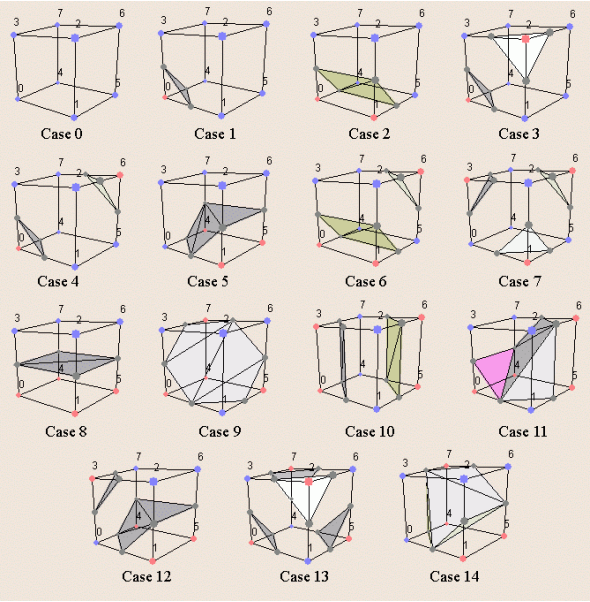
二维规则数据场示意图

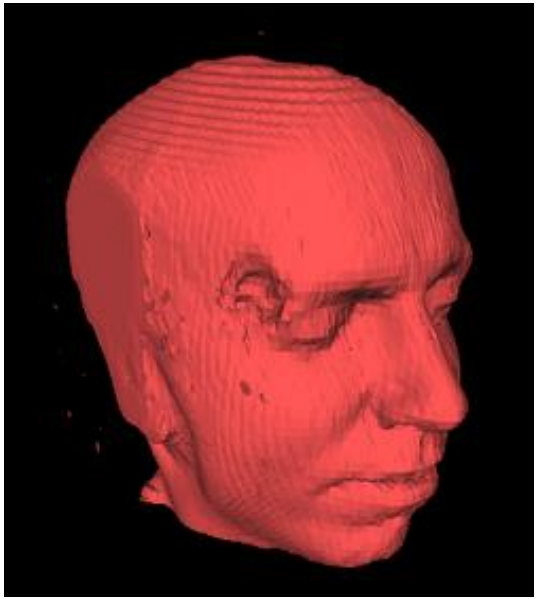


立体图法和层次分割法

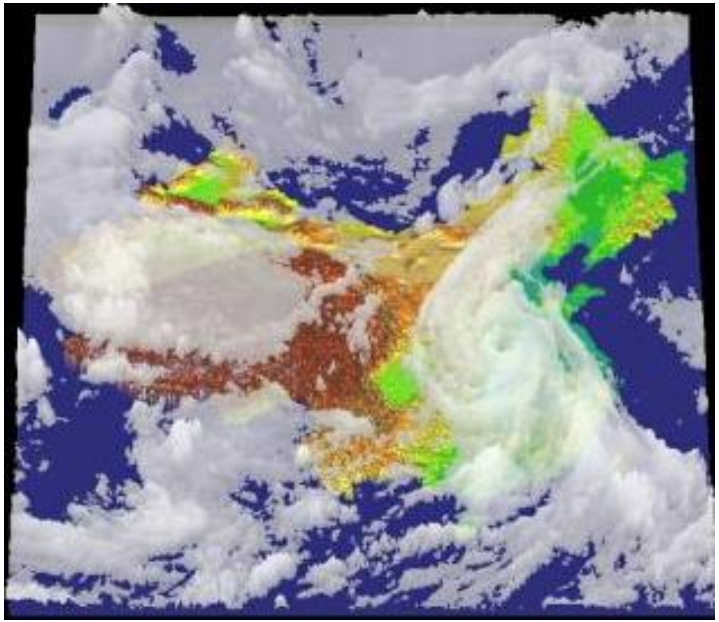


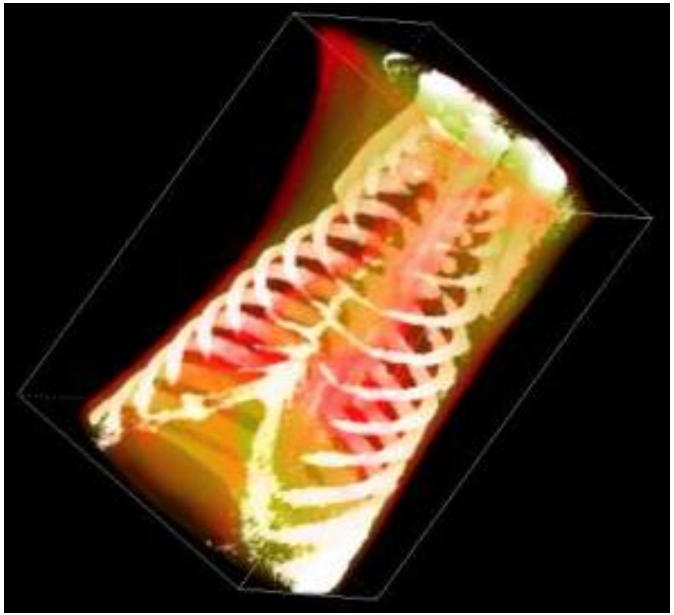
面绘制方法


Case 0
Case 1
Case 2
Case 3
Case 4
Case 5
Case 6
Case 7
Case 8
Case 9
Case 10
Case 11
Case 12
Case 13
Case 14

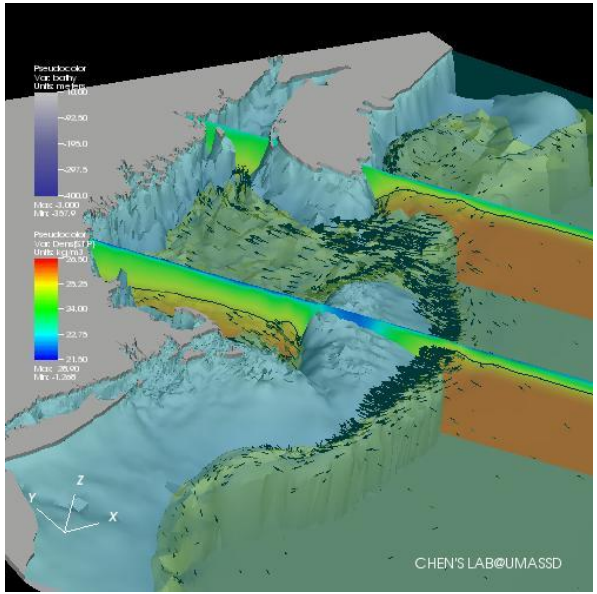


体绘制方法

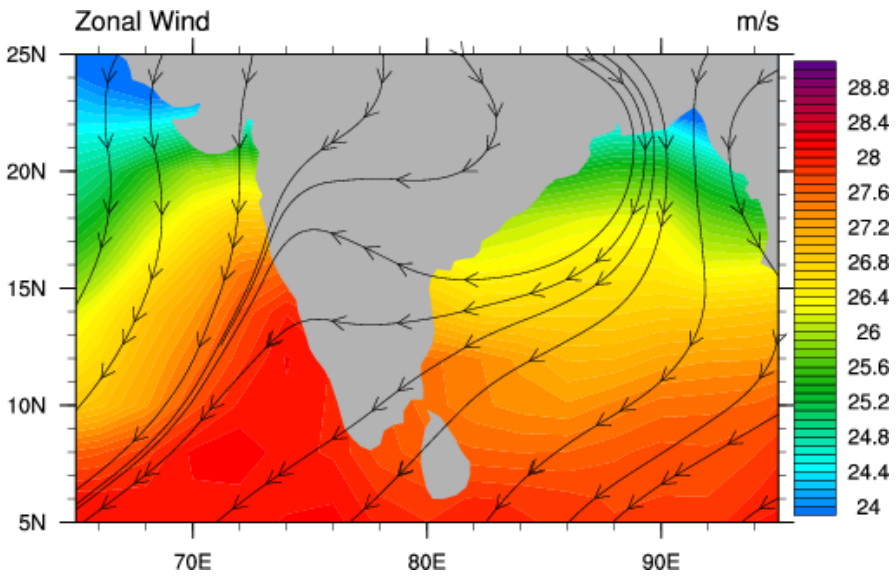


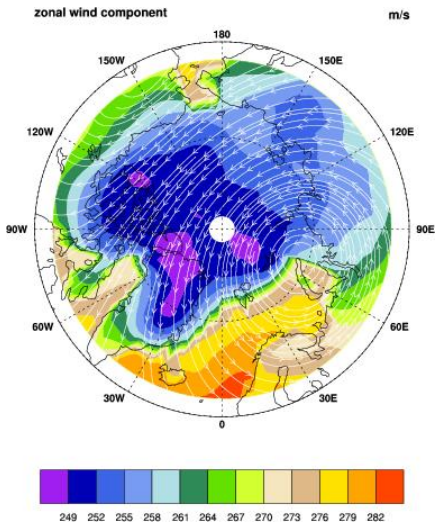


矢量数据场直接法


Pseudocolor
Var: bathy
Units: meters
-82.50
-165.0
-247.5
-330.0
Max: -3.000
Min: -337.9
Pseudocolor
Var: Density(T,P)
Units: kg/m3
28.50
26.25
24.00
22.75
21.50
Max: 28.00
Min: -1.266
Z
Y
X
CHEN'S LAB@UMASSD

矢量数据场流线法


Zonal Wind
m/s
25N
20N
15N
10N
5N
70E
80E
90E
28.8
28.4
28
27.6
27.2
26.8
26.4
26
25.6
25.2
24.8
24.4
24


zonal wind component
m/s
180
150W
150E
120W
120E
90W
90E
60W
60E
30W
30E
0
249 252 255 258 261 264 267 270 273 276 279 282

应用领域

地球科学

大气科学

医学/生命科学

生物/分子科学

航空/航天/工业

化学/化工

物理/力学

人类/考古学

地质勘探等

TVTK库的安装

code.ENTHOUGHT.com

home | www.enthought.com | www.scipy.org | installation

PROJECTS | INSTALL | SOURCE | SUPPORT

PROJECTS

http://code.enthought.com/projects/mayavi/

- Enthought Tool Suite
- Traits
- TraitsUI
- Enaml
- Chaco
- **Mayavi**
  - **Overview**
  - Documentation
  - Issues »
  - Screen Shots
  - Development Details

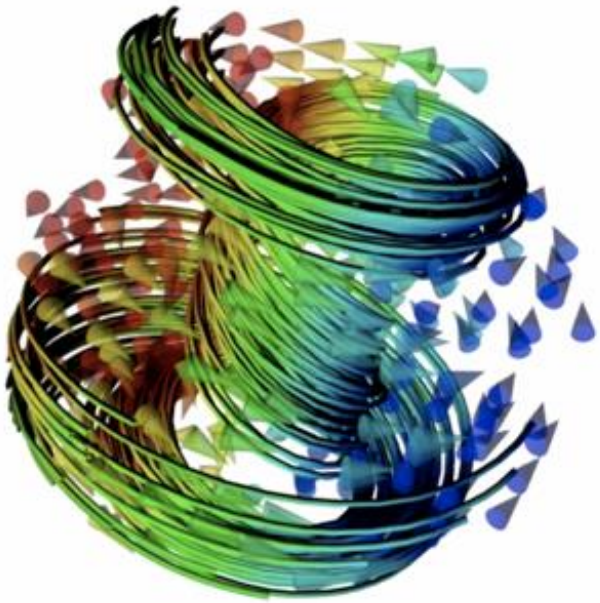
Mayavi Project

3D Scientific Data Visualization and Plotting

The Mayavi *project* includes two related *packages* for 3-dimensional visualization:

- **Mayavi**: A tool for **easy and interactive** visualization of data, with **seamless integration with Python scientific libraries.**
- **TVTK**: A Traits-based wrapper for the Visualization Toolkit, a popular open-source visualization library.

These libraries operate at different levels of abstraction. TVTK manipulates visualization objects, while Mayavi lets you operate on your data, and then see the results. Most users either use the Mayavi user interface or program to its scripting interface; you probably don't need to interact with TVTK unless you want to create a new Mayavi module.



TVTK库的安装

IDE编程环境：

Python3.6/IDLE3.6

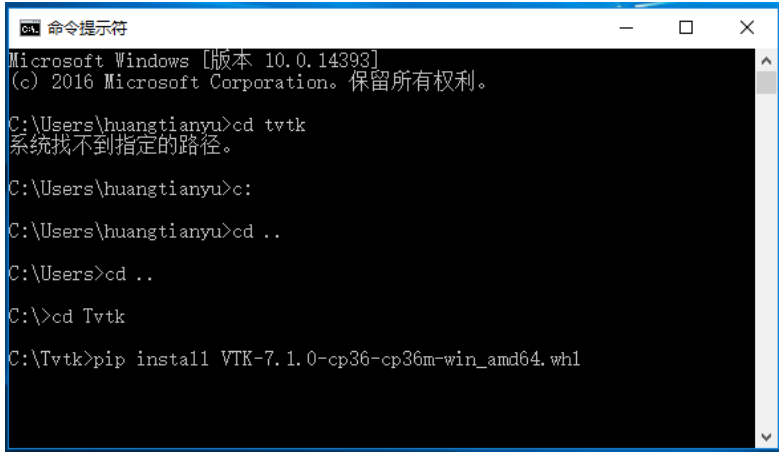
Pycharm Community Edition 2017

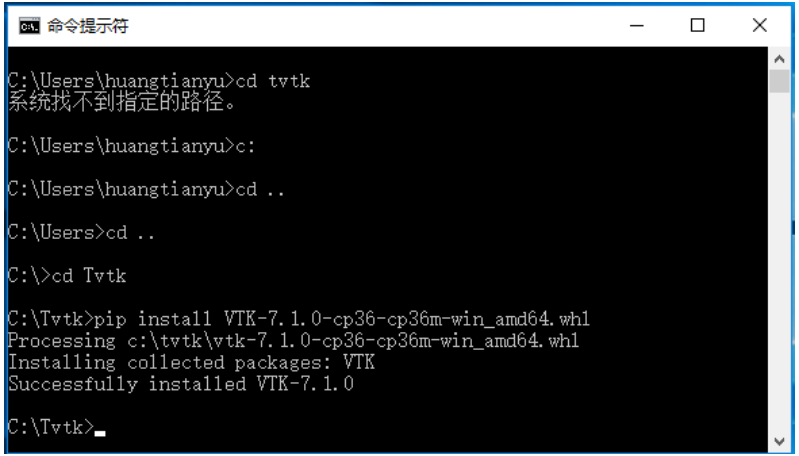
TVTK库的安装

下载 VTK-7.1.1-cp36-cp36m-win_amd64.whl

“以管理员身份运行” cmd

在下载目录执行pip install VTK-7.1.1-cp36-cp36m-win_amd64.whl





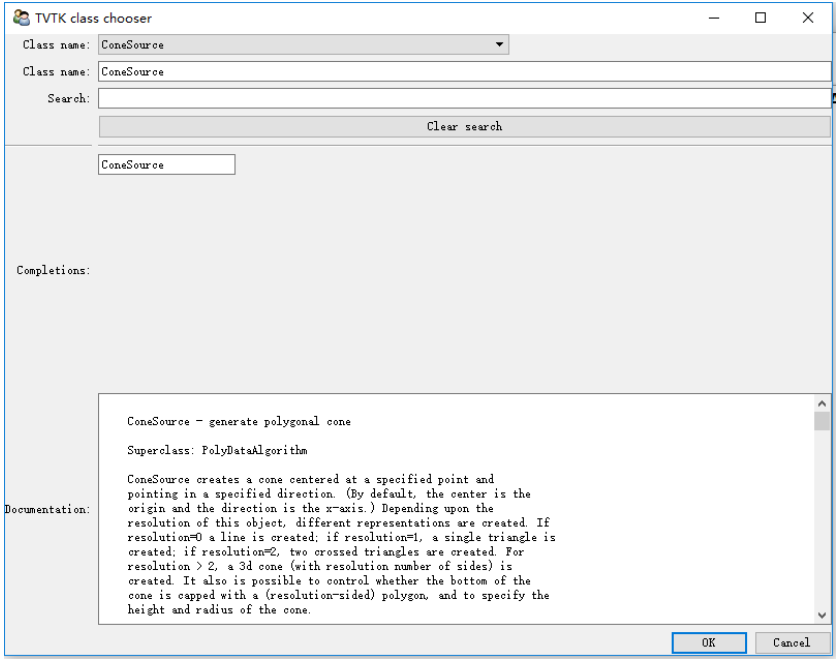
TVTK库的安装

Windows10 + Python 3.6 环境下安装：

- VTK-7.1.1-cp36-cp36m-win_amd64.whl
- numpy-1.12.1+mkl-cp36-cp36m-win_amd64.whl
- traits-4.6.0-cp36-cp36m-win_amd64.whl
- mayavi-4.5.0+vtk71-cp36-cp36m-win_amd64.whl
- PyQt4-4.11.4-cp36-cp36m-win_amd64.whl

TVTK库的安装小测

```
>>> from tvtk.tools import tvtk_doc
>>> tvtk_doc.main()
```



TVTK库的安装小测

建立tvtk长方体数据源

```
>>> from tvtk.api import tvtk
>>> s = tvtk.CubeSource(x_length=1.0, y_length=2.0, z_length=3.0)
>>> print(s)
vtkCubeSource (0000028E98EA6AA0)
  Debug: Off
  Modified Time: 96
  Reference Count: 2
  Registered Events:
    Registered Observers:
      vtkObserver (0000028E9A6FC3F0)
        Event: 33
        EventName: ModifiedEvent
```

TVTK帮助资源

http://www.vtk.org/doc/nightly/html/annotated.html

VTK

Main Page | Related Pages | Modules | Namespaces | Classes | Files

Class List | Class Index | Class Hierarchy | Class Members

Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

ADIOS	
boost	Forward declaration required for Boost serialization
detail	
tovtkm	
vtk	
vtkArrayDispatch	
vtkBlockSortHelper	Collection of comparison functions for std::sort
vtkDispatcherCommon	
vtkDispatcherPrivate	
vtkDoubleDispatcherPrivate	
vtkm	
vtkParticleTracerBaseNamespace	
vtkTemporalStreamTracerNamespace	
vtkTypeList	

TVTK帮助资源

- TVTK库中的类名去除了前缀vtk
- 函数名按照Python的惯例，采用下划线连接单词

  如AddItem->add_item

- VTK对象的方法在TVTK中用Trait属性替代

```
m.SetInputConnnection(c.GetOutputPort())    #VTK
m.input_connection(c.output_port)           #TVTK
```

创建一个基本三维对象

tvtk.CubeSource()

```
s = tvtk.CubeSource(traits)
```

返回一个三维对象实例

对象变量

构造具有一定traits属性值的长方体数据源对象

构造函数

tvtk.CubeSource()

```
s = tvtk.CubeSource(x_length=1.0, y_length=2.0,
                    z_length=3.0)
```

- **x_length**: 立方体在X轴的长度
- **y_length**: 立方体在y轴的长度
- **z_length**: 立方体在z轴的长度

CubeSource对象

```
>>> from tvtk.api import tvtk
>>> s = tvtk.CubeSource(x_length=1.0, y_length=2.0, z_length=3.0)
>>> print(s)
vtkCubeSource (0000028E98EA6AA0)
  Debug: Off
  Modified Time: 96
  Reference Count: 2
  Registered Events:
    Registered Observers:
      vtkObserver (0000028E9A6FC3F0)
        Event: 33
        EventName: ModifiedEvent
```

三维对象变量s包含了构建三维长方体的所有信息

CubeSource对象的属性

属性	说明
s.x_length	长方体对象在x轴方向的长度
s.y_length	长方体对象在y轴方向的长度
s.z_length	长方体对象在z轴方向的长度
s.center	长方体对象所在坐标系的原点
s. output_points _precision	长方体对象的精度

```
>>> from tvtk.api import tvtk
>>> s = tvtk.CubeSource(x_length=1.0, y_length=2.0, z_length=3.0)
>>> s.x_length
1.0
>>> s.y_length
2.0
>>> s.z_length
3.0
>>> s.center
array([ 0.,  0.,  0.])
>>> s.output_points_precision
0
>>> |
```

CubeSource对象的方法

VTK方法	Tvtk	说明
Set/GetXLength()	x_length	设置/获取长方体对象在X轴方向的长度
Set/GetYLength()	y_length	设置/获取长方体对象在Y轴方向的长度
Set/GetZLength()	z_length	设置/获取长方体对象在Z轴方向的长度
Set/GetCenter()	center	设置/获取长方体对象所在坐标系的原点
…		…

具体调用示例参看Page 24

Tvtk库的基本三维对象

三维对象	说明
CubeSource	立方体三维对象数据源
ConeSource	圆锥三维对象数据源
CylinderSource	圆柱三维对象数据源
ArcSource	圆弧三维对象数据源
ArrowSource	箭头三维对象数据源

创建一个圆锥数据源

```
>>> s = tvtk.ConeSource(height=3.0, radius=1.0, resolution=36)
>>> s.height
3.0
>>> s.radius
1.0
>>> s.resolution
36
>>> s.center
array([ 0.,  0.,  0.])
>>> print(s)
vtkConeSource (0000028E996E6560)
  Debug: Off
  Modified Time: 260
  Reference Count: 2
  Registered Events:
    Registered Observers:
      vtkObserver (0000028E9A6FC5D0)
        Event: 33
        EventName: ModifiedEvent
        Command: 0000028E9ABA7100
        Priority: 0
        Tag: 1
  Executive: 0000028E9A581420
```

高度、底面圆半径、底面圆的分辨率

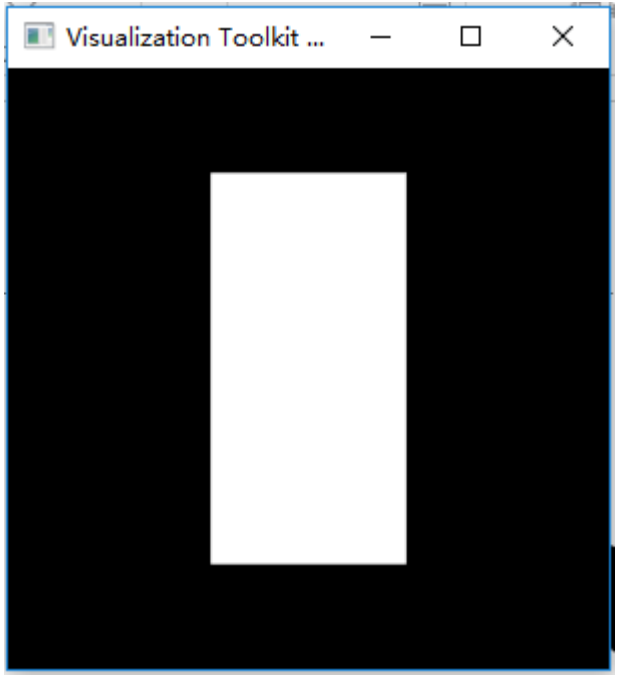
TVTK库显示一个三维对象

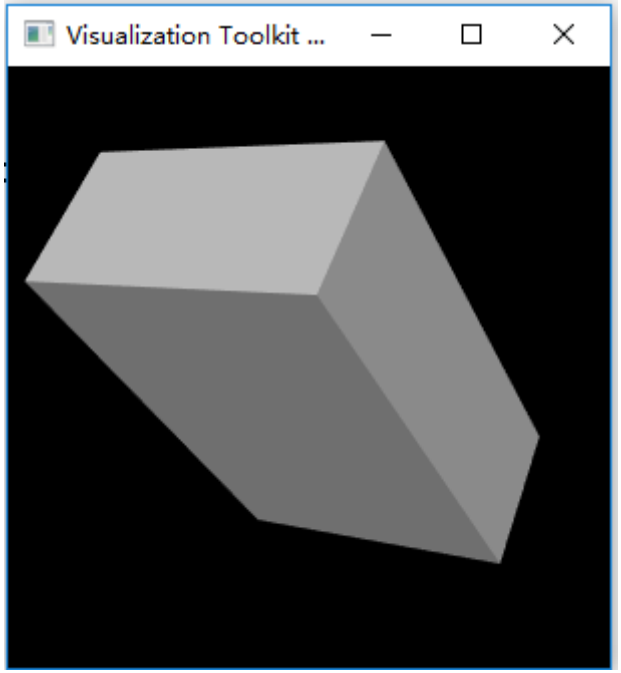
显示一个长方体

```
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()
```

显示一个长方体





原始数据转换为屏幕上图像，TVTK对象共同协调完成：

```
tvtk.CubeSource
tvtk.PolyDataMapper
tvtk.Actor
tvtk.Renderer
tvtk.RenderWindow
tvtk.RenderWindowInteractor
```

在TVTK中，这种对象之间协调完成工作的过程被称作**管线(Pipeline)**。