HippoDraw and Python

Paul F. Kunz

Stanford Linear Accelerator Center

Brief overview of HippoDraw

Use from Python

Two Versions

• Java GUI, uses Jython
• Qt GUI, uses Python

Java version used in screen dumps that follow
What is HippoDraw

An analysis package...

Canvas

- Canvas contains the displays

Inspector

- Inspector allows you to view properties and change them.
- The only windows except for modal dialogs
Document paradigm

- Canvas can be saved as multi-page document in XML
- Documents can be opened at a later time
- Multiple opened documents are allowed
- One document serves as template for multiple data sets
- Eliminates one need for scripts or commands
Data Inspector

- controls creation of displays
- controls data binding
- GUI enquires to C++ DataRepFactory
  allows for extendability
Plot Inspector

- controls a few display options
Axis options Inspector

- controls axis range
- controls bin width and offset if binned
- note use of sliders
- log on X axis has logarithmic sized bins
Cut Inspector

- controls creation and application of cuts
- cut range changed with sliders
- can use zoom/pan feature
Function Inspector

- controls creation and application of functions
- controls fitter
- GUI makes enquires to C++ FunctionFactory
- function parameter names from enquiry to C++ function objects
Stats Inspector

- controls adding of textual representations
- the reps are “live”
Commans and scripts

HippoDraw can be used without commands or scripts
• ease of use is very good
• learning period is short
• to quote one CERN user: “HippoDraw is so easy to use, even a 50 year old CERN physicist can use it”

However, one needs a script to...
• do repetative actions, e.g. 50 histos on different channels
• massaging data
• reading special data formats
• getting and putting data from/to other packages

Solution: make HippoDraw a Python module
• HippoDraw becomes the non-instrusive slave to Python
• HippoDraw still does not have script language
from hippo import HDApp

canvas = app.canvas()

from hippo import NTuple

nt = NTuple('examples/aptuple.hiptxt')

from hippo import Display

hist = Display("Histogram", nt, 'Cost')

canvas.addDisplay(hist)
hist.setRange('x', 0., 30000.)

- hippo is name of the Python module
- HDApp, NTuple, and Display are classes implemented in C++
- app.canvas() returns current canvas.
- canvas.addDisplay() adds display in next available free space
Result of script

- same as if one had used the GUI
- all GUI controls are active
Equal access

- Inspector can send commands and inspect canvas objects
- Python session or script can do the same
- They use the same member functions of the objects
Data access

In Python session or script

• create an empty ntuple

   \[ nt = NTuple() \]

• add columns of equal length

   \[ nt.addColumn( 'label', array ) \]

• add rows of equal size

   \[ nt = addRow( array ) \]

• can also replace row or column

• if ntuple used by displays changes, the displays update themselves
Complete example

Example of reading ASCII file

```python
from hippo import *
import sys, string

infile = open( 'aptuple.hiptxt', 'r' )
lines = infile.readlines()

labels = string.split( lines[1] )
nt = NTuple( len(labels) )

nt.setTitle( string.strip (lines[0]) )
nt.setLabels( labels )

for line in lines[2:]:
    words = string.split( line )
    row = map( float, words )
    nt.addRow( row )
```

- Python is strong on parsing, competitor to Perl
Data sources for Python

Python has many modules for reading data

Here are some...

- parse a file
- RPC library
- PyFITS (Astrophysics standard)
- RootPython (Pere Mato)
- Excell spreadsheet
- easy to roll your own (PAW?)

Other data sources...

- other Python modules, e.g. PyGaudi, PyGeant4
- algorithms implemented in Python
- HippoDraw ntuples, e.g. get data, massage, add new column
There are a number of them...

- **SWIG**, the original
  - parses your C++ header files and generates code
  - limited C++ capability

- **SIP**, used and maintained by PyQt

- **boost::python**
  - you write one line per constructor or member function.
  - capable of handling templated classes like `vector<>` and `string`
  - version 2 coming soon.
boost::python

Example code you must write

```cpp
python::class_builder<HiNTuple>
   NTuple_cl(this_module, "NTuple");

NTuple_cl.def ( boost::python::constructor <> () );
NTuple_cl.def ( boost::python::constructor
   < const std::string & > () );
NTuple_cl.def ( boost::python::constructor < int > () );

NTuple_cl.def ( &HiNTuple::setTitle, "setTitle" );
NTuple_cl.def ( &HiNTuple::setLabels, "setLabels" );
NTuple_cl.def ( (int (HiNTuple::*)(const std::string &, std::vector< double > *))
   &HiNTuple::addColumn, "addColumn" );
NTuple_cl.def ( (int (HiNTuple::*)(int, const std::vector< double > &))
   &HiNTuple::replaceColumn, "replaceColumn" );
NTuple_cl.def ( &HiNTuple::addRow, "addRow" );
```

- allows for multiple constructors
- allows for function name overloading
- understands STL classes
- choice of PyGaudi, PyRoot, and HippoDraw
Grubby details

Hippodraw compiles with...

• egcs 1.1.2 thru gcc 3.2
• VC++ 6.0 sp 0 thru VC++ 7.0 (.NET)

Tested on...

• Red Hat Linux 6.1 thru 7.3
• DESY SuSE 6.x
• CERN Red Hat Linux 6.1
• Windows NT 4.0 and Windows 2000
• Mac OS X
Conclusions

HippoDraw as a stand-a-lone application offers the users great interactivity and document centric features.

HippoDraw as a module on the Python software bus effectively extends its usability to a much wider domain of applications.

HippoDraw drops into your Python environment, it is not intrusive.