鼠标选取操作

鼠标选取

- 选取一个物体，查看数据
- 选取物体上一点，查看数据

鼠标选取

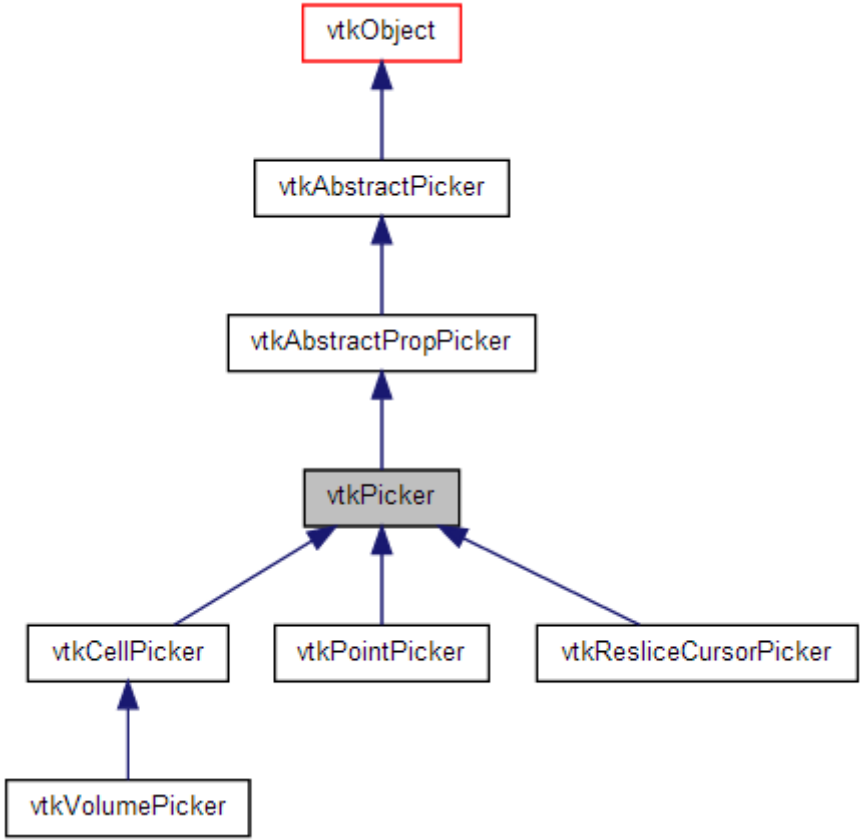
on_mouse_pick(callback, type='point',Button='Left',Remove=False)

Type:'point','cell'or 'world'

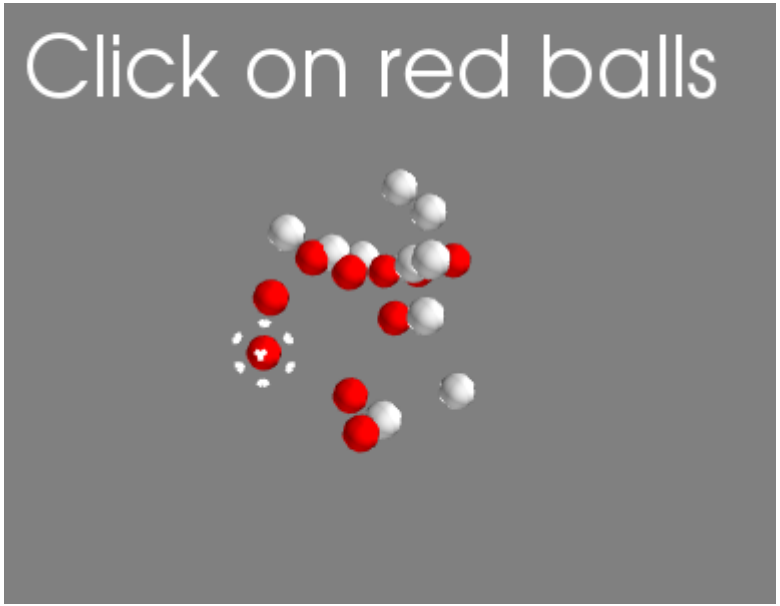
Button:'Left','Middle'or 'Right'

Remove:如果值为True，则callback函数不起作用

返回：一个vtk picker 对象


vtkObject
vtkAbstractPicker
vtkAbstractPropPicker
vtkPicker
vtkCellPicker
vtkPointPicker
vtkResliceCursorPicker
vtkVolumePicker

选取红色小球问题分析



- 建立一个figure
- 随机生成红、白小球
- 初始化红色小球选取外框
- 鼠标选取任意红色小球，外框移动到该小球上（callback）
- 建立on_mouse_pick()响应机制

程序框架

```
# 场景初始化
figure = mlab.gcf()
# 用mlab.points3d建立红色和白色小球的集合
#
… …
# 处理选取事件
def picker_callback(picker)
# 建立响应机制
picker = figure.on_mouse_pick(picker_callback)
mlab.show()
```

小球场景初始化建立

```
# 用mlab.points3d建立红色和白色小球的集合
x1, y1, z1 = np.random.random((3, 10))
red_glyphs = mlab.points3d(x1, y1, z1, color=(1, 0, 0),
                           resolution=10)
x2, y2, z2 = np.random.random((3, 10))
white_glyphs = mlab.points3d(x2, y2, z2, color=(0.9, 0.9, 0.9),
                             resolution=10)
```

选取框初始化建立

```
# 绘制选取框，并放在第一个小球上
outline = mlab.outline(line_width=3)
outline.outline_mode = 'cornered'
outline.bounds = (x1[0] - 0.1, x1[0] + 0.1,
                  y1[0] - 0.1, y1[0] + 0.1,
                  z1[0] - 0.1, z1[0] + 0.1)
```

“选取”回调函数的结构

```
#当选取事件发生时，调用此函数
def picker_callback(picker):
    if picker.actor in red_glyphs.actor.actors:
        # 计算哪个小球被选取
        # 确定该小球的ID
            # 找到与此红色小球相关的坐标
            # 将选取外框移到小球上
```

如何计算哪小球被选取

glyph_points：获取一个小球的顶点坐标列表

```
glyph_points =
red_glyphs.glyph.glyph_source.glyph_source.output.points.to_array()
```

如何计算哪小球被选取

```
def picker_callback(picker):
    if picker.actor in red_glyphs.actor.actors:
        # 计算被选取的小球的ID号
        point_id = int(picker.point_id / glyph_points.shape[0])
```

Picker对像选取的顶点ID　　每一个小球顶点的总数

如何计算哪小球被选取

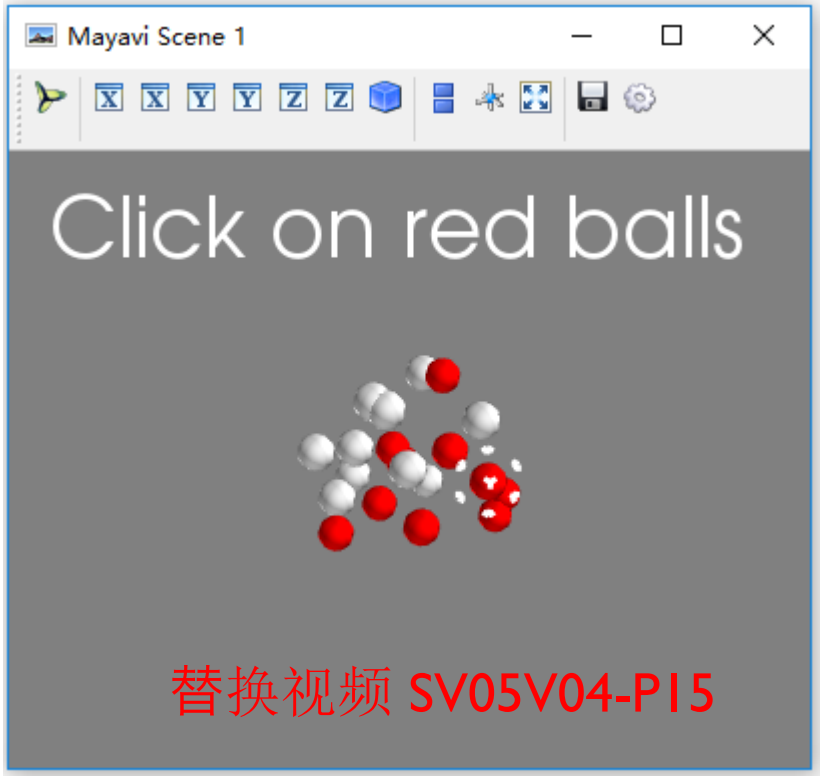
```
def picker_callback(picker):
    if picker.actor in red_glyphs.actor.actors:
        # 计算哪个小球被选取
        point_id = int(picker.point_id / glyph_points.shape[0])
        if point_id != -1:
            # 计算与此红色小球相关的坐标
            x, y, z = x1[point_id], y1[point_id], z1[point_id]
            # 将外框移到小球上
            outline.bounds = (x - 0.1, x + 0.1,
                              y - 0.1, y + 0.1,
                              z - 0.1, z + 0.1)
```

建立响应机制

```
picker = figure.on_mouse_pick(picker_callback)
mlab.title(‘Click on red balls’)#设置窗口的标题文字
mlab.show()
```

```
from mayavi import mlab

######场景初始化######
figure = mlab.gcf()

# 用mlab.points3d建立红色和白色小球的集合
x1, y1, z1 = np.random.random((3, 10))
red_glyphs = mlab.points3d(x1, y1, z1, color=(1, 0, 0),
                           resolution=10)
x2, y2, z2 = np.random.random((3, 10))
white_glyphs = mlab.points3d(x2, y2, z2, color=(0.9, 0.9, 0.9),
                             resolution=10)

# 绘制选取框，并放在第一个小球上
outline = mlab.outline(line_width=3)
outline.outline_mode = 'cornered'
outline.bounds = (x1[0] - 0.1, x1[0] + 0.1,
                  y1[0] - 0.1, y1[0] + 0.1,
                  z1[0] - 0.1, z1[0] + 0.1)

######处理选取事件#####
# 获取构成一个红色小球的顶点列表
glyph_points = red_glyphs.glyph.glyph_source.glyph_source.output.points.to_array()
#当选取事件发生时调用此函数
def picker_callback(picker):
    if picker.actor in red_glyphs.actor.actors:
        # 计算哪个小球被选取
        point_id = int(picker.point_id / glyph_points.shape[0])  # int向下取整
        if point_id != -1:#如果没有小球被选取，则point_id = -1
            # 找到与此红色小球相关的坐标
            x, y, z = x1[point_id], y1[point_id], z1[point_id]
            # 将外框移到小球上
            outline.bounds = (x - 0.1, x + 0.1,
                              y - 0.1, y + 0.1,
                              z - 0.1, z + 0.1)

picker = figure.on_mouse_pick(picker_callback)
mlab.title('Click on red balls')
mlab.show()
```

程序的运行结果



程序框架的优化

程序运行两个问题：

- 小球初始速度太慢
- 鼠标选取不精确

程序框架的优化

```
# 场景初始化
figure = mlab.gcf()
figure.scene.disable_render = True
# 用mlab.points3d建立红色和白色小球的集合
… …
figure.scene.disable_render = False
# 处理选取事件
def picker_callback(picker)
# 建立响应机制
picker = figure.on_mouse_pick(picker_callback)
mlab.show()
```

所有物体全部建立完再绘制！

程序框架的优化

```
# 场景初始化
figure = mlab.gcf()
figure.scene.disable_render = True
# 用mlab.points3d建立红色和白色小球的集合
… …
figure.scene.disable_render = False
# 处理选取事件
def picker_callback(picker)
# 建立响应机制
picker = figure.on_mouse_pick(picker_callback)
Picker.tolerance = 0.01
mlab.show()
```

设置tolerance参数，提高选取精度！