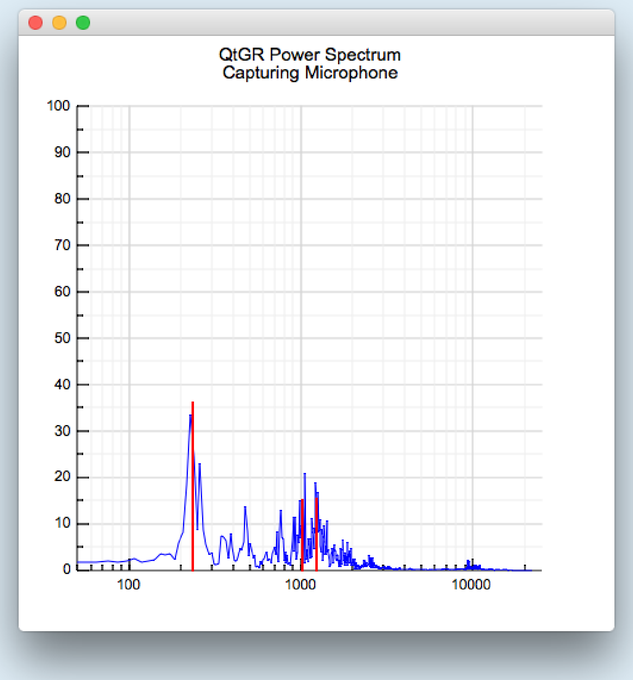
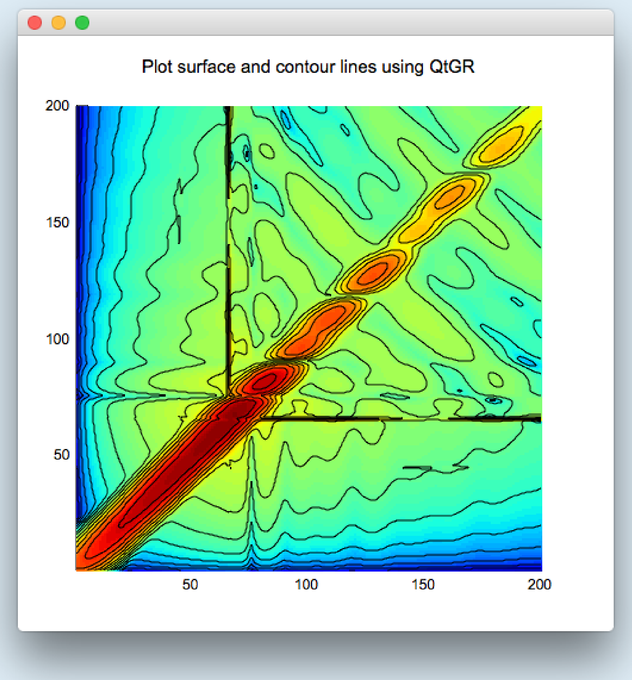
Embedding Visualization Applications with PyGR

Object oriented interfaces and PyQt/PySide bindings for GR - a universal framework for visualization applications

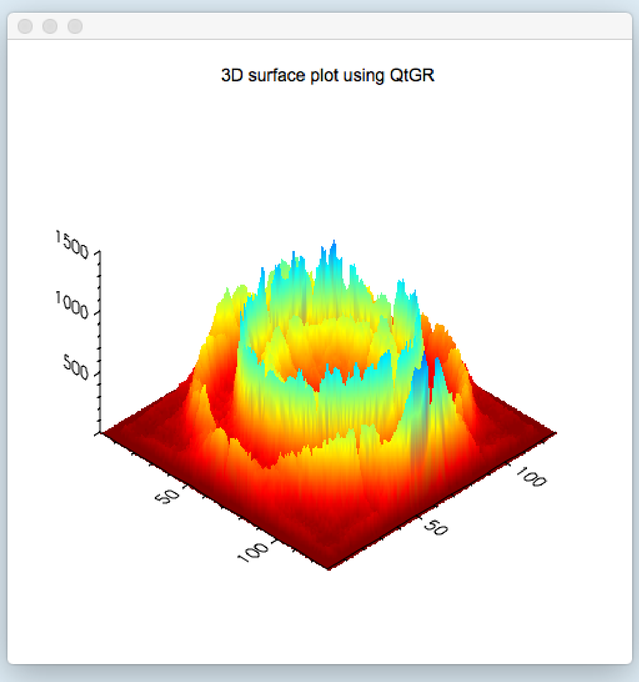
Georg Brandl, Christian Felder, Josef Heinen



(a) Microphone Power Spectrum and Peaks (computed using FFT)



(b) Surface Plot with Contour Lines



(c) 3D Surface Plot

GR

- GRaphics Library optimized for real-time data visualizations
- Procedural graphics backend (written in C)
  - Does not rely on creation of figures

  ⇒ Representation of continuous data streams
- Based on a Graphical Kernel System
  - Device and Platform independent API
  - Large scale of logical device drivers/plugins:
    - Qt, wxWidgets
    - X11, Quartz, Win32
    - PostScript, PDF, SVG
    - GS (BMP, JPEG, PNG, TIFF)
    - MOV (MPEG4)
    - HTML5
- Builtin support for 2D plotting and OpenGL (GR3)

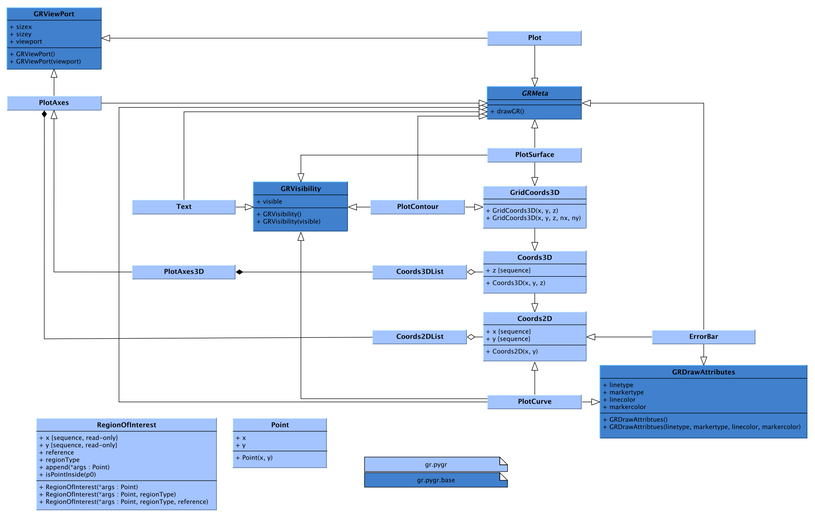
  ⇒ Coexistent 2D and 3D world

QtGR

- Qt Widgets for drawing and interacting with GR
- Specialized Qt Events, e. g. Mouse Events in different coordinate systems (Normalized-, Device- and World Coordinates)
  - MouseEvent
  - WheelEvent
  - PickEvent
  - ROIEvent
  - LegendEvent
- Support for PyQt and PySide

PyGR

- Object oriented interface on top of GR
- Convenience functions for
  - Zooming
  - Panning
  - Selecting a Region
  - Detecting predefined *Regions Of Interest* (point in polygon test)
- Independent of GUI Toolkits



- Manipulating object states instead of fiddling with procedure calls
- Adding new functionality using derived objects, e.g.
  - Clamp minimum values for selecting a *Region of Interest* ($x_{min}, y_{min} > 0$)

```
class ContourAxes(PlotAxes):

    def setWindow(self, xmin, xmax, ymin, ymax):
        if xmin < 1:
            xmin = 1
        if ymin < 1:
            ymin = 1
        return PlotAxes.setWindow(self, xmin, xmax, ymin, ymax)
```

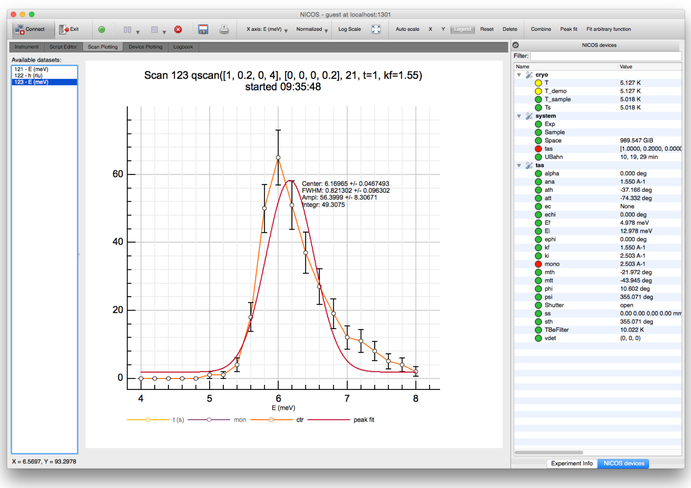
  - Drawing a bar at a given position x_i, y_i to indicate a peak position

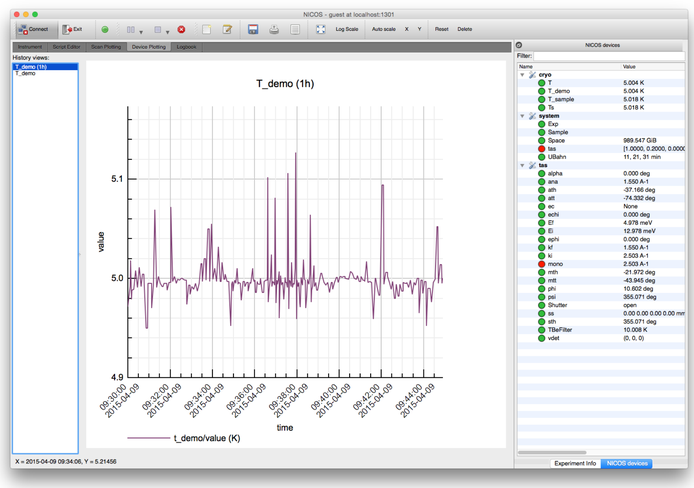
```
class PeakBars(PlotCurve):

    def drawGR(self):
        if self.linetype is not None and self.x:
            # preserve old values
            lcolor = gr.inqlinecolorind()
            gr.setlinewidth(2)
            gr.setlinecolorind(self.linecolor)

            for xi, yi in zip(self.x, self.y):
                gr.polyline([xi, xi], [0, yi])

            # restore old values
            gr.setlinecolorind(lcolor)
            gr.setlinewidth(1)
```

  - Hide/Unhide dependent objects

```
class DependentPlotCurve(PlotCurve):

    # Constructor...
    # dependent property...

    # pylint: disable=W0221
    @PlotCurve.visible.setter
    def visible(self, flag):
        PlotCurve.visible.__set__(self, flag)
        for dep in self.dependent:
            dep.visible = flag

    def drawGR(self):
        PlotCurve.drawGR(self)
        for dep in self.dependent:
            if dep.visible:
                dep.drawGR()
```

GR has been integrated into NICOS, a network-based experiment and instrument control system used for neutron scattering experiments at FRM II in Munich.





NICOS
Network Instrument COntrol System

- Replaced existing Panels
  - based on PyQwt (unmaintained)
- Auto scaling in x- and/or y- direction
  - Adjust axes to fit all shown curves including error bars
  - Adjust in respect to user specified region of interests

    → Adjust axes in altering direction only
- Support for different x axes
- Fitting Peaks or arbitrary functions
  - Picking of fit parameters

- Logarigthmic scale
- Dynamic tick marks for different time domains
- Legend items for curves
  - Un/hide curve and dependencies, e.g. error bars or fit parameters
- General purpose functions:
  - Zooming into a specific point (Mouse wheel zoom)
  - Panning
  - Selecting a *Region of Interest*

Ongoing Projects

- General purpose 3D plotting objects in PyGR
- Integrated live view for three-dimensional data
- Generation of SVG plots for HTML status monitor

Contact: c.felder@fz-juelich.de **- Website:** www.fz-juelich.de, gr-framework.org