Cross-platform approach to create the interactive applications based on ROOT and Qt GUI libraries

R.Brun[a], V.Fine[b], J.Lauret[b] and F.Rademakers[a]

[a]CERN, European Organization for Nuclear Research
CH-1211, Geneva. Switzerland

[b]Brookhaven National Laboratory,
Upton, NY 11973-5000, USA

This paper presents the approach and an implementation of the Qt-based low level graphical plug-in of the ROOT package. In spite of other existing implementations, this work uniquely allows to create applications which can seamlessly use the ROOT GUI classes as well as the Qt Widgets.

The current code can be built from a single ROOT-base package and can be assembled for both Unix and Windows OS. No alteration of either user code or core ROOT code is required.

An example application using this approach is present as well.

1. Virtual Graphical and Operating System Layers of ROOT

It is not unusual for a major HENP application to have a life cycle spanning over decades [1,2]. Practically, such life time exceeds the life time of any known system components like an operating system, a file system, a GUI technology or a specific hardware. To ensure longevity, a sophisticated HENP application must integrate some layers which separate the application from the concrete system environment evolution. This is possible by introducing a virtual operating system model described by an abstract interface the application then, relies on; creating, improving, and adjusting the implementation(s) of that model as soon as operating environment evolves.

Such approach is used by many successful software packages including ROOT[3] and Qt[4]. Even though the approach does allow prolonging the life cycle of the software package, this does come at a price. For the entire application's life time, one is still required to learn and maintain the specific pieces of the code for many different platforms, and the different versions of one and the same platform. The magnitude of this task can sometimes lead the developers to narrow the number of supported platforms for a given package. On the other hand many features one has to deal with are not application specific. This implies that a good layer can be "borrowed" from some third party dedicated package.

For example, at present, the ROOT package does define a virtual graphical layer by introducing a base class `TVirtualX` which has several implementation, one of which is the Qt interface class[5]. `TVirtualX` defines the features ROOT, being An Object Oriented Data Analysis Framework, needs to implement in order to provide its functionality for each supported platform (see Figure 1) and isolate the package core from the concrete visualization system dependencies.

2. Qt layer

2.1. Qt library and GUI applications

The present model has been serving ROOT and its users very well for almost 8 years. In fact, most ROOT GUI problems are not ROOT specifics and have been successfully solved by fixing the implementation in other layered packages.

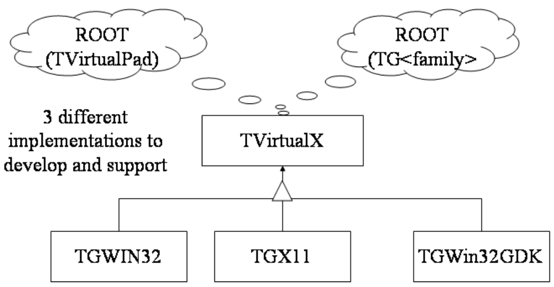

Figure 1. ROOT low level GUI interface and its concrete implementations.

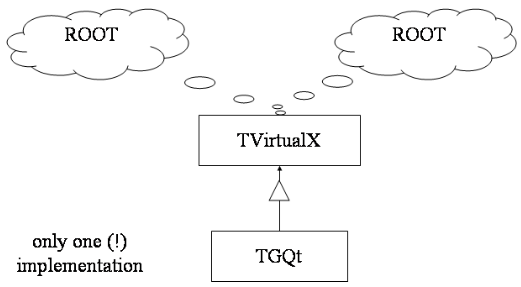

Figure 2. ROOT Qt-based GUI interface

A GUI package such as Qt has been designed specifically to address GUI portability and functionality problems for other applications [6]. The Qt package from *TrollTech AS* (`http://www.trolltech.com`) is therefore especially attractive not only because of its superior quality and high level technical support, but also because it comes with the source code and tools to build it in place (including a commercial version for Windows). The rich set of Qt documentation can be found on the Web and available from the leading publishers.

Qt is a multi-platform C++ application framework one can use to write single-source applications that run natively on Windows, Unix (Linux, ...), Mac OS X and even embedded Linux. It has been used to build thousands of successful commercial applications worldwide, and it is the basis of the open source KDE desktop environment.

Hence, the Qt GUI library from TrollTech appears to be a good choice to make the Qt version of the ROOT `TVirtulaX` interface and deliver a package combining the best features of both and simplify the ROOT-based applications development and maintenance (see Figure 2).

2.2. ROOT GUI class library and Qt layer

The first version of Qt layer was present on Acat2002 at Moscow[5] but had a significant drawback. Its goal was to solve the problem of combining the ROOT Canvas and Qt widgets within one single application. However, it was not compatible with the ROOT GUI class library and the conventional ROOT development.

Here, we present the next generation of the layer (See: Figure 3). In spite of the first version, and to achieve the full compatibility with the traditional ROOT GUI, not only the graphical output primitives but the internal ROOT event structure needed to be mapped to the Qt input event objects model and vice-versa.

The normal Qt input event object model assumes the input `QEvent` class objects are to be dispatched by Qt event loop to the destination Qt widgets. This schema contradicts the ROOT GUI design which requires for the input events defined by the `Event_t` C-structure, to first be delivered to the ROOT `TGClient` class and later dispatched for the ROOT GUI class objects.

In order to satisfy this requirement and preserve the system performance, a Qt event filter and specifics QWidget `TQtClientWidget` and `TQtWidget` classes were designed. These classes were made to catch all incoming `QEvent` objects and map them to the `Event_t` structure instances. The later are to be re-dispatched to the `TGClient` class via `TVirtuaX::NextEvent` method (see Figure 3).

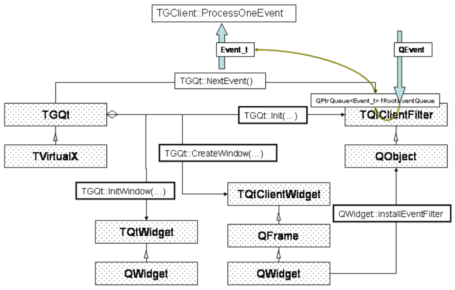

Figure 3. Transformation the `QEvent` Q-object into the ROOT `Event_t` structure.

2.3. Qt initialization

ROOT and Qt package have very similar design. To run ROOT one is requires:

1. Instantiate a `TApplication` class object
2. Enter the ROOT event loop with the method `TApplication::Run`

To run Qt one needs:

1. Instantiate a `QApplication` class object
2. Enter the Qt event loop with the method `QApplication::exec`

The Qt plug-in for ROOT initializes Qt. As a result, as soon as the user creates `TApplication` object he/she can start using Qt classes as well. To understand this point, one can compare the pure Qt code (see Figure 4) with the ROOT-based code implementing the Qt-layer (see Figure 5) to see that it is enough to create the ROOT application (instantiate `TApplication` class) to use Qt classes as well.

```
#include <qapplication.h>
#include <qpushbutton.h>
int main( int argc, char **argv ) {
    QApplication a( argc, argv );// to be replaced
    QPushButton hello( "Hello world!", 0 );
    hello.resize( 100, 30 );
    a.setMainWidget( &hello );   // to be removed
    hello.show();
    return a.exec();             // to be replaced
}
```

Figure 4. Qt application.

```
#include <TRint.h>
#include <qpushbutton.h>
int main( int argc, char **argv )
{  // Create an interactive ROOT application
    TRint *theApp = new TRint("Rint", &argc, argv);
    QPushButton hello( "Hello world!", 0 );
    hello.resize( 100, 30 );
    hello.show();
    theApp->Run(); // and enter the event loop...
}
```

Figure 5. ROOT application with Qt layer.

3. CONCLUSIONS

The approach presented in this paper allows the ROOT developers as well as ROOT users to work with code that has no specific (X11/WIN32 for example) graphics subsystem dependencies while, at the same time, opens unrestricted access to a rich set of ready-to-use commercial and free GUI Qt-based widgets [7]. The Qt layer is provided as a standard ROOT plug-in share library allowing user to turn the layer on/off at run time with no change of recompilation of the user's code. At the moment we are testing the layer to provide the full ROOT GUI classes' implementation. For further information about progress as well as the up-to-date source codes, test suits and examples can be found on BNL Web site: `http://root.bnl.gov/`. The layer is planned to be completed and fully compatible with the

ROOT abstract interface and the Qt layer will be available in future with the standard ROOT distribution. The first fully functional implementation is planned to be available by the first quarter of 2004, the required step would be for the user to install the Qt layer packages and modify the ROOT resource files.

REFERENCES

1. R.Brun et all. PAW. PAW - Physics Analysis Workstation. User's Guide. CERN Program Library Long Writeup Q121, CERN, Geneva, 1999.
2. GEANT - Detector Description and Simulation Tool. CERN Program Library Long Writeup W5013, CERN, Geneva, 1993.
3. R.Brun et all. The ROOT Framework, AIHEPN-96, Lausanne, Switzerland, August 1996
4. M.K.Dalheimer, Programming with Qt. Second Edition, O'Reilly & Associates, 2002.
5. V.Fine. Qt-Based Implementation of Low Level ROOT Graphical Layer. Proceeding of "ROOT2002 IV International Workshop, CERN, Geneva, 2002.
6. J.Adamczewski et all. Go4 v2.0 Status & Overview. Proceeding of "ROOT2002 IV International Workshop, CERN, Geneva, 2002.
7. V.Fine, OO model of the STAR offline production "Event Display" and its implementation based on Qt-ROOT, Proceedings of CHEP03. Computing in High Energy and Nuclear Physics 2003 Conference Proceedings, March 24-28, 2003, La Jolla, California