

# Qt and ROOT

by  
Valeri Fine  
STAR

Brookhaven National Laboratory



9/28/2005  
fine@bnl.gov

ROOT 2005 Workshop  
CERN



# Overview

It presents the "user" components and tools of the QtRoot layer to facilitate:

- creation the Qt-based ROOT application and shared library;
- the ROOT-based Qt-application;
- using QT GUI "designer" and Qt "project file";
- generating the class HTML documentation of the Qt/Root-based user classes;
- ROOT-based and stand-alone QtRoot layer installation;
- embedding 2D and 3D Qt widgets rendering the ROOT graphical objects.

The list of the features, achievements and problems along with the "ToDo" list are present also.



# TGQt – Qt-based implementation of TVirtualX

- Qt-ROOT implementation of **TVirtualX** (Qt-layer) is to provide a convenient way of creating the complex end-user applications those require both Qt GUI and ROOT features.
- The primary goal is to allow “embedding” the ROOT classes like **TCanvas** and **TPad** into the arbitrary Qt widgets and using it seamlessly with other Qt-based components and Qt-based third party libraries.
- **TGQt** ROOT class, a Qt-based implementation of **TVirtualX** interface is an optional ROOT component. The implementation was developed and is supported by STAR collaboration at Brookhaven National Laboratory.
- The history of the project and the QtRoot technical implementation details were present at ACAT 2002 (Moscow) and ACAT 2003 (Tsukuba).



9/28/2005

[fine@bnl.gov](mailto:fine@bnl.gov)

ROOT 2005 Workshop

CERN



3

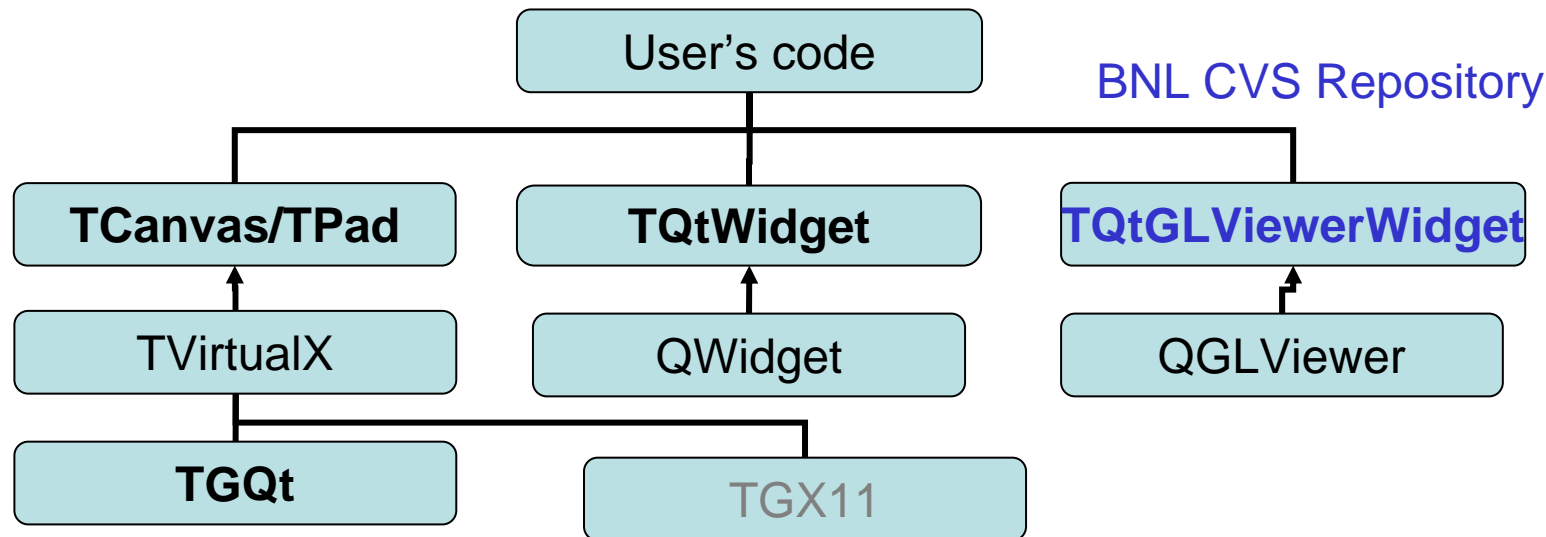
# “Spitted” repository

At the moment the project is “splitted” by two separate CVS Repository: “Qt layer” and “Qt Extension”

- The full version that “Qt-layer” + “Qt Extension” is available from BNL CVS Repository
  - CVSROOT : `pserver:cvsuser@cvs.bnl.gov:/date01/cvs`
  - Web (STAR): `http://root.bnl.gov`
  - Mail list: `qt-root-1@lists.bnl.gov`
- The “Qt-layer” is a part of the official ROOT repository. See: `http://root.cern.ch`



# Two QtRoot end-user “use cases”



Even though the TGQt class is a “soul” of the Qt layer it just implements at the TVirtualX interface and is normally hidden from the end-user code

**1. Qt-based ROOT applications** use “TCanvas”

**2. ROOT-based Qt application** use TQtWidget  
TQtGLViewerWidget



9/28/2005  
fine@bnl.gov

ROOT 2005 Workshop  
CERN



# *Use case: “Qt-based ROOT applications”*

“**ROOT application**” is the application that either instantiates the ROOT **TApplication** / **TRint** class and enters the ROOT event loop or is the shared library that can be loaded into the already running ROOT application via **TSystem::Load** method or via ROOT plug-in mechanism. You must neither initialize Qt **QApplication** nor enter the Qt event loop. Qt-layer takes care about both of these steps. What you need is just to instantiate the Qt object of your choice and keeps playing ROOT rules.



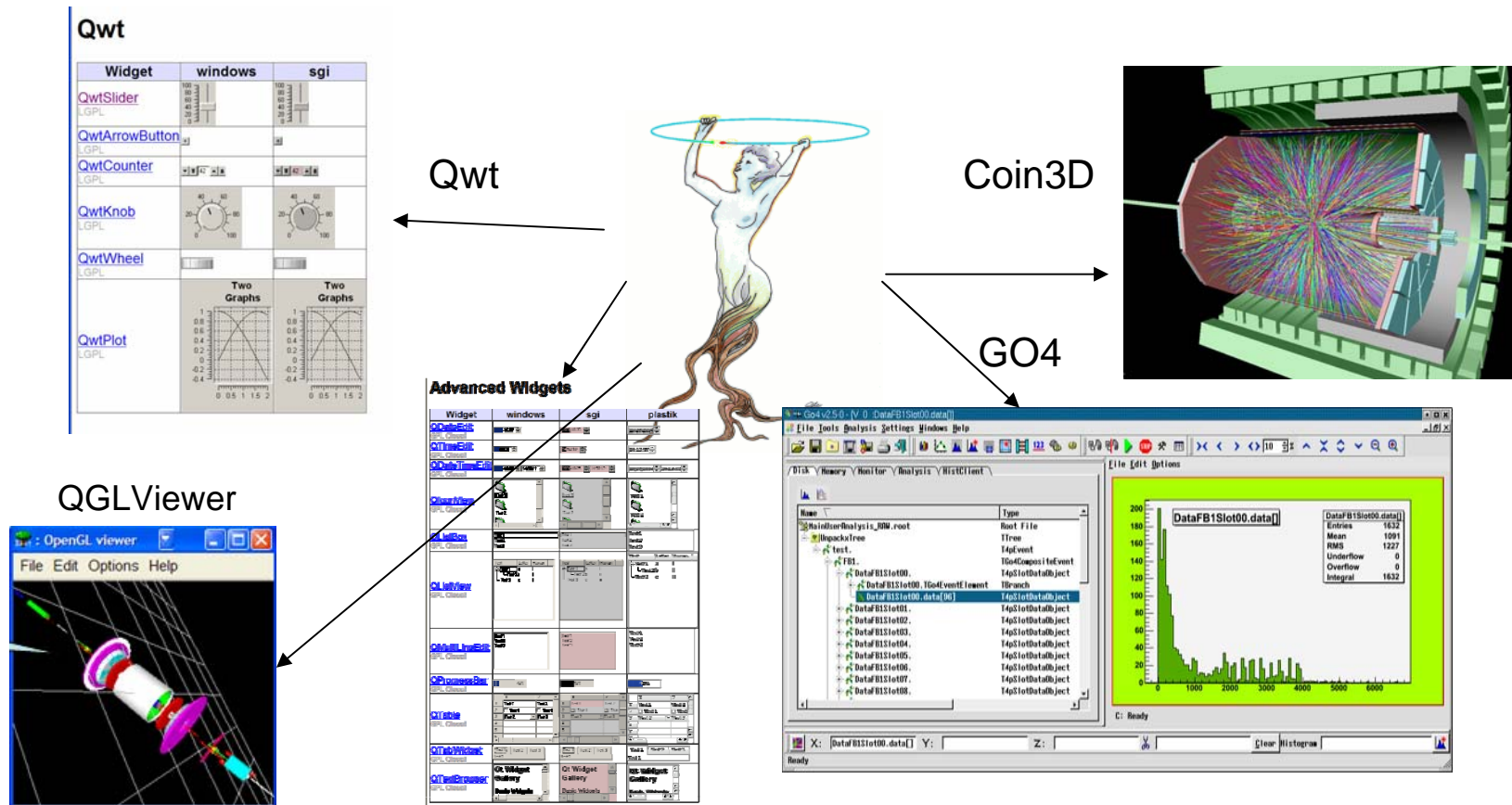
# *Use case: “ROOT-based Qt applications”*

“Qt application” is the application that either instantiates the Qt QApplication and enters the Qt event loop or is the shared library that can be loaded into the already running Qt application via Qt plug-in mechanism. You must neither initialize **ROOT TApplication / TRint** nor enter the ROOT event loop. Qt-layer takes care about both of these steps. What you need is just to instantiate the embedded and regular ROOT objects of your choice and keep playing Qt rules.



# Benefits

One can include Qt GUI components and packages into both cases



9/28/2005  
fine@bnl.gov

ROOT 2005 Workshop  
CERN





# 2D ROOT graphics: TQtWidget – QWidget for embedded TCanvas

- Can be used with Qt “designer”
- Provided with a bunch of the proxy-methods to access the embedded TCanvas object
- Provided with the convenient Qt signals to notify the client code about the object selection, mouse events and interactions
- Can be subclassed to meet some special needs.

```
#include "TQtWidget.h"

TQtWidget

class description - source file - inheritance tree (.pdf)

class TQtWidget : public QWidget

Inheritance Chart:
QWidget <- TQtWidget

private:
    TQtWidget(const TQtWidget&)
    void operator=(const TQtWidget&) const
    void operator=(const TQtWidget&)

protected:
    void AdjustBufferSize()
    TCanvas* Canvas()
    virtual void customEvent(QCustomEvent*)
```



# Simple histogram browser (60 lines of C++ code)

ROOT (Qt inside)

Click the histograms from the ListView to draw them onto the 1d or 2d canvases on the right

<http://root.bnl.gov>

Name	Type
hprof	TH1
hpx	TH1
hpxpy	TH2

hpx

hpxpy

```
gROOT->Macro("qcanvas.CC(gPad)");
```

GO4 "design"



9/28/2005  
fine@bnl.gov

ROOT 2005 Workshop  
CERN



10

# 3D ROOT graphics – QtGLViewerWidget

- Can be used with Qt “designer”
- Very simple the end-user interface to pass the “GL list” to be rendered
- Provided with the convenient Qt signals to notify about the GL object selection
- Can be subclassed to meet some custom needs also

The image displays two screenshots from Microsoft Internet Explorer. The left screenshot shows the 'Index of QT classes' page for TQtGLViewerWidget, displaying its inheritance chart and source code. The right screenshot shows the libQGLViewer website, featuring a logo, version information (2.1), and a presentation section.

**Index of QT classes - Microsoft Internet Explorer**

Address: <http://root.bnl.gov/QtRoot/html/doc/TQtGLViewerWidget.html>

**T Qt-layer classes**

```
#include "TQtGLViewerWidget.h"
```

### TQtGLViewerWidget

[class description](#) - [source file](#) - [inheritance tree \(.pdf\)](#)

**TQtGLViewerWidget : public QGLViewer**

Inheritance Chart:

```
QGLViewer <- TQtGLViewerWidget
```

Private:

```
TQtGLViewerWidget(const TQtGLViewerWidget&);  
void operator=(const TQtGLViewerWidget&);
```

Protected:

```
virtual QDomElement domElement(const QString& name, QDomDocument* doc, const QDomElement* parentElement) const;  
virtual void draw();  
virtual void drawFooter();  
virtual void drawGLLights();  
virtual void drawSelectedObject();  
virtual void drawWithNames();  
virtual void init();  
virtual void postDraw();  
virtual void postSelection(const QPoint& point);
```

public:

**libQGLViewer - Microsoft Internet Explorer**

Address: <http://artis.imag.fr/Members/Gilles.Debunne/QGLViewer/index.html>

Home Documentation Download Examples Developer



## libQGLViewer

OpenGL Qt

Version 2.1 is now available !  
Supports Qt 4. Dual licensing.

### Presentation

libQGLViewer is a free C++ library based on Qt that enables the quick creation of OpenGL 3D viewers. It features a powerful camera trackball and simple applications simply require an implementation of the `draw()` function. It is a tool of choice for OpenGL beginners and assignments. It provides screenshot saving, mouse manipulated frames, stereo display, interpolated keyFrames, object selection, and much more. It is fully customizable and easy to extend to create complex applications, with an optional Qt GUI.

libQGLViewer is *not* a 3D viewer that can be used directly to view 3D scenes in various formats. It is more likely to be the starting point for the coding of such a viewer. Since it is based on the Qt toolkit, it compiles on any architecture (Unix-Linux, Mac, Windows, ...). Full



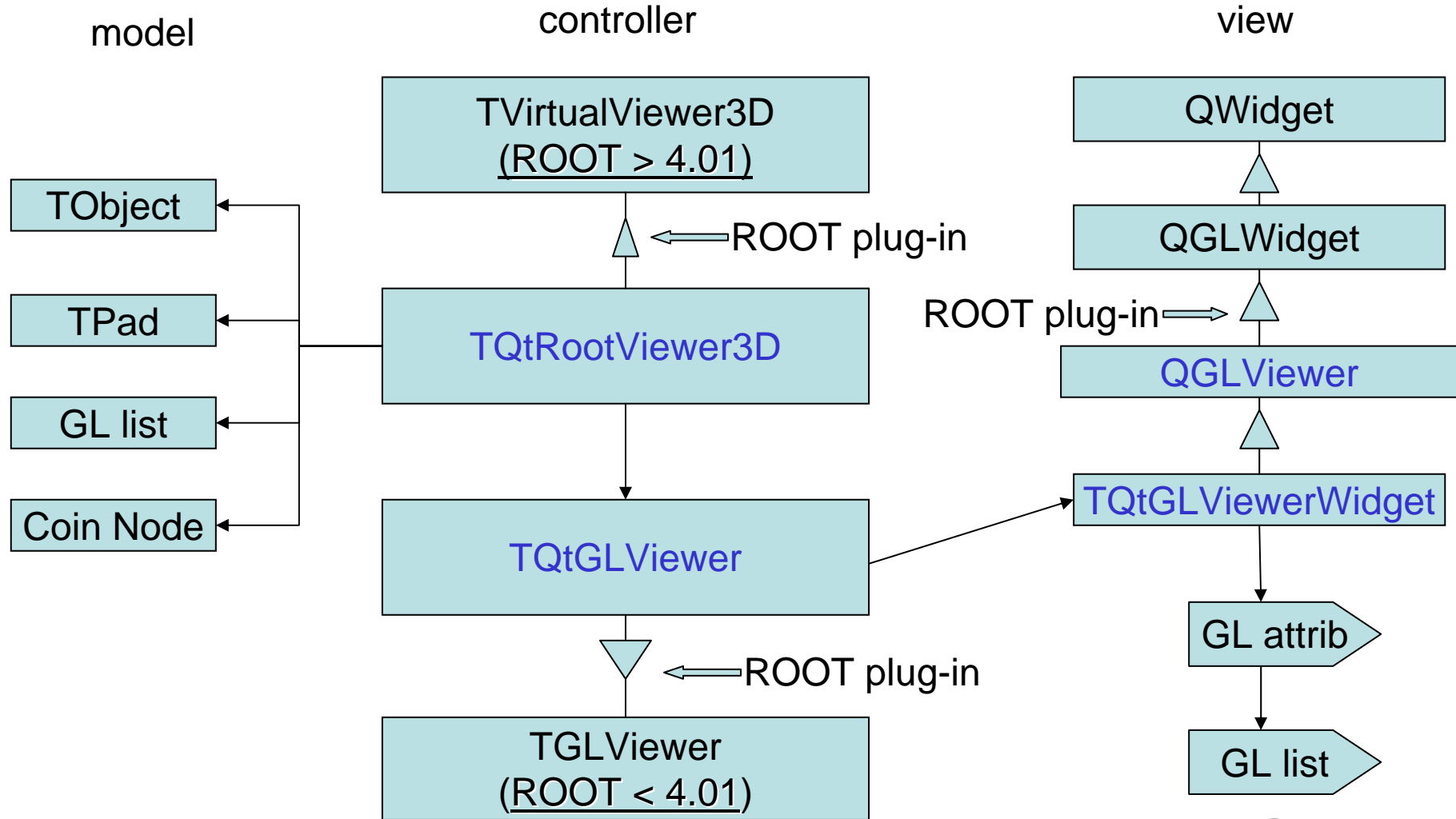
9/28/2005  
fine@bnl.gov

ROOT 2005 Workshop  
CERN








**BROOKHAVEN**  
NATIONAL LABORATORY

11

# Qt implementation of ROOT 3D viewer interface classical: Model-View-Controller



# Qt 3D features

- No “soft” matrix / coordinate transformations 
  - *it is as fast as your GL video hardware*
- One to one “model-matrix”  “view-matrix”,   
“model-points”  “view-points”, “hierarchical model”   
“hierarchical view”
  - *Trivial navigation “model->view” and “view->mode”*
  - *Change the model does entail change neither in the controller not in view source code*
  - *“Controller” and “view” implementations are trivial, therefore it is robust and stable* 
- Unlimited number of the simultaneously rendered views / sub-views / widgets / objects 
- No user’s model (TObject for example) pointer needed
  - *The destruction of the model does not entail the viewer crash and versa verse.*
- Functionality satisfies the most STAR collaboration requirements



# Click mouse action of QGLViewer

ClickAction	Description	Default binding
ALIGN_CAMERA	Align the camera axis with the world coordinate system axis.	Double click left button
ALIGN_FRAME	Align the <code>manipulatedFrame()</code> axis with the camera.	Control + double click left button
CENTER_FRAME	Translates the <code>manipulatedFrame()</code> to the center of the screen.	Control + double click right button
CENTER_SCENE	Translates the camera so that the <code>sceneCenter</code> is in the center of the screen.	Double click right button
NO_CLICK_ACTION	No action, only used as a specific return value in <code>QGLViewer::clickAction()</code> .	
SELECT	Calls the <code>QGLViewer::select()</code> function.	Shift + Left button
RAP_FROM_PIXEL	Set the camera <code>revolveAroundPoint()</code> to the point under pixel (if any).	Double click left button with right button pressed
RAP_IS_CENTER	Makes the <code>sceneCenter</code> the new camera <code>revolveAroundPoint()</code> .	Double click right button with left button pressed
SHOW_ENTIRE_SCENE	Translates the camera so that the entire scene is visible.	Double click middle button
ZOOM_ON_PIXEL	Makes the camera zoom on the pixel under the mouse (if any).	Double click left button with middle button pressed
ZOOM_TO_FIT	Makes the camera zoom to see the entire scene.	Double click right button with middle button pressed

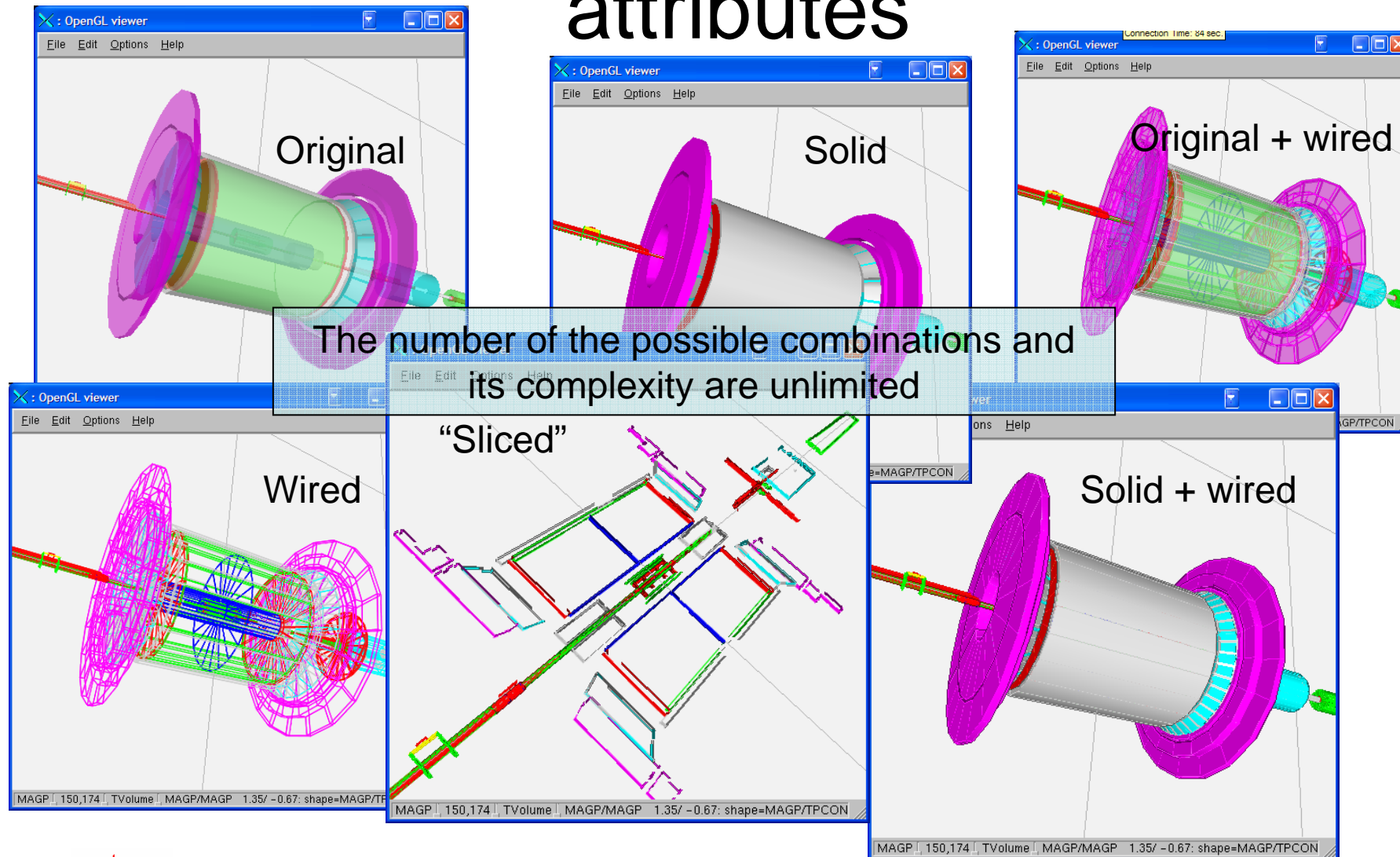


# Mouse action of QGLViewer

MouseAction	Handler	Description	Default binding
NO_MOUSE_ACTION		No action, only used as a specific return value in <code>QGLViewer::mouseAction()</code> .	
ROTATE	CAMERA	Rotates the camera around its <code>revolveAroundPoint()</code> .	Left button
	FRAME	Rotates the <code>manipulatedFrame()</code> around its origin.	Control + Left button
ZOOM	CAMERA	Makes the camera zoom in/out. Speed depends on distance to the scene center.	Middle button
	FRAME	Makes the <code>manipulatedFrame()</code> move closer or further from the camera.	Control + Middle button
TRANSLATE	CAMERA	Translates in the camera XY plane.	Right button
	FRAME		Control + Right button
MOVE_FORWARD	CAMERA	Makes the camera go forward at <code>flySpeed()</code> and view direction can be changed.	
LOOK_AROUND	CAMERA	Change the viewing direction. The camera position is not modified.	
MOVE_BACKWARD	CAMERA	Same as <code>MOVE_FORWARD</code> but backward.	
SCREEN_ROTATE	CAMERA	Rotates around an axis orthogonal to the screen.	Left and middle buttons
	FRAME		Control + Left and middle buttons
ROLL	CAMERA	Rolls camera according to horizontal mouse displacement.	Left and middle buttons (fly mode only)
SCREEN_TRANSLATE	CAMERA	Translates purely horizontally or vertically on screen.	Middle and right buttons
	FRAME		Control + Middle and right buttons
ZOOM_ON_REGION	CAMERA	Draws a rectangular region on screen and zoom on it.	Shift + Middle button

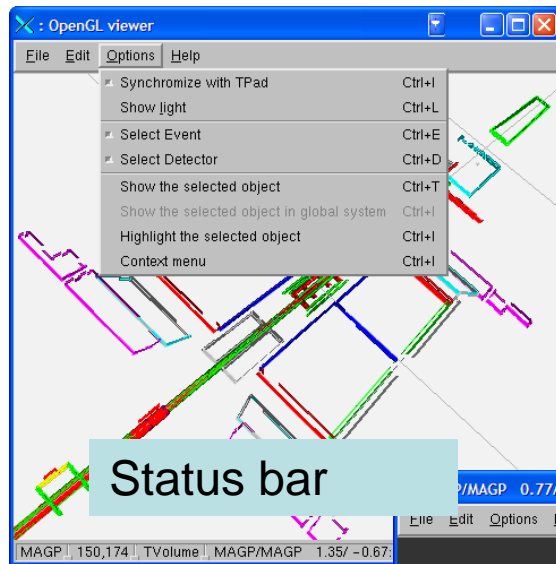


# Combination of the different GL attributes

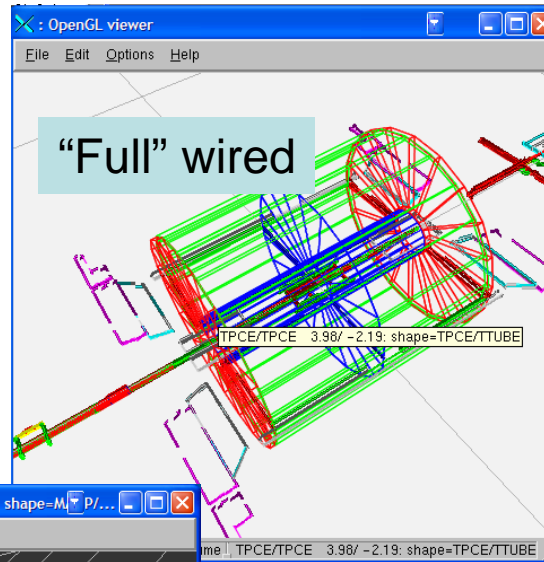




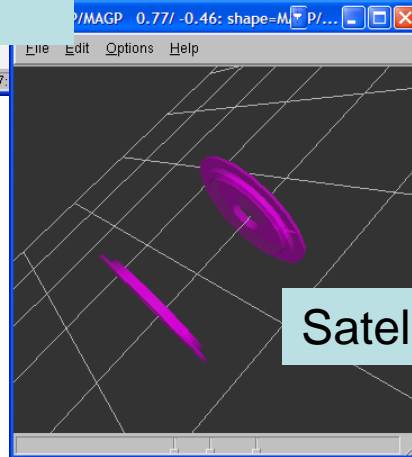
# 3D selection and highlighting



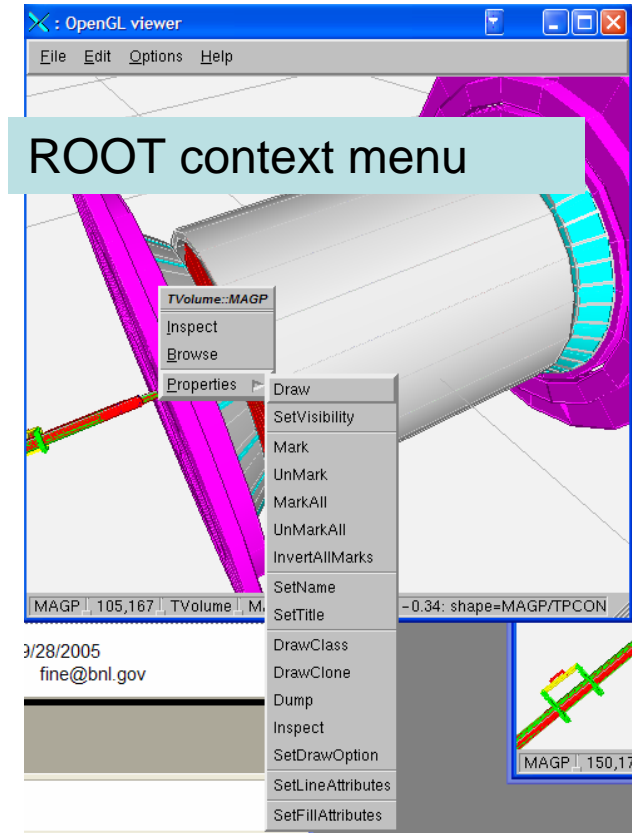
Status bar



"Full" wired



Satellite widget



ROOT context menu



# 3D customization

Is trivial and it is defined by the Qt and QGLViewer implementation, “philosophy” and design

- 3D viewer attributes – base class
  - Use – predefined subclasses
  - Subclass yourself and use
- Rendering
  - Ask controller to generate the view for the “known” ROOT classes
  - Provide your own custom view factory for your class or for the “know” ROOT class
  - Provide the Qt slot to respond the “QGLViewer “draw” Qt signal
- Selection ( like “Rendering”)
  - Provide the Qt slot to respond to the QGLViewer “select” signal
- Provide your own “controller”/”view” (see “UAL” talk )



# Installation

- ROOT bundle

```
./configure [...] --enable-qt
```

- Stand-alone, either from ROOT CVS or from full version BNL CVS

```
qmake <subdir>.pro  
make
```

See: <http://root.bnl.gov>



9/28/2005  
fine@bnl.gov

ROOT 2005 Workshop  
CERN



19

# Qt project and Qt designer

- Qt designer

Import the pre-defined Qt “Custom widget” definition:

```
$ROOTSYS/include/TQtWidget.cw
```

```
$ROOTSYS/include/TQtGLViewerWidget.cw
```

- **Qt “qmake” utility**

Include the qmake include file into your project

```
$ROOTSYS/include/rootcint.pri
```



# Documentation tools

- On-line documentation is prepared with the ROOT THtml class. In addition there is a [script](#) to post-process the output of the THtml class. It replaces all references of the Qt classes with the proper hyperlinks to TrollTech or QGLViewer online documentation. To be useful for the other application as well.
- There is a Qt section with the latest ROOT 4.04 User's Manual



# Readiness

- Works on UNIX and “native” Mac. It was tested on Windows with ROOT 4.00.08.
- 2D / 3D Qt Root widget are working and stable
- ROOT GUI needs work inside TGQt class that affects no end-user interface
- In other words, QtRoot interface is stable and it is safe to use it within Qt-based applications.



# Obstacles

- Splitting repository
    - inconvenient, cost some man-power
  - ROOT GUI “unusual” design
    - Normally, the ROOT philosophy (see: CERNLIB, PAW, ROOT I/O, ROOT 2D graphics, TSystem elsewhere) use the system feature as low / less as “reasonable”
    - ROOT GUI uses very X11 as hard as possible (Not a ROOT team fault by the way)
    - The same problem seen with the GL branch
- It has little impact on the “ROOT-based Qt application”



# To Do

- Find and fix the remaining bugs
- Negotiate ROOT team to adopt more project “pieces”
- To make Qt an optional plug-in (need remove compile-time dependency from the ROOT code.  
`#ifdef WIN32GDK`
- Add QtThreadImp class to the official CERN distribution.
- Improve support for mouse manipulation keyboard grabbing for vanilla ROOT Gui (That does not affect Qt application) – needs change on ROIT GUI side
- Qt Proxy for ROOT Editor classes



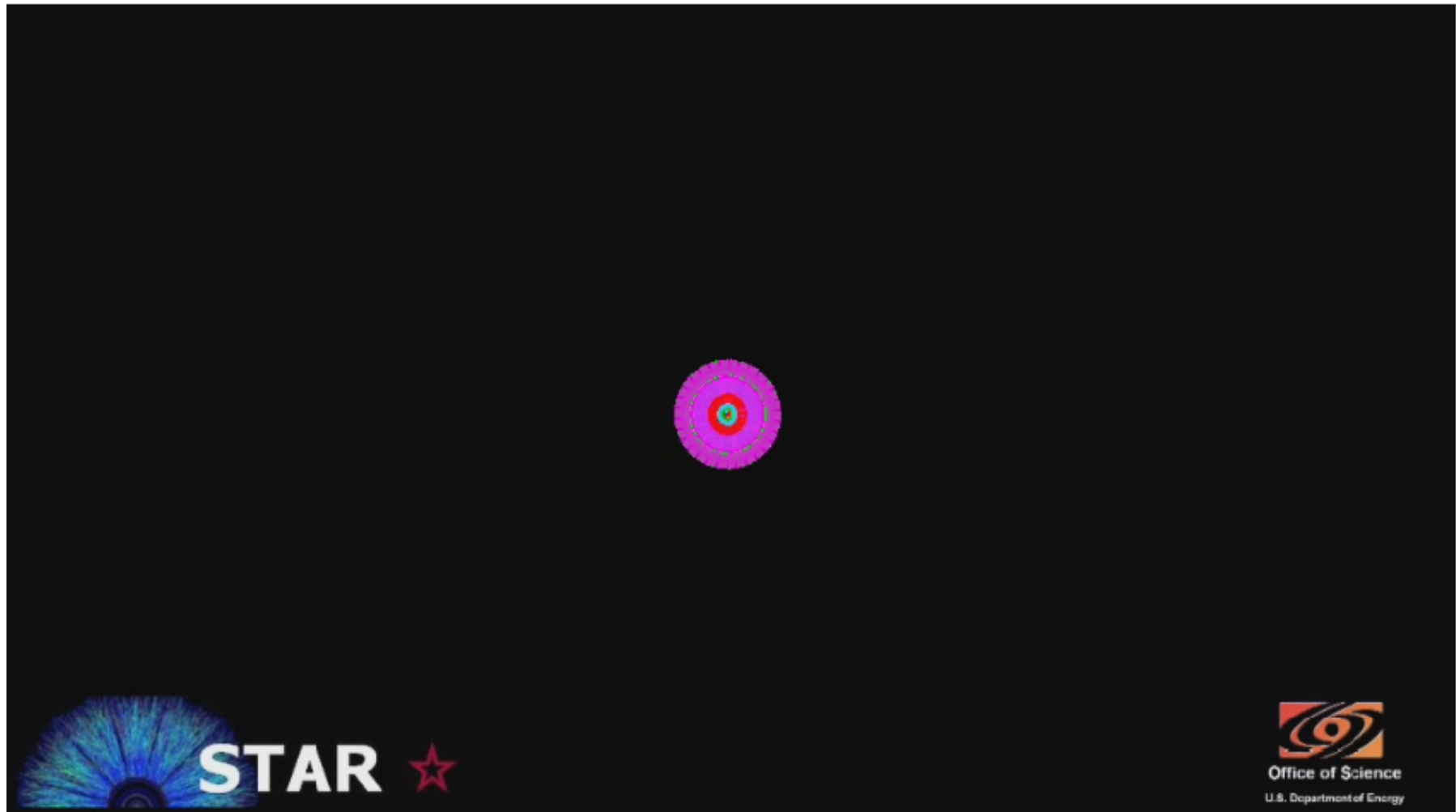


# References

- CINT status, Masa Goto  
<http://www.slac.stanford.edu/BFROOT/www/Computing/Distributed/ROOT2004/files/goto.ppt>
- Cross-platform approach to create the interactive application based on ROOT and Qt GUI libraries.  
<http://www-conf.kek.jp/acat03/prog/presen/id0112.ppt>
- ROOT in GO4, Joern Adamczewski  
<http://www.slac.stanford.edu/BFROOT/www/Computing/Distributed/ROOT2004/files/adamczewski.ppt>
- Cross-platform Qt-based implementation of low level GUI layer of ROOT  
<http://acat02.sinp.msu.ru/presentations/fine/Acat2002.ppt>
- Cross-platform approach to create the interactive application based on ROOT and Qt GUI libraries  
<http://www-conf.kek.jp/acat03/prog/presen/id0112.ppt>
- Visualization of the ROOT 3D class objects with Open Inventor-like viewers  
<http://www-conf.kek.jp/acat03/prog/presen/id0113.ppt>
- C++ GUI Programming with Qt3  
[http://phptr.com/content/images/0131240722/downloads/blanchette\\_book.pdf](http://phptr.com/content/images/0131240722/downloads/blanchette_book.pdf)
- ROOT 4.04 User's Manual



# Au-Au Collision (seen at DOE booth)



9/28/2005  
fine@bnl.gov

ROOT 2005 Workshop  
CERN



26