Vispy – High Level API

Roman Weis

29.08.2017

Overview

- Introduction to vispy
- Examples: high level API
- Vispy in PyQt applications
- Examples: custom plots
- Conclusions

Full code for examples: https://github.com/weisro/MeetVispy

What is vispy

- http://vispy.org/
- From the website:
  1. “**Users knowing OpenGL**, or willing to learn OpenGL, who want to create beautiful and fast interactive 2D/3D visualizations in Python as easily as possible.”
  2. “**Scientists without any knowledge of OpenGL**, who are seeking a high-level, high-performance plotting toolkit. […] (**WARNING**: experimental / developing code).”
- Code/examples: https://github.com/vispy/vispy
- Install: pip install –e git+https://github.com/vispy/vispy#egg=vispy-dev

The layers of vispy

High level API

vispy.plot

- high level API similar to matplotlib
- Qt, GLFW, SDL2, Wx, or Pyglet as backend
- Pre-defined PlotWidget

vispy.scene

- hierarchy of visual objects
- interactive objects like Line, Box, Rectangle etc.
- embed charts into PyQt apps through the SceneCanvas class

Low level API

vispy.visuals

- Visual objects that can be used with higher level modules
- Custom objects

vispy.gloo

- Object oriented interface to OpenGL
- Expertise in OpenGL required
- Expertise in GLSL required

vispy.plot

```
1    #full code in examples/plot/plot_1.6m.py
2
3    from vispy import plot as vp
4    from vispy.color import get_colormap
5
6    #create plot widget
7    fig = vp.Fig(size=(800, 800), show=False)
8
9    #create a line plot
10   line = fig[0, 0].plot((x, y),
11                         width=3,
12                         title='~1.6 mill. points',
13                         xlabel='Time',
14                         ylabel='Price',
15                         color=colors[j])
16
17   # add grid to plot
18   grid = vp.visuals.GridLines(color=(0, 0, 0, 0.5))
19   grid.set_gl_state('translucent')
20   fig[0, 0].view.add(grid)
21
22   # console output fps measure
23   fig.measure_fps()
24
25   if __name__ == '__main__':
26       fig.show(run=True)
27
28
```

PyQt

- Python binding for Qt (https://www.qt.io/)
- Use of Qt Designer or directly python code
- ui xml code from Designer automatically translated to python code
- Rapid GUI prototyping

- Design ui in the «Designer»
- Use ui in the code

```
1   import sys
2   from PyQt5 import uic
3
4   Ui_MainWindow, MainWindowBase = uic.loadUiType('MainWindow.ui')
5
6   class MainWindow(MainWindowBase):
7       def __init__(self):
8           MainWindowBase.__init__(self)
9           self.ui = Ui_MainWindow()
10          self.ui.setupUi(self)
11
12  if __name__ == "__main__":
13      app = QtWidgets.QApplication(sys.argv)
14      mw = MainWindow()
15      mw.show()
16      sys.exit(app.exec_())
17
18
```

Vispy with PyQt and vispy.plot Example 01

- Fig extends SceneCanvas ➔ we can embed it directly into our PyQt app

  (with a little trick)

```
#full code in examples/gui/01_main_window.py
class MainWindow(MainWindowBase):
    def __init__(self):
        MainWindowBase.__init__(self)
        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)

        self.viewer = vp.Fig()
        # create native pyqt widget
        self.viewer.create_native()
        #use the native widget
        self.ui.plotFrameLayout.addWidget(self.viewer.native)


```

Vispy with PyQt – Custom PlotWidget Example 02

- Instead of Fig we can use SceneCanvas directly
  - Lets have a look at Fig code first

```
1    #see Fig class in vispy for full code
2    class Fig(SceneCanvas):
3
4        def __init__(self, bgcolor='w', size=(800, 600), show=True, **kwargs):
5            self._plot_widgets = []
6            self._grid = None  # initialize before the freeze occurs
7            super(Fig, self).__init__(bgcolor=bgcolor, keys='interactive',
8                                      show=show, size=size, **kwargs)
9            self._grid = self.central_widget.add_grid()
10           self._grid._default_class = PlotWidget
11
12
```

- Use same approach and just swap the PlotWidget with a custom implementation

Vispy with PyQt – Custom PlotWidget Example 02

- Lets check PlotWidget code first

```
1     #see PlotWidget class in vispy for full code
2     from ..import scene
3
4     class PlotWidget(scene.Widget):
5         def __init__(self, *args, **kwargs):...
28
29        def _configure_2d(self, fg_color=None):...
128
129       def histogram(self, data, bins=10, color='w', orientation='h')
130
131       def image(self, data, cmap='cubehelix', clim='auto', fg_color=None)
132
133       def plot(self, data, color='k', symbol=None, line_kind='-', width=1.,
134                marker_size=10., edge_color='k', face_color='b', edge_width=1.,
135                title=None, xlabel=None, ylabel=None):
136           self._configure_2d()
137           line = scene.LinePlot(data, connect='strip', color=color,
138                                 symbol=symbol, line_kind=line_kind,
139                                 width=width, marker_size=marker_size,
140                                 edge_color=edge_color,
141                                 face_color=face_color,
142                                 edge_width=edge_width)
143           self.view.add(line)
144           self.view.camera.set_range()
145           self.visuals.append(line)
```

Vispy with PyQt – Custom PlotWidget Example 02

- CustomPlotWidget

```
1    #see examples/gui/custom_plot_widget.py
2    class CustomPlotWidget(scene.Widget):
3        def __init__(self, *args, **kwargs):
4            self.grid = None
5            self.section_y_x = None
6            super(CustomPlotWidget, self).__init__(*args, **kwargs)
7            self.grid = self.add_grid(spacing=0, margin=10)
8
9        def configure(self, xlabel=None, ylabel=None)
10
11       def plot(self, data, color='k', width=1.0, kind="line"):
12           obj = None
13           if kind == "line":
14               obj = self._plot_line(color, data, width)
15           if kind =="candle_stick":
16               obj = self._plot_candle_sticks(data)
17           return obj
18
19       def _plot_line(self, color, data, width):
20           line = scene.LinePlot(data, connect='strip', color=color, width=width)
21           self.view.add(line)
22           self.view.camera.set_range()
23           return line
24
```

Vispy with PyQt – Plot Controls Example 03

- Toggle visibility of line plots witch check boxes

```
1    self.list_view_model = QtGui.QStandardItemModel()
2
3    for key in data_model.keys():
4        item = QtGui.QStandardItem(key)
5        item.setCheckable(True)
6        item.setEditable(False)
7        self.list_view_model.appendRow(item)
8
9    self.list_view_model.itemChanged.connect(self.on_item_changed)
10
11   self.line_plots = {}
12   def on_item_changed(self, item):
13       key = item.text()
14       if item.checkState() == QtCore.Qt.Checked:
15           if key not in self.line_plots:
16               line = self.chart.plot((data_model[key]["x"], data_model[key]["y"]))
17               self.line_plots[key] = line
18            else:
19               self.line_plots[key].visible = True
20        else:
21           if key in self.line_plots:
22               self.line_plots[key].visible = False
23
```

Vispy with PyQt – Picking Example 04

- Toggle visibility of line plots witch check boxes

```
1    line = self.chart.plot((data_model[key]["x"], data_model[key]["y"]))
2    line.interactive = True
3
4    self.viewer.connect(self.on_mouse_press)
5    self.viewer.connect(self.on_mouse_move)
6
7    def on_mouse_press(self, event):
8        if event.handled or event.button != 1:
9            return
10       if self.selected_line is not None:
11           self.selected_line.set_data(color="k")
12           self.selected_line = None
13           return
14       for v in self.viewer.visuals_at(event.pos):
15           if isinstance(v, scene.visuals.LinePlot):
16               self.selected_line = v
17               break
18       if self.selected_line is not None:
19           self.selected_line.set_data(color="r")
20
21   def on_mouse_move(self, event):
22       if self.selected_line is None:
```

Vispy with PyQt – Custom Plot Example 05

- Combine Visuals to create a custom Plot
- Candlestick chart consists of:
  - Rectangle (vispy.scene.visuals.Rectangle)
  - Line high (vispy.scene.visuals.Line)
  - Line low (vispy.scene.visuals.Line)

```
1   def _plot_candle_stick(self, data):
2       time = data["time"]
3       open = data["open"]
4       high = data["high"]
5       low = data["low"]
6       close = data["close"]
7
8       width = data["time_frame"] * 0.8
9
10      bullish = close > open
11      height = abs(open - close)
12      center = (time, open + height * 0.5 if bullish else open - height * 0.5)
```

Conclusions

- Vispy provides a high level API to OpenGL
- Enables engineers and scientists w/o OpenGL expertise to visualize large data sets
- Straight forward integration into PyQt apps

- Relatively new library, still experimental code
- Drawbacks in performance when combining visuals through high level API

- Vispy encourages to learn OpenGL and GLSL

Questions?