onlineGUI HOWTO

Bryan Moffit
College of William and Mary

April 26^{th}, 2004
(Updated December 5^{th}, 2004)

Abstract

This paper provides a description of the basic idea and functionality of the onlineGUI written for online Replay and Monitoring of ROOT based analyses. A description of the `.cfg` (configuration) files will be followed by a brief overview of the GUI layout.

1 Introduction and Motivation

Graphical User Interfaces (GUIs) are useful for providing a user with easy access to useful information, while keeping its more complicated and repetitive structure (e.g. methods for drawing buttons) less visible. The primary purpose for writing the onlineGUI was to ease the transition of Shift Workers from the use of PAW Kumacs for online analysis to the new Hall-A analyzer which uses `C++` and ROOT libraries. As a constraint, this GUI was written to be portable to ANY analyzer that writes its histograms and output to a ROOT file.

2 Compilation and Execution

To compile and execute the onlineGUI, one must be at a ROOT CINT prompt. The may be accomplished by running the `root`, or in some situations, running the analyzer's executable without any commandline options. Compiling the onlineGUI is accomplished with the command:

```
.L online.C+
```

This will tell CINT to compile the onlineGUI (if the source code has changed), and load it into memory. Execution of the onlineGUI can then be done with:

```
online(``standard'')
```

where ``` ``standard'' ``` (quotation marks are **required**) is the basename of the configuration file (`.cfg`). If properly configured (explained in the next sections), one may also provide a runnumber:

```
online(``standard'',9917)
```

Compilation and execution may also be done within the same command with:

```
.x online.C+(``standard'',9917)
```

2.1 Generation of sample plots

The onlineGUI also has the capability of generating a set of sample plots, for a given configuration and runnumber. This is done by providing one more argument on the commandline:

```
.x online.C+(``standard'',9917,kTRUE)
```

The output is a PostScript file, it's name with the format:

```
sampleplots_9917.ps
```

Each page defined in the `.cfg` file is converted into a PostScipt page, with the title at the top of each page.

3 Configuration Commands

The configuration files must be located in the same location that the onlineGUI is executed, and must have the suffix `.cfg`. An example (`standard.cfg`) is provided in the Appendix.

`rootfile`

At a bare minimum, a configuration file should have an entry for the `rootfile` command:

`rootfile` *ROOT_file_location*

This command provides the onlineGUI with the ROOTfile in which to find its histograms to plot. *ROOT_file_location* may be relative to the directory in which the onlineGUI is executed, for example:

```
rootfile Afile.root
```

or an absolute file location can be indicated:

```
rootfile /adaql3/work1/rootfiles/e29110_phys.root
```

The relative filename location is useful for situations when only the histograms from the last analyzed file is to be plotted. To implement this, the analysis routine creates a softlink to `Afile.root`.

`protorootfile`

One means of informing the onlineGUI the means to construct the rootfile name and location is with the `protorootfile` command:

`protorootfile` *ROOTFILE_*`XXXXX`*.root*

where *ROOTFILE_* is the file prefix (which may also contain absolute directory location). When the onlineGUI is executed with a non-zero runnumber option, `XXXXX` is replaced with that runnumber (the five `X`'s are important in the configuration, but the number of digits of the runnumber is not).

`goldenrootfile`

This command notifies the onlineGUI of a rootfile that will be used to show "golden" histograms, along with the current run's histograms (this command does not work with TTree variables). Utilize this command by simply indicating the location of this "golden" rootfile:

`goldenrootfile /adaql3/work1/rootfiles/e29110_phys_9916.root`

`guicolor`

This command provides the user with the ability to change the default background color (lightblue, or lightgreen if the `watchfile` command is used). This command is particular useful if the gui is displayed on terminals that have trouble showing the default color (e.g. SUN terminals). Usage of this command is simple:

`guicolor` *color*

If the *color* is not recognized by root (using the `TGClient::GetColorByName()`), the default onlineGUI colors will be used.

`definecut`

This command is only useful for drawing variables from TTrees. If a cut is used multiple times within a configuration file, it may be desirable to use the `definecut` command:

`definecut` *cut_name* *cut_definition*

where *cut_name* is the name of the cut to be substituted by *cut_definition*. *cut_definition* can be any logically expression normally used with the second option of the `TTree::Draw()` command. There are no limits on how many `definecut`'s that can be defined in a configuration file.

`watchfile`

If a rootfile is being generated from online data (sometimes gathered from the Event Transfer (ET) system) and is periodically stored in a rootfile, the online GUI can update its canvases every two seconds if the `watchfile` command is indicated. Use of this command effectively turns the onlineGUI into an online Monitor.

`newpage` and its subcommands

Th `newpage` command tells the onlineGUI that the next set of commands are defined for the selected canvas. Used alone, the onlineGUI will determine the number of pads required to plot all of the indicated histograms. The user may opt to tell the GUI the layout of the pads in the canvas with the command:

`newpage` *COLUMNS ROWS*

where *COLUMNS* and *ROWS* are positive, non-zero integers. One may also indicate that the plots are to have a log-scale along the Y-axis with the command:

`newpage` *COLUMNS ROWS* `logy`

A title for the present canvas can be indicated with the subcommand:

`title` *Canvas Title*

Any subcommand provided (besides the `title` subcommand) after `newpage` is then interpreted as a histogram in the ROOT file to be plotted. Any TH1, TH2, TH3 objects may be plotted by just referring to their object name, for example (for plotting `Rsala10`):

```
newpage 1 1 logy
   title Just one TH1D
   Rsala10
```

If the `.cfg` contains the `goldenrootfile` command, one may opt to exclude this specific histogram from being shown along with the “golden” histogram by including `noshowgolden`, e.g.

```
newpage 1 1 logy
   title Just one TH1D
   Rsala10 noshowgolden
```

Any expression normally used in a `TTree::Draw()` can also be plotted with a similar command (for `asym_bcm1`):

```
newpage 1 1 logy
   title Tree Variable
   asym_bcm1
```

The onlineGUI will automatically determine if the object name provided is one derived from TH1, or is a Tree Variable. One limitation of this method, is that one may not use the same tree variable name in separate trees. If this is the case, the onlineGUI will just plot the variable in the first tree in which it finds that variable.

A second option may be used as a cut to that histogram, and may contain a `definecut` name that has been defined. For example:

```
definecut AsymmetryCut abs(asym_bcm1)<5000
newpage 1 3
   title Tree Variables
   asym_bcm1              ok_cut
   asym_bcm1              AsymmetryCut
   asym_bcm1:asym_bcm2    ok_cut&&(AsymmetryCut)
```

More advanced commands for drawing TTree variables include:

`-title` "*Title*"	Replaces the default TTree::Draw() title with that specified within quotes. Note that the quatation marks are vital for this option to work correctly.
`-tree` *TreeName*	Informs the onlineGUI which tree to find the variables it needs to draw. This option is effective when drawing complicated functions of TTree variables.. E.g. `abs(asym_bcm1):m_ev_num -tree P`
`-type` *DrawType*	Informs the onlineGUI how to draw the specifed variables. Types include: `scat, box, prof` If this option is not specified for 2D plots, the `box` type will be used.

Finally, one can utilize the `macro` subcommand to draw more complicated/complex histograms within the canvas. This method is quite useful because one still has the ability to pass command optics to the macro. The macro must be located in the same directory that the onlineGUI is executed. An example is shown below (with the name of the histogram passed to the macro routine `vdc_plot.C(TString)`):

```
newpage 2 2
   title Macro Draw
   macro            vdc_draw.C(``Lu1eff'')
   macro            vdc_draw.C(``Lu2eff'')
   macro            vdc_draw.C(``Lv1eff'')
   macro            vdc_draw.C(``Lv2eff'')
```

```
// Contents of vdc_plot.C
void vdc_plot(TString vHist) {

  TH1D *vdc = (TH1D*)gDirectory->Get(vHist);

  gStyle->SetOptStat(0);
  vdc->SetMinimum(0.8);
```

```
  vdc->Draw();

}
```

4 Helper Routines

The onlineGUI utilizes two Helper Routines in order to obtain information that may be specific to the type of ROOT analysis performed. These routines **must** be located in the same directory that the onlineGUI is executed. A compilation error will occur if these files do not exist.

GetRootFileName.C

This routine supplies a ROOT filename, if the onlineGUI is run with a non-zero runnumber option. NOTE: The `protorootfile`, if present in the configuration file, will override this routine. In the case that this routine uses a class that is not included in the standard ROOT libraries, the shared library that contains this class must be loaded before the onlineGUI is executed. An example of this routine is shown below. In this case, `TaFileName` is a `C++` Class specific to the Parity Analysis (PAN).

```
#include "src/TaFileName.hh"

TString GetRootFileName(UInt_t runnumber)
{
  TaFileName::Setup (runnumber, "standard");

  TString filename = (TaFileName ("root")).Tstring();
  return filename;

}
```

GetRunNumber.C

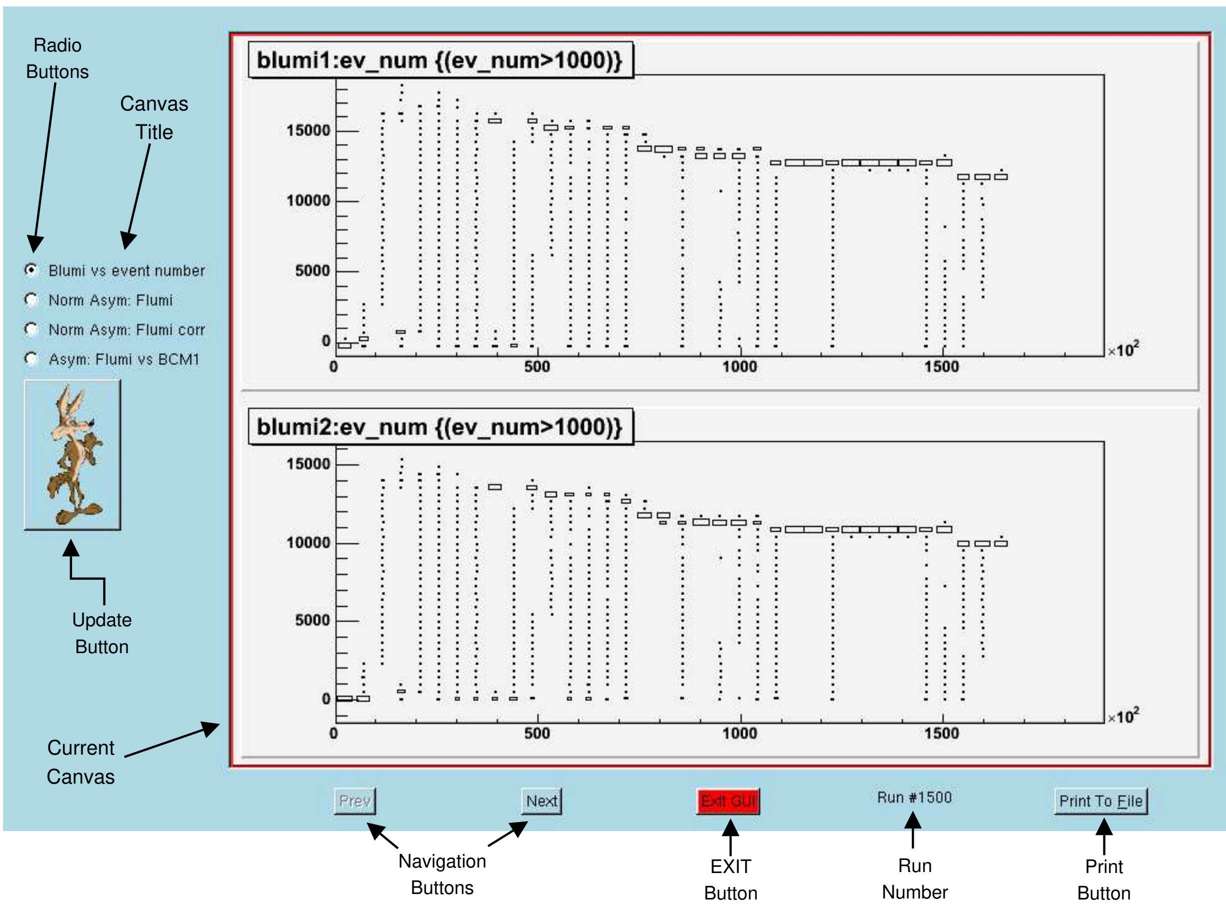
This routine supplies a runnumber for the current ROOT file. The same situation regarding the loading the shared libraries in the previous subroutine, applies here. In the following example, `THaRun` is a `C++` class specific to the Hall-A Analyzer.

```
#include "/opt/analyzer-1.1/src/src/THaRun.h"

UInt_t GetRunNumber()
{

  THaRun* runinfo = (THaRun*)gROOT->FindObject("Run_Data;1");
  if(runinfo==NULL) return 0;
```

```
  return runinfo->GetNumber();

}
```

If there is not a method contained with a user's analyzer to retrieve the run number of the rootfile, one should write a routine that simply returns `0`.

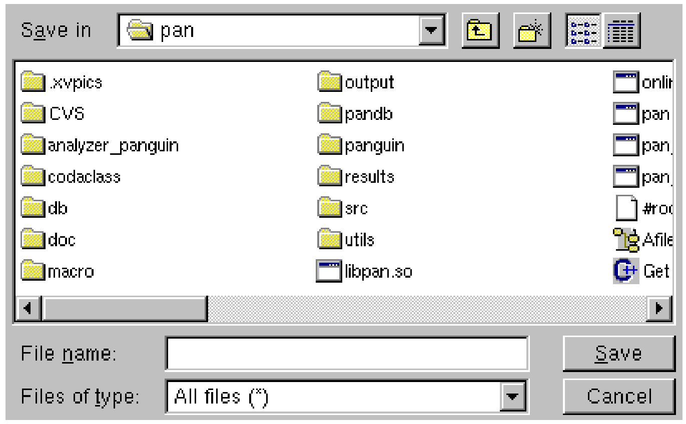
5 GUI Layout



The onlineGUI layout for a sample configuration is shown above. On the left-hand-side, radio buttons indicate which canvas is currently shown along with their canvas title. Each radio button is "click-able" to allow for non-consecutive navigation.

An update button containing a picture Wile E. Coyote (Super Genius) is not just "eye-candy". It's function is to redraw the current canvas (useful if the `macro` subcommand is changed during onlineGUI execution). OR if the `watchfile` is defined in the configuration file, this button will clear all canvases so that they can be filled starting with the most recent events.

On the bottom of the GUI, the `Next` and `Prev` navigation buttons allow the user to go back and forth through the canvases one at a time. The `Exit GUI` button closes the GUI. And the `Print to File` button brings up a Save File dialog box:



allowing one to save the current canvas as a `GIF`, `PostScript`, `Encapsulated PostScript`, `ROOT macro`, or a histogram object contained within a `ROOTFILE`. This button is particularly useful for generating example online histograms for Shift Workers to cross-check with current runs.

6 Known Limitations/Bugs

The current version of the onlineGUI was used during the Jefferson Laboratory experiments: E00-110 and E03-106 (DVCS on the proton and neutron). The only observed bug has been when the onlineGUI is executed in the same CINT window after a Hall-A analyzer process has executed. Exiting CINT with the `.q` command results (sometimes) in an interpreter error. This behavior does not manifest itself if the same is done with the Parity Analyzer.

The only known limitation at this time, is that scatterplots performed when issuing a Tree::Draw(), cause a memory leak. This bug was fixed in ROOT versions $\geq$ 4.00/04.

If you have a comment or suggestion, please contact Bryan Moffit(`moffit@jlab.org`).

7 Appendix: A sample configuration file

```
# Configuration file for the online GUI
# This file is for checking luminosity monitor data
#  All lines containing "#" are treated as comments (ignored)
#  Last line MUST be blank.
#  Multiple spaces are treated as one space.
# Only Tree variables are in the PAN rootfile.
#   (No histograms)

# The default rootfile to look for.
rootfile      pan.root

# If the onlineGUI is run with the runnumber option,
#  replace the X's below with the runnumber.
protorootfile /adaql1/work1/parity/parity04_XXXXX_standard.root

# Some defined cuts
# only view events greater than 1000 in the R tree.
definecut evcut      (ev_num>1000)
# only view the last 900 pairs in the P tree.
definecut asymevcut  (Entries$-Entry$)<900

# Uncomment this line to monitor events as they are written to the rootfile
#watchfile

# Begin the canvas definitions
newpage 1 2
        title Blumi vs event number
        blumi1:ev_num evcut -title ``BLumi1''
        blumi2:ev_num evcut -title ``BLumi2''

newpage 1 2
        title Norm Asym: Flumi
        asym_n_flumi1 (asymevcut)&&ok_cut -title ``FLumi1''
        asym_n_flumi2 (asymevcut)&&ok_cut -title ``FLumi2''

newpage 1 1
        title Norm Asym: Flumi corr
        asym_n_flumi1:asym_n_flumi2 (asymevcut)&&ok_cut

newpage 1 2
        title Asym: Flumi vs BCM1
        asym_flumi1:asym_bcm1  (asymevcut)&&ok_cut
        asym_flumi2:asym_bcm1  (asymevcut)&&ok_cut
```