vtkdatawidgets Documentation

Release 0.1.0.dev

Simula Research Laboratory

Feb 27, 2018

Installation and usage

Version: 0.1.0.dev

Jupyter data-widgets based on VTK sources.

CHAPTER 1

Quickstart

To get started with vtkdatawidgets, install with pip:

```
pip install vtkdatawidgets
```

or with conda:

```
conda install vtkdatawidgets
```

CHAPTER 2

Contents

2.1 Installation

The simplest way to install vtkdatawidgets is via pip:

```
pip install vtkdatawidgets
```

or via conda:

```
conda install vtkdatawidgets
```

If you installed via pip, and notebook version < 5.3, you will also have to install / configure the front-end extension as well. If you are using classic notebook (as opposed to Jupyterlab), run:

```
jupyter nbextension install [--sys-prefix / --user / --system] --py vtkdatawidgets

jupyter nbextension enable [--sys-prefix / --user / --system] --py vtkdatawidgets
```

with the appropriate flag. If you are using Jupyterlab, install the extension with:

```
jupyter labextension install jupyter-vtk-datawidgets
```

If you are installing using conda, these commands should be unnecessary, but If you need to run them the commands should be the same (just make sure you choose the *–sys-prefix* flag).

2.2 Introduction

2.3 Examples

This section contains several examples generated from Jupyter notebooks. The widgets have been embedded into the page for demonstrative pruposes.

2.3.1 Introduction

Jupyter - VTK bridge

```
In [1]: import vtk
        from vtkdatawidgets.vtk_binding import VtkJupyterBridge
```

First we set up our bridge object:

```
In [2]: bridge = VtkJupyterBridge()
```

This object is a VTK sink (algorithm with 1 input, 0 outputs), that translates the input data to a jupyter widget representation on `RequestData` (i.e. when `.Update()` is called).

Let's set up some placeholder VTK source (a simple sphere in this case), and set it as the input of our bridge:

```
In [3]: sphere = vtk.vtkSphereSource()
        bridge.SetInputConnection(sphere.GetOutputPort())
        bridge.Update()
```

```
C:\cleanconda\envs\vtk\lib\site-packages\ipydatawidgets\ndarray\serializers.py:62: UserWarning: Canno
  warnings.warn('Cannot serialize (u)int64 data, Javascript does not support it. '
```

At the `.Update()` call, the data is serialized and sent to the browser. There we can use it however we want. For now, let us visualize it using a simple vtk.js based rendering. This will set up a similar bridge in the browser: from widgets to a vtk.js data source (0 inputs, 1 output):

```
In [4]: from vtkdatawidgets import VtkRenderer
        renderer = VtkRenderer(dataset=bridge.widget, background=(0.5, 0, 0), size=(600, 400))
        renderer
```

```
VtkRenderer(background=(0.5, 0.0, 0.0), dataset=MutableDataSet(containers=(DataContainer(attributes=
       [ 0.0000000e+00,  0.0000000e+00, -1.0000000e+00],
       [ 4.3388373e-01,  0.0000000e+00,  9.0096885e-01],
       [ 7.8183150e-01,  0.0000000e+00,  6.2348980e-01],
       [ 9.7492790e-01,  0.0000000e+00,  2.2252093e-01],
       [ 9.7492790e-01,  0.0000000e+00, -2.2252093e-01],
       [ 7.8183150e-01,  0.0000000e+00, -6.2348980e-01],
       [ 4.3388373e-01,  0.0000000e+00, -9.0096885e-01],
       [ 3.0680212e-01,  3.0680212e-01,  9.0096885e-01],
       [ 5.5283833e-01,  5.5283833e-01,  6.2348980e-01],
       [ 6.8937814e-01,  6.8937814e-01,  2.2252093e-01],
       [ 6.8937814e-01,  6.8937814e-01, -2.2252093e-01],
       [ 5.5283833e-01,  5.5283833e-01, -6.2348980e-01],
       [ 3.0680212e-01,  3.0680212e-01, -9.0096885e-01],
       [ 2.6567716e-17,  4.3388373e-01,  9.0096885e-01],
       [ 4.7873372e-17,  7.8183150e-01,  6.2348980e-01],
       [ 5.9697115e-17,  9.7492790e-01,  2.2252093e-01],
       [ 5.9697115e-17,  9.7492790e-01, -2.2252093e-01],
       [ 4.7873372e-17,  7.8183150e-01, -6.2348980e-01],
       [ 2.6567716e-17,  4.3388373e-01, -9.0096885e-01],
       [-3.0680212e-01,  3.0680212e-01,  9.0096885e-01],
       [-5.5283833e-01,  5.5283833e-01,  6.2348980e-01],
       [-6.8937814e-01,  6.8937814e-01,  2.2252093e-01],
       [-6.8937814e-01,  6.8937814e-01, -2.2252093e-01],
       [-5.5283833e-01,  5.5283833e-01, -6.2348980e-01],
       [-3.0680212e-01,  3.0680212e-01, -9.0096885e-01],
       [-4.3388373e-01,  5.3135432e-17,  9.0096885e-01],
       [-7.8183150e-01,  9.5746745e-17,  6.2348980e-01],
       [-9.7492790e-01,  1.1939423e-16,  2.2252093e-01],
       [-9.7492790e-01,  1.1939423e-16, -2.2252093e-01],
       [-7.8183150e-01,  9.5746745e-17, -6.2348980e-01],
```

```
[-4.3388373e-01,  5.3135432e-17, -9.0096885e-01],
[-3.0680212e-01, -3.0680212e-01,  9.0096885e-01],
[-5.5283833e-01, -5.5283833e-01,  6.2348980e-01],
[-6.8937814e-01, -6.8937814e-01,  2.2252093e-01],
[-6.8937814e-01, -6.8937814e-01, -2.2252093e-01],
[-5.5283833e-01, -5.5283833e-01, -6.2348980e-01],
[-3.0680212e-01, -3.0680212e-01, -9.0096885e-01],
[-7.9703147e-17, -4.3388373e-01,  9.0096885e-01],
[-1.4362011e-16, -7.8183150e-01,  6.2348980e-01],
[-1.7909135e-16, -9.7492790e-01,  2.2252093e-01],
[-1.7909135e-16, -9.7492790e-01, -2.2252093e-01],
[-1.4362011e-16, -7.8183150e-01, -6.2348980e-01],
[-7.9703147e-17, -4.3388373e-01, -9.0096885e-01],
[ 3.0680212e-01, -3.0680212e-01,  9.0096885e-01],
[ 5.5283833e-01, -5.5283833e-01,  6.2348980e-01],
[ 6.8937814e-01, -6.8937814e-01,  2.2252093e-01],
[ 6.8937814e-01, -6.8937814e-01, -2.2252093e-01],
[ 5.5283833e-01, -5.5283833e-01, -6.2348980e-01],
[ 3.0680212e-01, -3.0680212e-01, -9.0096885e-01]], dtype=float32), name='Normals'),), kind='Po
[ 0.0000000e+00,  0.0000000e+00, -5.0000000e-01],
[ 2.1694186e-01,  0.0000000e+00,  4.5048442e-01],
[ 3.9091575e-01,  0.0000000e+00,  3.1174490e-01],
[ 4.8746395e-01,  0.0000000e+00,  1.1126047e-01],
[ 4.8746395e-01,  0.0000000e+00, -1.1126047e-01],
[ 3.9091575e-01,  0.0000000e+00, -3.1174490e-01],
[ 2.1694186e-01,  0.0000000e+00, -4.5048442e-01],
[ 1.5340106e-01,  1.5340106e-01,  4.5048442e-01],
[ 2.7641916e-01,  2.7641916e-01,  3.1174490e-01],
[ 3.4468907e-01,  3.4468907e-01,  1.1126047e-01],
[ 3.4468907e-01,  3.4468907e-01, -1.1126047e-01],
[ 2.7641916e-01,  2.7641916e-01, -3.1174490e-01],
[ 1.5340106e-01,  1.5340106e-01, -4.5048442e-01],
[ 1.3283858e-17,  2.1694186e-01,  4.5048442e-01],
[ 2.3936686e-17,  3.9091575e-01,  3.1174490e-01],
[ 2.9848558e-17,  4.8746395e-01,  1.1126047e-01],
[ 2.9848558e-17,  4.8746395e-01, -1.1126047e-01],
[ 2.3936686e-17,  3.9091575e-01, -3.1174490e-01],
[ 1.3283858e-17,  2.1694186e-01, -4.5048442e-01],
[-1.5340106e-01,  1.5340106e-01,  4.5048442e-01],
[-2.7641916e-01,  2.7641916e-01,  3.1174490e-01],
[-3.4468907e-01,  3.4468907e-01,  1.1126047e-01],
[-3.4468907e-01,  3.4468907e-01, -1.1126047e-01],
[-2.7641916e-01,  2.7641916e-01, -3.1174490e-01],
[-1.5340106e-01,  1.5340106e-01, -4.5048442e-01],
[-2.1694186e-01,  2.6567716e-17,  4.5048442e-01],
[-3.9091575e-01,  4.7873372e-17,  3.1174490e-01],
[-4.8746395e-01,  5.9697115e-17,  1.1126047e-01],
[-4.8746395e-01,  5.9697115e-17, -1.1126047e-01],
[-3.9091575e-01,  4.7873372e-17, -3.1174490e-01],
[-2.1694186e-01,  2.6567716e-17, -4.5048442e-01],
[-1.5340106e-01, -1.5340106e-01,  4.5048442e-01],
[-2.7641916e-01, -2.7641916e-01,  3.1174490e-01],
[-3.4468907e-01, -3.4468907e-01,  1.1126047e-01],
[-3.4468907e-01, -3.4468907e-01, -1.1126047e-01],
[-2.7641916e-01, -2.7641916e-01, -3.1174490e-01],
[-1.5340106e-01, -1.5340106e-01, -4.5048442e-01],
[-3.9851573e-17, -2.1694186e-01,  4.5048442e-01],
[-7.1810057e-17, -3.9091575e-01,  3.1174490e-01],
[-8.9545676e-17, -4.8746395e-01,  1.1126047e-01],
```

```
       [-8.9545676e-17, -4.8746395e-01, -1.1126047e-01],
       [-7.1810057e-17, -3.9091575e-01, -3.1174490e-01],
       [-3.9851573e-17, -2.1694186e-01, -4.5048442e-01],
       [ 1.5340106e-01, -1.5340106e-01,  4.5048442e-01],
       [ 2.7641916e-01, -2.7641916e-01,  3.1174490e-01],
       [ 3.4468907e-01, -3.4468907e-01,  1.1126047e-01],
       [ 3.4468907e-01, -3.4468907e-01, -1.1126047e-01],
       [ 2.7641916e-01, -2.7641916e-01, -3.1174490e-01],
       [ 1.5340106e-01, -1.5340106e-01, -4.5048442e-01]], dtype=float32), name='Points'),), kind='Po
      26, 32,  0,  3, 32, 38,  0,  3, 38, 44,  0,  3, 44,  2,  0,  3,  7,
       1, 13,  3, 13,  1, 19,  3, 19,  1, 25,  3, 25,  1, 31,  3, 31,  1,
      37,  3, 37,  1, 43,  3, 43,  1, 49,  3, 49,  1,  7,  3,  2,  3,  9,
       3,  2,  9,  8,  3,  3,  4, 10,  3,  3, 10,  9,  3,  4,  5, 11,  3,
       4, 11, 10,  3,  5,  6, 12,  3,  5, 12, 11,  3,  6,  7, 13,  3,  6,
      13, 12,  3,  8,  9, 15,  3,  8, 15, 14,  3,  9, 10, 16,  3,  9, 16,
      15,  3, 10, 11, 17,  3, 10, 17, 16,  3, 11, 12, 18,  3, 11, 18, 17,
       3, 12, 13, 19,  3, 12, 19, 18,  3, 14, 15, 21,  3, 14, 21, 20,  3,
      15, 16, 22,  3, 15, 22, 21,  3, 16, 17, 23,  3, 16, 23, 22,  3, 17,
      18, 24,  3, 17, 24, 23,  3, 18, 19, 25,  3, 18, 25, 24,  3, 20, 21,
      27,  3, 20, 27, 26,  3, 21, 22, 28,  3, 21, 28, 27,  3, 22, 23, 29,
       3, 22, 29, 28,  3, 23, 24, 30,  3, 23, 30, 29,  3, 24, 25, 31,  3,
      24, 31, 30,  3, 26, 27, 33,  3, 26, 33, 32,  3, 27, 28, 34,  3, 27,
      34, 33,  3, 28, 29, 35,  3, 28, 35, 34,  3, 29, 30, 36,  3, 29, 36,
      35,  3, 30, 31, 37,  3, 30, 37, 36,  3, 32, 33, 39,  3, 32, 39, 38,
       3, 33, 34, 40,  3, 33, 40, 39,  3, 34, 35, 41,  3, 34, 41, 40,  3,
      35, 36, 42,  3, 35, 42, 41,  3, 36, 37, 43,  3, 36, 43, 42,  3, 38,
      39, 45,  3, 38, 45, 44,  3, 39, 40, 46,  3, 39, 46, 45,  3, 40, 41,
      47,  3, 40, 47, 46,  3, 41, 42, 48,  3, 41, 48, 47,  3, 42, 43, 49,
       3, 42, 49, 48,  3, 44, 45,  3,  3, 44,  3,  2,  3, 45, 46,  4,  3,
      45,  4,  3,  3, 46, 47,  5,  3, 46,  5,  4,  3, 47, 48,  6,  3, 47,
       6,  5,  3, 48, 49,  7,  3, 48,  7,  6], dtype=int64), name='cells'),), kind='Polys')), kind=
```

The `VtkRenderer` object is a standard Jupyter widget that we connect up to the widget side of the bridge (`bridge.widget`). It does not understand native VTK objects.

Being a widget, the `VtkRenderer` is an interactive object. Setting its properties are immediately reflected in the frontend:

```
In [5]: import ipywidgets
        picker = ipywidgets.ColorPicker(value='darkred')
        ipywidgets.jslink((picker, 'value'), (renderer, 'background'))
        picker
```

```
ColorPicker(value='darkred')
```

Similarly, we can create a new bridge (this time from a cone source), and set that as the input of the renderer:

```
In [ ]: cone = vtk.vtkConeSource()
        bridge2 = VtkJupyterBridge()
        bridge2.SetInputConnection(cone.GetOutputPort())
        bridge2.Update()
        renderer.dataset = bridge2.widget
```

Or, we can simply change the input of the current bridge:

```
In [ ]: bridge2.SetInputConnection(sphere.GetOutputPort())
        bridge2.Update()
```

Notice that nothing will happen until you call `Update()` on the bridge, as VTK only produces the data on request.

```
In [ ]: bridge2.SetInputConnection(cone.GetOutputPort())
```

```
In [ ]: bridge2.Update()
```

In the future, there will likely be support for front-end driven update calls, but for now this is unsupported.

Further notes:

Note that VTK.js currently only supports: - PolyData - ImageData

So if you need to transform any inputs you have to those formats before passing it to the bridge.

2.4 Developer install

To install a developer version of vtkdatawidgets, you will first need to clone the repository:

```
git clone https://github.com/vidartf/jupyter-vtk-datawidgets
cd jupyter-vtk-datawidgets
```

Next, install it with a develop install using pip:

```
pip install -e .
```

If you are planning on working on the JS/frontend code, you should also do a link installation of the extension:

```
jupyter nbextension install [--sys-prefix / --user / --system] --symlink --py␣
↪vtkdatawidgets

jupyter nbextension enable [--sys-prefix / --user / --system] --py vtkdatawidgets
```

with the appropriate flag. Or, if you are using Jupyterlab:

```
jupyter labextension install .
```