

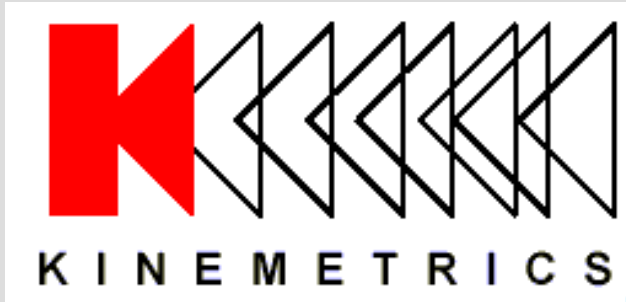
Python/Qt Graphics in Antelope

Danny Harvey
Boulder Real Time Technologies, Inc.
Antelope User Group Meeting, DPC, Rome
2016 May

Outline

- **Introduction**
- **Review of Qt Graphics Introduced in 5.5**
- **Rewrite of Qt Graphics for 5.6**
- **Python-Qt Bridge Development**
- **Coding Examples**
- **Plans for Further Development**

INTRODUCTION – KMI TEAM



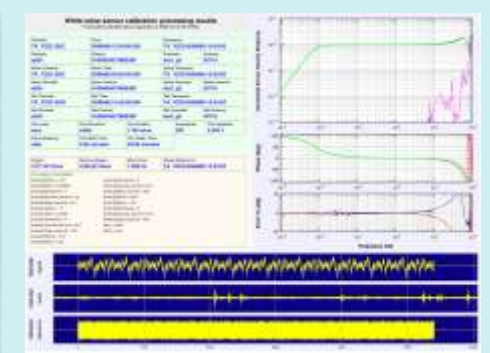
Designs and manufactures sensors and digitizers – Provides complete systems design, installation and operations



Designs High-End Digitizers



Designs High-End Sensors

A screenshot of a data table with multiple columns and rows, showing numerical data and status indicators.

Kinematics/BRTT – Comprehensive hardware, software and services

Kinematics Systems Solutions

- Turnkey complete systems including enterprise-class computing centers and full communications

Kinematics Hardware Manufacturer

- World class Kinematics and Quanterra dataloggers
- World class Kinematics, Metrozet and Streckeisen sensors

BRTT Software Developer

- World class acquisition software for all Kinematics hardware products
- Proven track record for large networks with difficult remote deployments (USArray)
- World class, comprehensive automated and interactive seismic processing software
- Data neutral architecture for support of non-seismic environmental monitoring networks
- Extraordinary Command & Control capabilities with SOH displaying

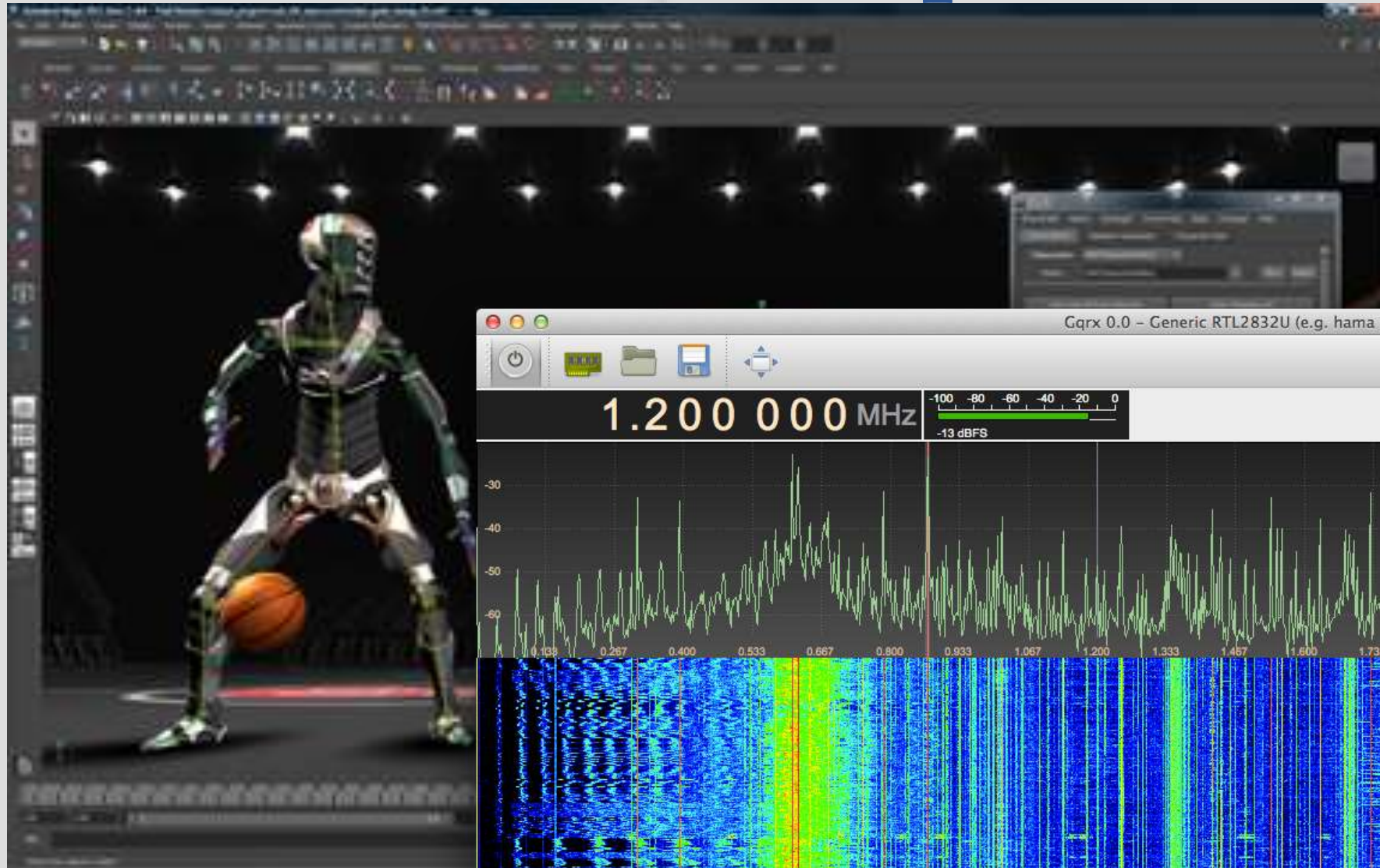
Kinematics Services

- Complete systems procurement, installation and training including all aspects of both hardware and software
- Network operations

Qt Graphics in 5.5

- Cross platform API (MacOSX, LINUX, Windows, iOS, Android, Windows Mobile)
- Commercially supported and licensed (Qt Company)
- High level support for modern graphics hardware (fonts, spatial antialiasing, alpha blending, 3D rendering, etc.)
- Very large user base (Nokia, KDE, Android apps, embedded devices) plus sophisticated extensions such as Marble
- QTWebkit and QTWebsockets plus XML interpreter
- Up to OpenGL API levels

Qt Graphics in 5.5



What is Qt?

- Graphics/Interaction middleware
- C++ API with ~500 classes
- High performance at various levels
- High functionality at various levels
- Cross platform API with common application code base for MacOSX/Cocoa, MacOSX/Xquartz/X11, Linux/X11, iOS, Android, Windows
- Both GPL and commercially licensed through Qt Company

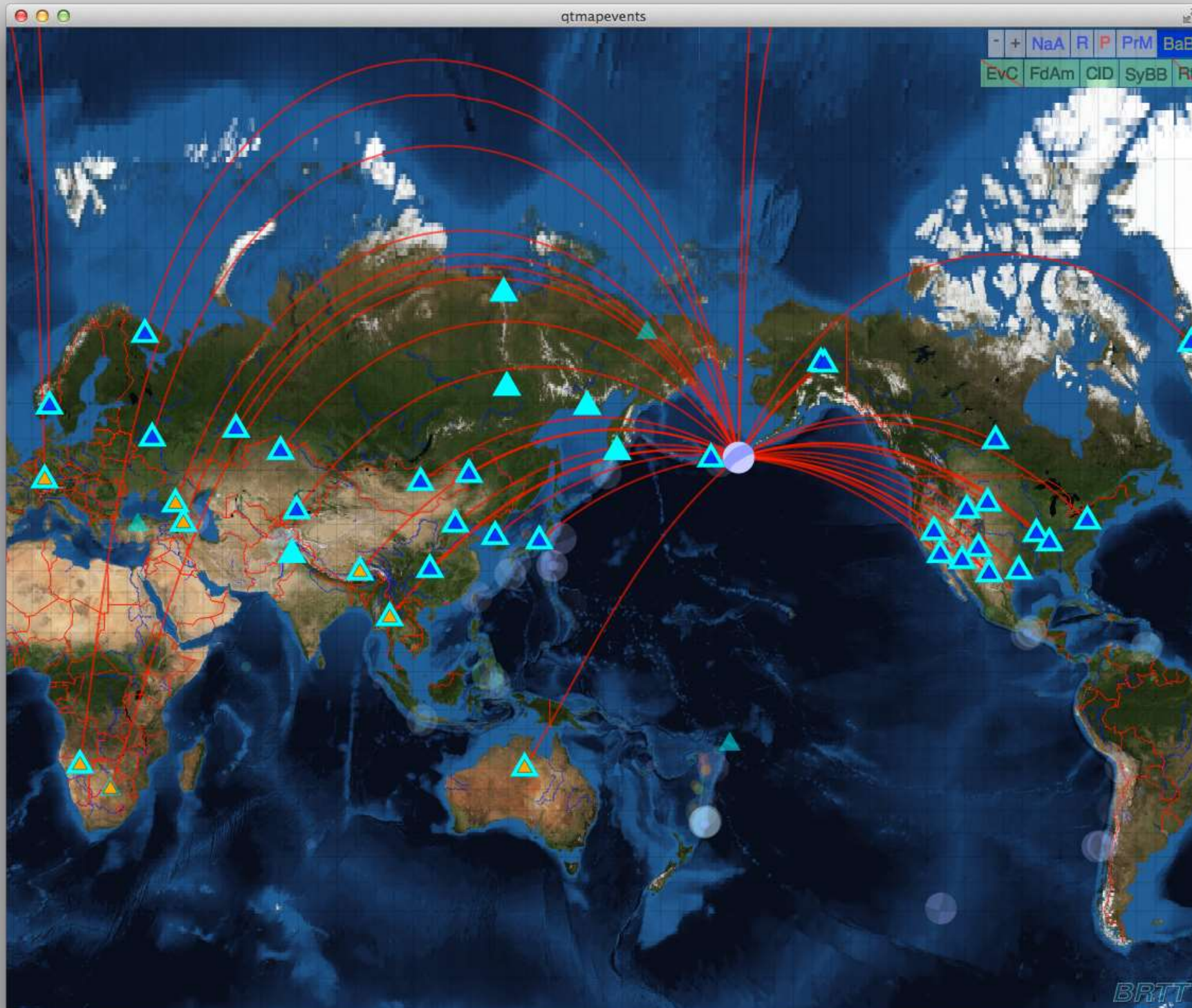
5.5 Graphics Development

- BRTT stopped all graphics/GUI development that uses X11/Tk. This included the TCL, perl and python extensions we have used and developed in the past.
- Starting with 5.5, new graphics/GUI software will be developed only using Qt
- Although there is a dual GPU/commercial PyQt python extension library for Qt, BRTT will not use PyQt for the 5.5 release (we have experimented with making our own version of PyQt)
- New BRTT developed graphics/GUI software written in c++

Qt-related Developments Introduced in 5.5

- New Qt-based library that introduces BRTT plot extensions into Qt (not available for development by our users)
- New Qt-based `dbe` prototype
- Rewrite of BRTT map display software
- Support for continuously scalable display transformations of image data such as NASA's Bluemarble earth image data
- High performance map projection transformations through threading
- New BRTT Map Data (`bmd`) format that supports multiple resolution and tiled image and vector data in both native compressed and uncompressed formats
- `qtmapevents`

5.5 - qtmapevents



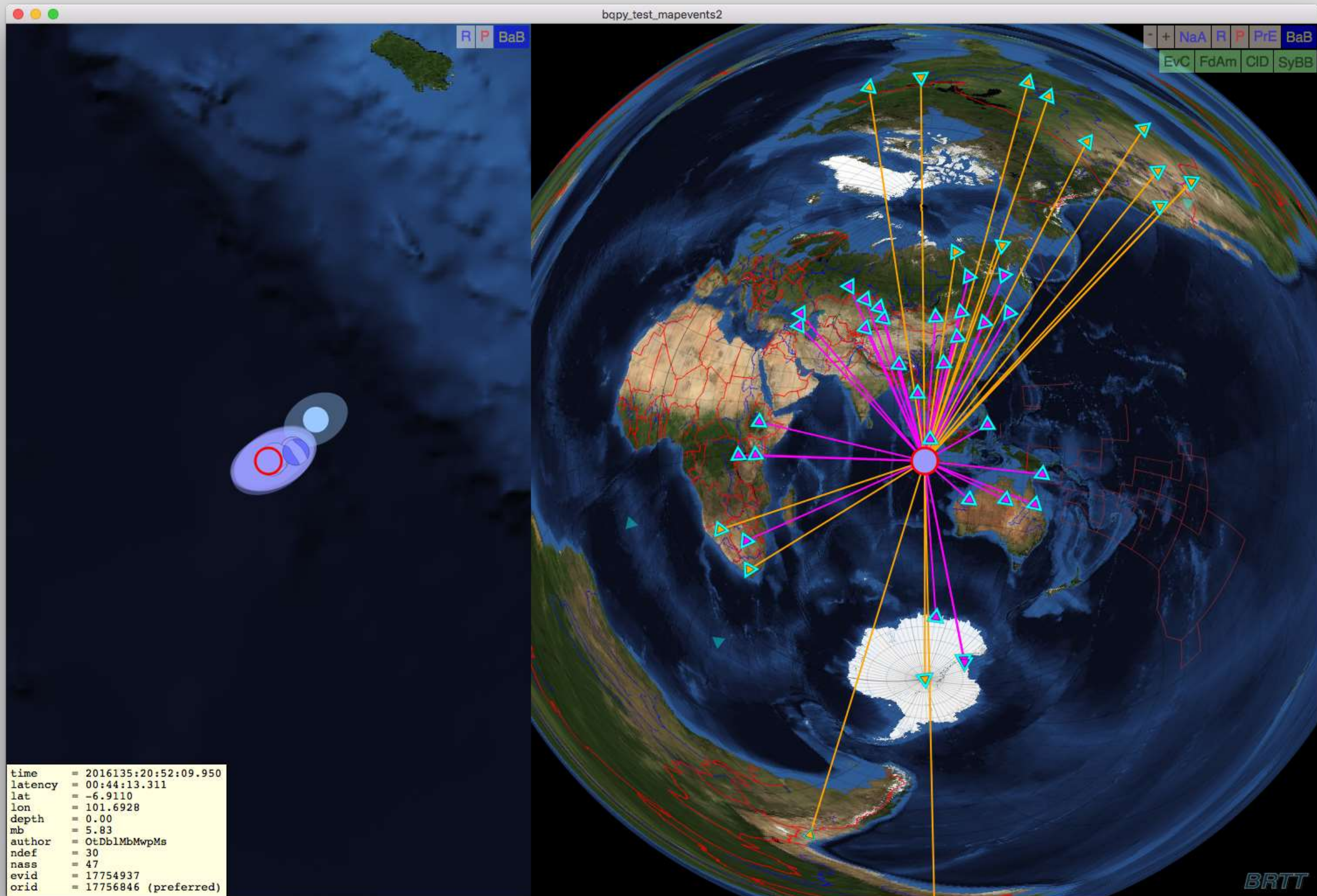
5.5 - qtmapevents

- 180 lines of c++ closed-source code
- Because of commercial Qt licensing restrictions, no user access to BRTT-developed Qt library
- The 5.5 prototype version of the Qt graphics library was developed through minimal changes to the existing Tk/X11 based graphics library

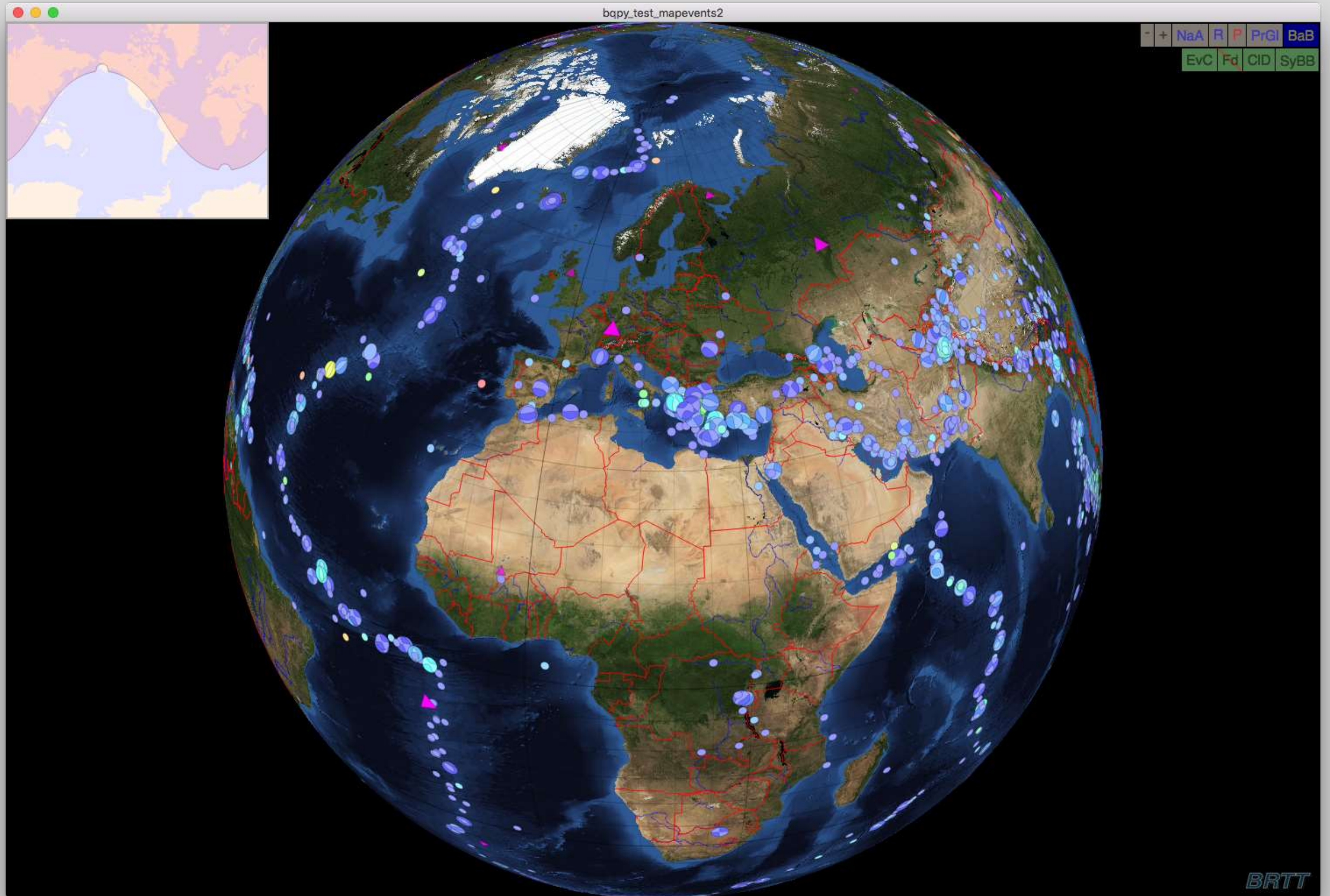
5.6 – first production version of Qt graphics library - `bqplot`

- Complete rewrite
- 5.5 prototype version did not take advantage of c++ coding capabilities – 5.6 production version takes full advantage of c++ coding capabilities resulting in code maintainability
- 5.6 `bqplot` library consists of 20 new classes that implement high level graphics function, 20,000 new lines of code and documentation
- Although derived from the old Tk/X11 buplot code, this version adds major new coding constructs that will ease development of further graphics capabilities.

5.6 – BQMapevents class



5.6 – BQMapevents class



5.6 – first production version of Qt graphics library - `bqplot`

- All `bqplot` classes are documented
- However, because of licensing restrictions, BRTT cannot provide a c++ Qt development environment as part of its distributions
- We needed to extend the new Qt graphics to a scripted environment like Python – would both ease our development tasks and provide our users development access to `bqplot`

Development of Python interpreter for `bqplot`

- `bqpy` in the 5.6 release provides a Python interpreter that will act as a bridge to the `bqplot` graphics library
- `bqpy` runs an embedded Python interpreter in one thread and a special `bqplot` server in a separate thread
- The `bqplot` server accepts commands and data through a serialized pipeline that is fed by the Python interpreter in the other thread. Note that with this design the Python interpreter and the `bqplot` server could be in separate processes

Development of Python interpreter for `bqplot`

- `qtmapevents` in the 5.6 release is now a 70 line open source Python script that runs `bqpy` (as opposed to 180 lines of c++ code in the 5.5 version)
- `displayttgrid` in the 5.6 release is now a 141 line open source Python script that runs `bqpy`
- `dbevents_pre` in the 5.6 release is now an open source Python script that runs `bqpy` and provides event graphics using the new Qt library
- BRTT will continue to convert old Tk/X11 based displays to Qt using this approach
- We encourage our users to develop graphics apps using this approach


```

from antelope.bqplot import *
from antelope.bueventview import *

def usage():
    print "usage: qtmapevents [dbname]"

nargs = len(sys.argv)
if nargs != 1 and nargs != 2:
    usage ()
    sys.exit (1)

dbname = None
if nargs == 2:
    dbname = sys.argv[1]

if dbname == None:
    map = Map ("toplevel")

    map.setdefault (
    map.configure ( \
        "latr", 0.0, \
        "lonr", 0.0, \
        "range", 380.0 )

    tbmap = map.gettaskbar ()
    tbmap.configure ( \
        "taskbar_exec", "projection=merc
" )

else:
    ev = bueventview_create ()
    bueventview_configure (ev, "dbname", dbnam
e )

mapevents = Mapevents ("toplevel")

mapevents.setdefault (

map = mapevents.getmap ()
map.configure ( \
    "latr", 0.0, \
    "lonr", 0.0, \
    "range", 380.0 )

```

```

mapevents.seteventview (ev)

tbmap = map.gettaskbar ()
tbmap.configure ( \
    "taskbar_exec", "projection=me
rc" )

tb = mapevents.gettaskbar ()
tb.configure ( \
    "taskbar_exec", "symbol=circle
" )

tb.configure ( \
    "taskbar_exec", "rt=rtp" )

mw = Root()
mw.setGeometry (2500, 1.5, 1)

mw.show ()

mw.qtmainloop ()

mw.pymainloop()

```

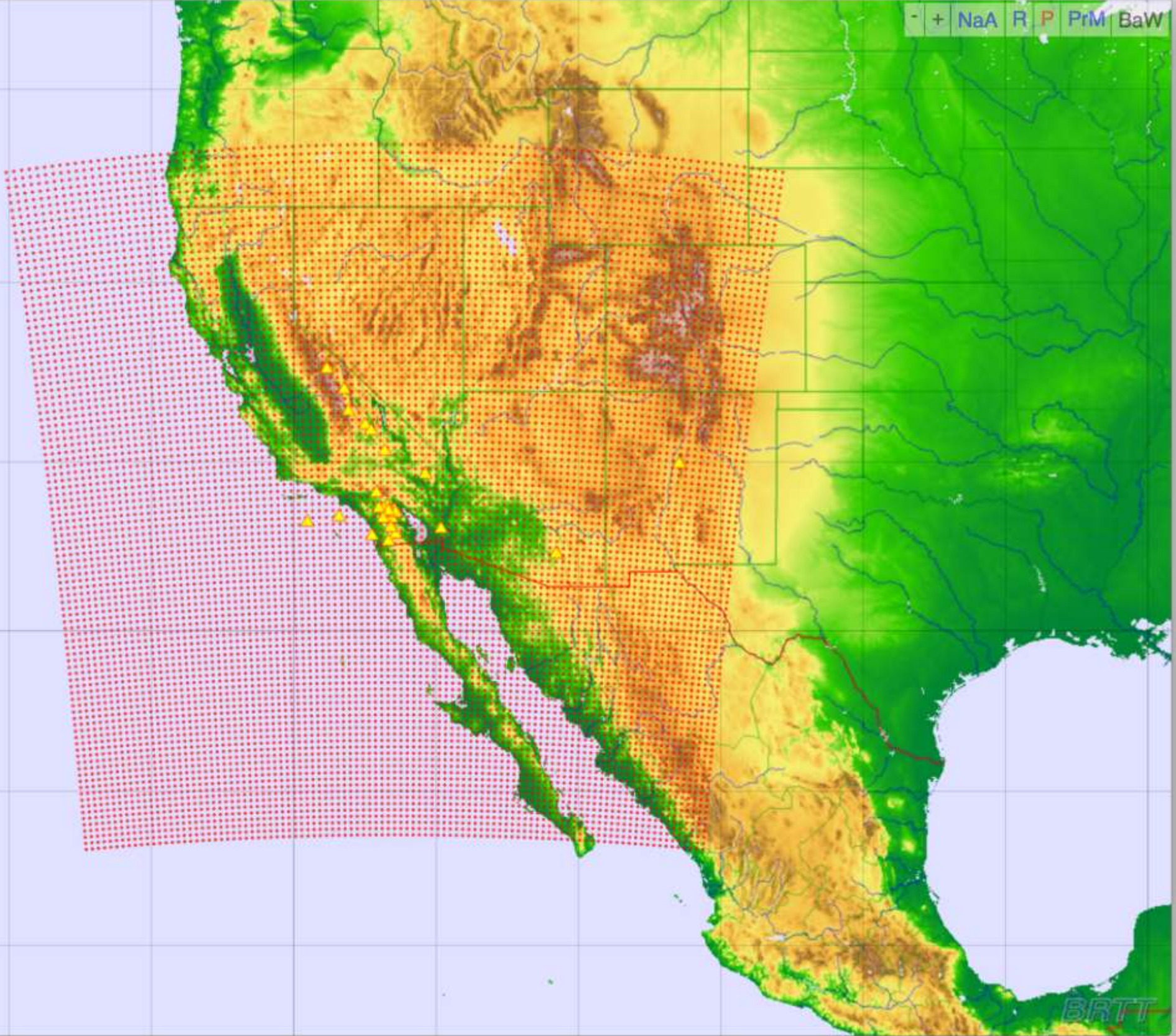
Development of Python interpreter for bqplot

- `man bqplot`
- `man pythonbqplot`

displayttgrid

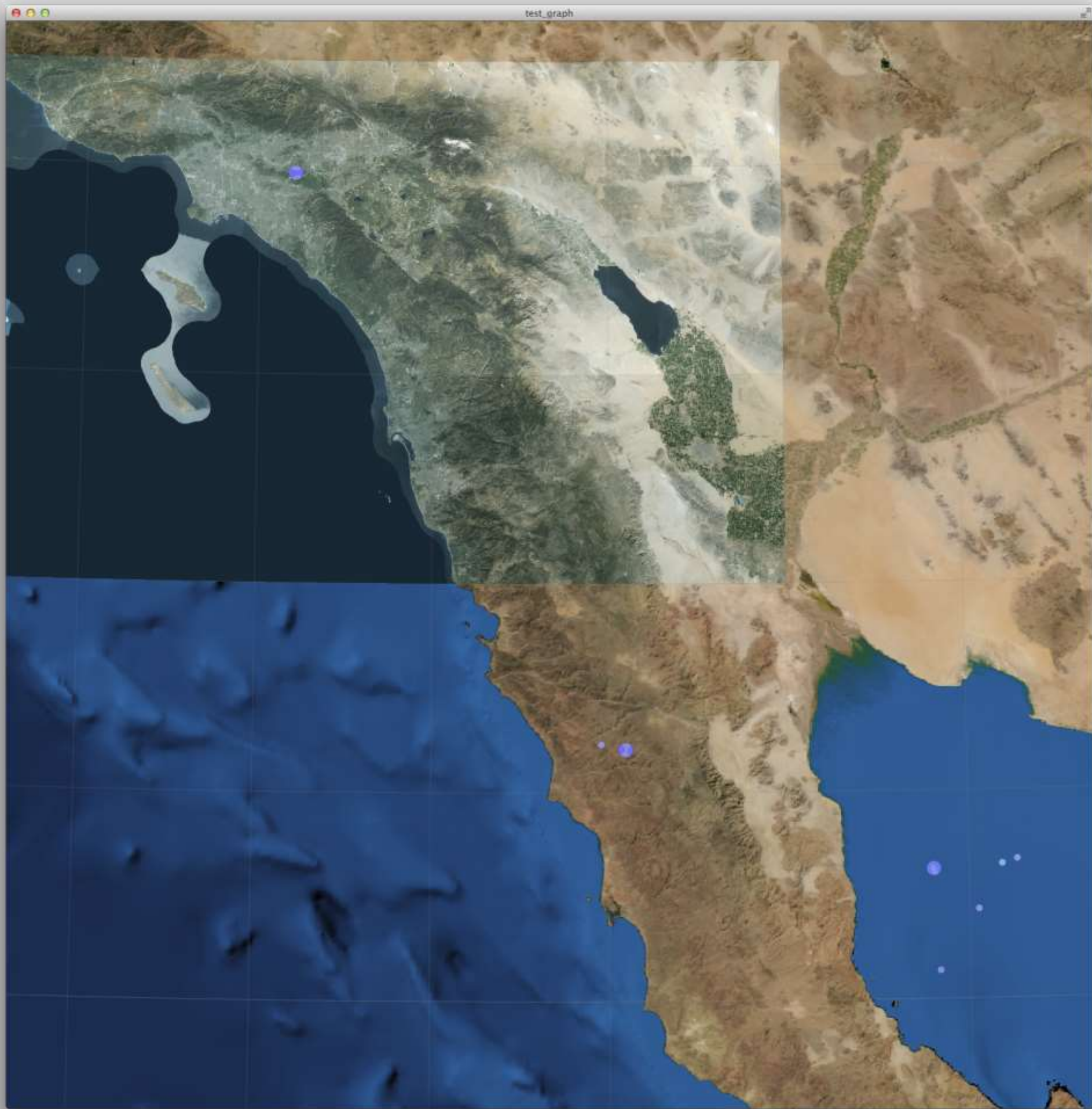


- + NaA R P PrM BaW



Further developments

- Continue `bqplot` Python extensions
- Add ability to ingest maps in other formats (gif, tiff, png, etc.)
- Add ability to ingest maps from Web Map Servers (WMS)
- Separate, standalone `bqplot` server
- Develop `bqplot` to add trace graphics and manipulation functions
- Develop `bqplot` to add simplified QUI widgets.



14:50



>> BREAKING OVERNIGHT

NORTH KOREA CLAIMS FIRST "H BOMB" TEST
AMBASSADOR KENNEDY CALLS IT A "PROVOCATION"

