Panoramix

G. Barrand, LAL, Orsay, France*

Abstract

Panoramix [1] is a visualization environment for LHCb. We shall present technological choices behind this software : GUI, graphic, scripting, plotting. We shall present the connection to the framework (Gaudi) and the today status. We shall present the plans to integrate various analysis tools in order to work with DaVinci, the physic analysis environment for LHCb.

GOALS

Panoramix intents to handle event display and statistical visualization in a consistent way. Panoramix had been subjected to a software agreement between LAL and LHCb in May 2002.

POSITION IN LHCB SOFTWARE

Panoramix is on top a whole hierarchy of softwares : SEAL, POOL, Gaudi, LHCbCore. From LHCbCore someone can visualize the detector, described in XML, and data of the event model. The reconstruction is done with the Brunel project. Panoramix uses now some of the Brunel packages to visualize data of the event model (RICH). The DaVinci project is the place where the physic analysis really takes place, where people enter the physic. Panoramix has now connection to DaVinci and can visualize some of its data (Particle class). Put all together, Panoramix is "On top of EVERYTHING" in the LHCb software environment. A direct consequence of that is all sublayers must be up and running so that someone can work...

ENGINEERING CHOICES

Panoramix is plugged directly to the Gaudi event model manager (or data framework) in order to be fluent in accessing data. We do not attempt to have a client / server model by passing representations through files or the network. Since the data framework is in C++ and that most of native visualization and GUI tools coming from desktop providers are in C or a derivative (C++, ObjectiveC), we use C++ for all. We target the three laptops of the moment : Linuxes, Windows, MacOSXs and this by using their native environment in order to get the best of desktop providers. We use the OpenInventor C++ scene manager for all 3D or 2D data representations (including plotting). And last we try to put in "reusable elsewhere packages" the code not fully dependant of the experiment. Here the "reusable elsewhere packages" are the ones put in the OpenScientist integration and distribution. See OpenScientist web site or CHEP'04 paper and slides for more on the engineering choices [2].

VISUALIZATION AND GUI PRODUCTS

For the rendering layer we use more than ever OpenGL which comes with desktop provider products. For the scene graph manager we use OpenInventor by using the GPL Coin implementation of the System In Motion company (Norway) [3]. We do all the graphic with the uppers, including histogram and function plotting. For the GUI, see OpenScientist slides and paper for the problematic of a choice. We describe the GUI in XML and use the OnX package to create the GUI by using native toolkit of desktop providers. This is simple and give the best performances. The callbacks are scripted. Python is used as long a simple string front end to the system dynamic loading (dlopen, dlsym on Linux). An XML GUI callback looks like :

```
<widget class="MenuItem">
  <label>Forward+match+unique</label>
  <activate exec="Python">
    from Panoramix import *
    sys_import('my_tracks')
  </activate>
</widget>
```

or :

```
<widget class="MenuItem">
  <label>B Decay</label>
  <activate exec="C++">
    Panoramix B_decay
  </activate>
</widget>
```

The first menu item will execute the my_tracks.py user Python script and the second, the C++ compiled callback function "B_decay" put in the Panoramix callback DLL. The signature of a callback function is of the form :

```
extern "C" {
  void B_decay(IUI& aUI,
    const std::vector<std::string>& Args) {
    // User code.
  }
}
```

Note that for Python we use the LCGDict wrapping to wrap (strangle) C++ code.

* barrand@lal.in2p3.fr

GUI PARADIGM

The GUI is organized ”a la PowerPoint”, that is to say one compact GUI panel organized around a document area made with a stack of Inventor viewers (the slides). At left of the document area ther is a data tree browsing widget, at the top a menu bar and at the bottom a command typing area. Various dialog panels can be mapped through the menu bar items in order to parametrize and trigger an action (like printing, changing modeling parameters, etc...).

CONNECTION TO THE DATA FRAMEWORK

”data framework” should be understood as the software permitting to manipulate the event and detector models and permitting to connect the event (”data’) and detector models to facilities like storage, graphic, GUI, scripting. The data framework is here Gaudi. Gaudi does not do any kind of graphic (no Gaudi::DataObject::draw() method around, which is correct). Gaudi manipulates services, converters, algorithms,etc... that permit someone to operate the ”transient store” containing in memory instances of the event model.

OnX is here the ”interactivity framework”. It does the connection between GUI (from an XML description), viewers, scene manager (Inventor), renderer (OpenGL) and scripting. The connection ”data framework” / ”interactivity framework” is done through the Gaudi OnX service, put in a standalone package, the Vis/OnXSvc. The various elements of the event model have a ”representation” (a datarep). A datarep code is in general a Gaudi converter for the Inventor technology (an SoConverter). An SoConverter, for example SoMCParticle, builds from a data instance (here an MCParticle) an Inventor scene graph. When built, the scene graph is send to OnX to be displayed, in general, in the ”current viewer”. A visualization request starts from a scripted GUI callback ; the OnXSvc is searched, then a ”data convertion for Inventor” request is activated for various pieces of selected data.

It must be pointed out that due to a massive usage of abstract interfaces, various parts are nicely decoupled. For example an SoConverter is not a StorageConverter. A direct consequence is that someone can get rid of a technology and migrate to a new one, without touching to everything. It permits also to open the system to various other visualization technologies. IT PERMITS TO EVOLVE.

THE VIS/SO PACKAGES

Physicists enter into action by programming some SoConverters put in some Vis/So package. For the moment exists the SoDet (G.Barrand, S.Ponce) to build Inventor representation of the detector. This package is very generic and is now sparely touched ; used with the LHCb XML detector description it offers a very flexible way to enter and view a geometry. Note that there are various interactive goodies that permits to ”expand” or ”contract” volumes (from J.Boudreau SoDetectorTreeKit). The GUI can view the detector and event tree in the browsing tree widget.

The SoEvent package is historical and had been used to gather first datareps like SoMCParticle, SoMCHit (P.Mato) and representations for the trackers (J.Van.Tilburg NIKHEF). The SoCalo (I.Belyaev ITEP) is a dedicated package to build representations for calorimeters. The SoRich (C.Jones Cambridge) is for representations for the RICH data. The SoStat (G.Barrand LAL) permits to represent with Inventor histograms put in the Gaudi transient store ; it uses the HEPVis SoPlotter Inventor nodekit. And at last there is the SoHepMC used to visualize HepMCs and used in conjunction of the LHCb Geant4 simulation (Gauss).

We must point out that people did not seem to suffer of Inventor programming.

APPLICATIONS

Two packages remain to present : Vis/Panoramix and Vis/LaJoconde. Panoramix contains the Panoramix_main.exe program which is a standard Gaudi main. This package contains also the Panoramix.onx and other *.onx XML files describing the GUI. It contains also some C++ callbacks along with various Python scripts sufficiently generals to be put in the common pot. It is in scripts and callbacks that people can setup their views (number of regions with their size and position, background colors, etc...) and decide of what they put in their scenes. This could be done in Python or directly in C++ in case some Python wrapping is lacking, some too tricky C++ is needed (no comment) or intensive speed is required.

The LaJoconde package had been introduced historically to connect to ”The DaVinci Code” (where the secrets of LHCb physic is hidden sometime in mysterious algorithms). It had been introduced for coarse graining organization but will probably be transferred back to Panoramix in future so that physicists manipulate only one front end interactive program. This will depend of the capabilities to build coarse graining job options files in a way to connect/disconnet easily the access to DaVinci.

SPEED

Due to basic choices (stick to desktop providers and be plugged directly to the data framework) the system is fast (and the author is ready to challenge anybody claiming to have something faster). BUT, we must always ”have an eye” on Inventor. It is a powerful tool and then can be dangerous. A scene graph may become too fat and slow down the creation of a data representation. But there are numerous ways to optimize the size of scene graphs (share color, line styles nodes, create dedicated nodes to handle huge collections of data, etc...).

Experience shows that on local machines scene graph with 10000 up to 100000 nodes are ok. For for more orders

of magnitude, things must be optimized. But someone must be aware that a crowdy scene graph oftenly means a crowdy picture on the screen. Then the secret of being fluent is not in being able to handle big scene graphs ; it is in being fluent in selection into the data of what is really relevant. Up to now, at least for what is done in LHCb, speed is sufficient. (The author is more worried by what is (not) done in another more bigger experiment around)

PROBLEM AROUND GEOMETRIC BOOLEAN OPERATIONS

Problem remains around the visualization of some particular cases of boolean operations over volumes (we call that the "coplanar faces" problem). Dixit E.Chernyaev, the author of the algorithm (the BooleanProcessor of Geant4) there are cases where things cannot be decided (easily) and the best way is to avoid to have coplanar faces on some operations. Fine, but it is hard to explain to the detector people that they have to add or remove some matter here or there in order to pass a visualization algorithm.

A non-algorithmic solution is perhaps possible by using the stencil capabilities of OpenGL. Code had been found in Australia but not yet put in an Inventor node and integrated / tested in the system. A side effect would be that it will probably kill the vector PostScript production for these views. During CHEP'04 the author had noticed that the CMS visualization team had found another way to deal with this problem (by working on triangles) ; to be explored...

INSTALLATION AND DISTRIBUTION

We are not yet fluent in fully stand alone installation on a (remote) local machine. We remember that a display is on top of everything and that the three common desktop are targeted. Then be fluent in installing the display passes by being fluent in installing all the below. Today, binaries (Linux, Windows) are regularly reconstructed on the /afs/cern.ch areas (lot of thanks to Florence Ranjard) and users run them on their local machine. (Doing remote X11 on lxplus is definitely not recommended). Then we use CERN as a central integration / binary construction / distribution place and THIS IS GOOD. (The author strongly believe that this is the number one role of CERN). But we do not use CERN at run time but use local machines seeing CERN with afs. In this schema CERN is anyway around at run time as a STRONG afs server provider. For more, we strongly rely on efforts done around CMT (and PACMAN ?) to provide at some time straight forward source or binary local installation of everything. (See CHEP'04 Christian Arnault poster about ATLAS software distribution).

LCG, WHERE DO WE GO ?

Before LCG, situation around the basic layers where simple. We had Gaudi and the GaudiRootDb to handle event file reading. At some moment the author had even a way to read event by using the light Rio package (see CHEP'04 Rio paper). Now, the situation is more complicated ; Gaudi is over SEAL and "POOL over ROOT" is used for storage. Panoramix is then on top of much more things not really deeply though to work together. The port on MacOSX had been a pure nightmare and problems still remains for that platform : the only presence of ROOT induces to run in "bind at launch" mode and that slows down the startup of everything. There are also mysteries around dynamic loading and execution of static object constructors done with Gaudi and SEAL / PluginManager

SURVEY BY A PHYSICIST

In the software agreement, was mentioned that a physicist should be involved (at partial time) to do the synchronisation engineering / physic during the whole story. This never happened. For example, despite that Panoramix received code contributions from numerous people and that it is used by various people, no physicist "strongly visualization driven" spontaneously emerged to survey that critical elements of the event model received a representation. The software engineer (the plumber) can't have all the LHCb detector along with all the event model and physic algorithms in head in order to decide what and how things should be represented. It would be welcome to avoid a last minute rush on that point.

DOCUMENTATION

The specific Panoramix html pages and paper documentation are produced with Doxygen from dedicated .h files put in the Panoramix package. Some of the files are loadable from the program to form a help pull down menu and contextual helps. For an SoConverter programmer, the doc of Gaudi, Inventor and some of the OpenScientist packages (like OnX) are available online and on paper (for Inventor there are the Adison and Wesley books [4]). The nice user web page of Thomas Ruf could be found on the web [5]. Is it a basement for a wiki ? About the available representations of the LHCb event model, the content of the documentation is clearly insufficient ; but is it to the software engineer to produce this part of the documentation ?

INTEGRATION OF STATISTICAL TOOLS

Someone can browse (for long now) the "stat" tree of the Gaudi transient store and click an item to visualize an histogram. The plotter used is the HEPVis/SoPlotter one done with Inventor. Efforts had been done in order to cover good part of the well known plotting cases used in HEP (see CHEP'04 OpenPAW paper). In the LaJoconde GUI someone has already a dedicated panel to visualize algorithms (and their properties) and reinitialize them. We have to compleete this panel in order to change the properties of algorithms. Someone can loop on events. Clearly an in-

tegrated GUI tool permitting to visualize and program the chaining of the algorithms would be ideal.

INTEGRATION OF OPENPAW

The integration of OpenPAW (based on the same technologies) should be straightforward. It will consist to make the Gaudi/stat tree accessible from the KUIP commands in order that someone can type :

```
OPAW> CD //GAUDI/STAT
OPAW> H/PLOT MY_DAVINCI_10
OPAW> NT/PLOT MY_TUPLE.(E*1000) PT>1000
```

This integration is forseen for end of 2004.

INTEGRATION OF OTHER TOOLS

Had been demonstrated than from a python shell, someone can spawn a DaVinci task with Panoramix (to display an event or plot an histo), Hippodraw and ROOT. Then physicists will have tools to plot and fit histograms. But, are we going to be happy with that ? Probably not. A physicist will be let with a desktop crowded with standalone GUI windows, each having its own logic. At the engineering point of view the situation is not going to be satisfactory too because all these tools have incompatible backend technologies (rendering, GUI). At LAL, some job (low priority) is done to try to integrate some of these tools within Panoramix. The way to do will be the same for all ; arrange to have for these tools an OpenGL (or Inventor) driver to capture their visualization in one of the Panoramix viewer (here an OpenGL or Inventor native widget managed by OnX) ; after try to integrate their GUI in a way or another, the best being to have an OnX XML description for them.

HIPPODRAW INTEGRATION

LHCb people showed some interest to this tool. The design of it is so that someone can easily get rid of the default GUI and rendering (all in Qt), and comes with new ones. The author had given the basement of an OpenGL driver for it and an OnX description of the famous ”Inspector panel” starts to exist at LAL (OpenScientist / Hippo package).

ROOT INTEGRATION

The integration of ROOT will go through an implementation over Inventor of the painful TVirtualPad (some kind of strange mixture of GUI, rendering and scene manager virtual methods). A prototype exists (OpenScientist / Mangrove package). With the SoRootPad nodekit, someone can visualize a TGeo with OnX and Inventor and also the bins of a TH. But more work have to be done to have the full THistPainter renders over Inventor (or over OpenGL). But this is feasible. Any volunteer ? One ideal situation would be that the ROOT team provides an OpenGL driver for the plotting (today this exists only for the ”3D”).

JAVA, FREEHEP, JAS

Here the situation is more difficult, not so much because of the language but more because of the default GUI system promoted by the java camp (AWT/SWING). Up to now it had not been possible to integrate Inventor viewers within an AWT/SWING GUI and then to use jas as a GUI integratation place. The other way around seems not so easy too ; that is to say capture the plotting of FreeHEP/jas to have it displayed in native windows managed by OnX. Then strong expertise would be required here to have a direct integration. Someone may though to have indirect integration, like converting Inventor scene graph to java3d or FreeHEP graphic system and vis versa, but this will clearly be less fluent at run time.

ATLAS AGORA PROTOTYPE

Must be mentionned here that at LAL the AGORA prototype had been done in order to use the Panoramix visualization environment over the ATLAS Athena framework (based also on Gaudi). Some piece of detector had been seen along that a couple of representation of the ATLAS event model.

CONCLUSIONS

Panoramix is regularly released and used and had already given services (for the XML detector description debugging and the reconstruction debugging). Even if not yet an overflow, views appear in meetings and workshops and some are very very nice. Engineering choices are done (let us say up to the startup). But the overall architecture will permit to migrate parts if needed. More work around the release building and distribution are needed if the three common user laptops are targeted (Linux, Windows, Mac). At the engineering level, efforts will be put now on the final (!) integration of statistical tools to work on DaVinci : OpenPAW for sure, hippodraw probably, ROOT without conviction. Rendez vous at next CHEP for the status of all that.

REFERENCES

[1] http://www.lal.in2p3.fr/Panoramix.

[2] http://www.lal.in2p3.fr/OpenScientist.

[3] http://www.coin3d.org

[4] OpenInventor Mentor and Toolmaker. Josie Wernecke Addison Wesley.

[5] http://lhcb-reconstruction.web.cern.ch/lhcb-reconstruction/Panoramix/Running_Panoramix.htm