The Frameworks of Wrath

Guy Barrand
CNRS/IN2P3/LAL

April 2017

Abstract

In the HEP Software Foundation (HSF) visualization group, people propose to have a collective work to define a common "exchange file format". It seems to me that this group should first have as a primary target to promote deep changes in the today frameworks of the HEP experiments so that event display makers can build nice and portable applications permitting to enjoy in an effective way all new technologies that exist now to do visualization.

1 An exchange file format?

I do not share the idea that a visualization group of a HEP foundation for software has more or less as primary target to define "yet another file format". It seems to me that HEP software is in a so poor situation that there are more urgent things to do than to add an extra layer of complications. There is a lot of new nice interactive and visualization technologiess now and force to state that there is no way, just no way to enjoy them properly in HEP. No way to read and exploit an event file natively on macOS, Windows, iOS, Android which constitute 99% of interactive devices that people have in hands now. I even do not want to mention wall of screens, virtual reality and augmented reality. No way then to do intensive visualization where the graphical computing power is, mainly in the hands and on the head of people. What is the point of a framework if it does not permit to read an event file and interact with data straight on devices close to people? No way for the moment to have experiment detector and event models there.

It seems to me that the urgent point to attack is all these fancy knotty heteroclite undrinkable and monstruous bags of... code that some dare to call "frameworks" and that we are compelled to build onto to do event displays, visualization for simulation, visualization for reconstruction, outreach and all that. These frameworks are just a pure engineering shame and some sociological structure that dare to call itself a "foundation" for HEP should do something here.

We have already all the needed file formats to export graphical primitives toward viewers that are agnostic to physics (postscript, jpeg, png, VRML, Inventor, FBX and a lot of others) (there is probably a new one per day). But an agnostic-viewer does not know that a line is a track or a box a calorimeter cell, and then is unable to give us, for example, the energy associated when picking a line or the box. For that you need more, for that you need the "event data" and also the detector and event models. For sure you may say: ok my framework is a forever pain, I am going to write a viewer-knowing-physics (a not-agnostic-viewer) not tied to the painfull-framework and arrange to pass some file-format-with-physics in order to recover some physics in my viewer-knowing-physics. Sure you can do that, and some chose this strategy: WIRED on Babar, Atlantis on ATLAS. On Babar they invented the "heprep" and in Atlantis the "JiveXML". In JiveXML you can find the words energy, cluster, cell, then it is a format-with-physics.

But if you do that, you finish to duplicate the framework! You have to reinvent a file format beside the one used for the event file, you have to write/find code to write/read these files and then duplicate what is done in the fancy-framework to read an event file. You have to also duplicate in the viewer the code of the event and detector models to pass, for example, from a "cluster id" in your XML file to some boxes in position in a 2D or 3D space representing the "id".

I had been taught that duplicating information in software and computing is never good.

Ok, you may fall on java or python integrists (jihad came from computing in fact) that absolutely want to do a viewer in java or with the sneaky python, despite that the plumbing (the framework) is in C++ and graphics layers are in C/C++ too. Because they do not want, beurk, to link their things to any C++, they finish to introduce a client/server architecture and then a format-with-physics to do that. Ok, these people want to complicate their life by having two languages in the story, one being not compiled, a client/server architecture, maintain or pay for servers somewhere, have an extra format and a hell of extra code to visualise a track. Why not, it is their life. But the problem is that the "exchange file format with physics" introduced here finishes to be an excuse to not do the necessary cleanup in the plumbing that would permit to build very effective event displays built straigh on the plumbing and graphics drivers. (Arrange as you want, at each layer that you add between the data and the screen, you loose reactivity, and the client/serveur model add layers). To do intensive visualization you must stick to the data and the silicium and right now we can't do that properly in HEP experiments. An "exchange format with physics" may be in the way of effectiveness.

Layers, and passing by files, induce technicalities and coding that cost time, time which is then not passed in creating nice representations of data that speaks to a physicist. Activity which is the essence of data visualization in HEP and elsewhere. Your time is passed on secondary problems instead of focusing on the essential.

In all the presentations about event display shown in March it seems to me that CMS with its Fireworks goes in the right direction. From what I understand from the slides,

Fireworks is built on some "light framework" which is perhaps close of the "minimum framework" needed to do an effective event display straight on the data. A minimum framework being defined as composed of the IO system to read an event file and the event and detector models. If so the event display makers here had been eared by their core software team to get that. If so, good for them, it is the way to go. Good boys.

2 More on CMS/Fireworks

I played the game to activate Fireworks on my Mac. It was a nice surprise to see that it installed and run smoothly without problem. Fine to be able to install it without having to "sign with blood" to the "collaboration". I saw, at last, some HEP events with an application running locally on my Mac. The size of the binary distribution is reasonnable (222 Mbytes for the tar.gz). It is a little bit too much for a mobile app on the stores, but I saw that unfolded 110 Mbytes are to embark the whole CERN-ROOT (and 35 Mbytes for the libcling-clang-clong interpreter alone). Is all here really needed? Could some weeding be done? The startup is 15 secs, which is a little bit sluggish. I learnt that if dealing with interactivity, after a click/touch/return someone expect some answer within three seconds and if having not, he starts to be nervous. It is pretty true, I took that as a rule of thumb reference timing for interactive things. Then for a better user experience I would suggest to arrange to have the GUI appearing as soon as possible and then do the rest of the startup after that. It is purely psychological, but it would improve the user experience. Else the "next event" is as fast as expected on a local machine, good. (About running on the Mac, distribution size and timing startup, I even do not want to think to compare to pit 8).

The Fireworks distribution comes with an event file that permits to show straight something, and there is a workable "File/Open". This is fine too. Hell, you can't imagine the pain to get that on LHCb. Ages to have the modifications in Gaudi in order to be able to do a "File/Open" menu item. (I even finished to doubt if someone in the core software team grasped the idea of the mouse).

For the Mac I would suggest to have a clickable "Fireworks.app". The installation and startup would be even more straightforward (mainly in two clicks). With that a submission for the MacAppStore would be at hand which would give an even better visibility. Then probably CMS would be the first experiment to have a full HEP event display here.

(After all, there is perhaps Hope, some Awakening of the Force in HEP...)

3 LHCb/Panoramix

We could have an outstanding event display for long for LHCb. Especially due to the fact that the particular asymmetric geometry of LHCb is an added value that permits to do very aesthetic views.

Fifteen years ago, technically all pieces for GUI and graphics were around and I had the strong conviction that the LHC would be a tremendous party for graphics (event display everywhere, etc...). The disapointement had been high. What I did not see coming had been what happened around the "data framework" composed of CERN-ROOT, Gaudi and the "LHCb soft" for the event and detector models. I did not anticipate that, at the single moment that I had two incompatible pieces in the plumbing (technically and in the sociology) I would have to build on some messy knotty basement. Adding python in the cauldron did not help. It spoiled too the magic potion recipe. All these led to some bad tasting soup that finished to be tied to only one operating system (Linux) leading to the fact that right now there is no way, just no way, to run natively on all interactive devices that people have in hands now. Well, I say "Linux", but it is in fact tied to a distribution, worst to a version of a distribution. A lot of time had been passed to have nice data representations for good part of things, but no way for me and users to enjoy all the nice technologies full speed that we have now. Years of programming for few rewards at end. The magic potion had not been delivered, and the cooker returned in his hut. Running natively on the Mac, an easy platform (it is a UNIX with gcc and clang), had been blocked by the plumbers (the "core software team") for ages, and this long before virtual machines. (And you know what? Most of plumbers had and probably still have Macs!). Running on my Mac a fat and clumsy virtual machine? NO. I want to run straight the application on my machine. (A virtual machine is a way for some to hide under the carpet their inability to master their software). Handling in a effective way smartphones, tablets, laptops, desktops, wall of screens, virtual reality and augmented reality is pure science fiction here.

Did LHCb ever wanted an event display in fact?

At some point the release was 30 Gbytes, three times the size of an operating system as macOS! To do what? To open a file and draw a bunch of boxes and lines. It makes no sense. Madness. Startup time? Ages. (You know, around Geneva in winter you have to heat the engine of the car, melt the ice on the windshield, shovel the snow in the alley. Then the morning startup takes time, a tradition. The same for the raclette grill, and the same for a framework: it has to warmup first). (ROOT+Gaudi, the raclette framework, developed at the Jurassic lab. After all, it looks consistent).

All this is an interactive disaster. What solution remains now? Put all that on servers and do a web layer to hide the whole thing under the carpet: web carpet hiding software architecture, at last! With a probably poor remote reactivity when doing serious work. Web carpet hiding at which cost? Who is going to pay the servers? Knowing that people have already the visualization computing power in their hands.

Again: what is the point of a framework if we can't read the data (event and detector) on all the devices around?

About installation, I proposed ages ago to have some "installation challenge". The idea was to target and reach a situation where the display would be installable and workable on a local user machine with a minimum of clicks. Rejected. Another wall of bricks. It

seems that CMS people are pretty close to reach this goal, good for them.

(As for workers on a building site, a HEP event display maker should be equipped with a helmet but also with a madness film badge dosimeter).

For me, it had been obviously a huge lesson of "collaboration". Force to state that concerning software, a big collaboration is not a synonym of great engineering and high quality in the code, definitely not. Especially if there are in the same collaboration multiple Revealed Gods. A collaboration permits only to be complete, to cover all parts of a large detector, that's all. The result in the code is some kind of weird patchwork. For new display makers in the field, take care on what you are going to build onto, and of the various bad ingredients that some want to put in your cauldron. If having to sign with blood a "MOU" and it smells Sulphur around, read the little characters. It is clear that some foundation may help concerning the relationships of engineers and computer scientists with an experiment. A HEP collaboration is some kind of fermionic sociology, and a foundation should be a bosonic one. Can we build a bosonic sociology with fermionic people? What is the spin of people? Do we have supersymmetry in HEP sociology?

4 Why not writing a specific detector and event models for the visualization?

NO! The detector and event models are the spirit of the software of an experiment. They must be shareable and/or copyable. If not, the display maker is going to pass his time to import by hand changes, put by the detector specialists, in the primary place drowned in the framework. A pain. On mid and long term, the display finishes to be no more in sync with the data. The detector and event models must be thought from the beginning to be shareable and/or copyable for all kind of tasks: batch analyses and interactive displays.

5 Visualization by the web?

I have strong moods about enforcing the web for visualization, or any client/server architecture for visualization. Passing by web browsers is ok for outreach (including masterclass, education) but not for intensive visualization. It is not ok if faraway from the servers, for example at home. I tried that, then producing JavaScript with WebGL seen from my sofa. Ok "it works", but the reactivity is a true pain on real data if being remote. The user experience is a disaster on a large detector or on a noisy event or on an event with a large multiplicity. (A user experience similar as doing "remote lxplus" in the afternoon). A great user experience in interactivity is tied to the reactivity of the whole system. Arrange as you want, a web approach adds layers of hardware and software between the data and the user, each of them inducing a loss of reactivity. Targeting a maximum of reactivity locally is the way to go for intensive visualization.

The web or client/server can't be a primary target for a reactive visualization. The primary target should be to run locally as much as possible, and we must arrange to fix first the problem of portability of event reading and of detector and event models.

6 THE Framework of Wrath: CERN-ROOT

First, a strong positive point: I had and have nothing against the CERN-ROOT file format. Yes, yes. In the 1990s, it was clear that the problem of writing/reading HEP data in an effective way was on the table and a Cooper pair solved it. Great, applaud!

It is more the three millions of lines of code of the implementation that pose me problems. Along with the fundamentalist/extremist spirit that gravitates around it. Hard to have a reasoned argumentation with an extremist. For me a scientist is someone coming with a reasoned argumentation about something but, fundamental, that can change his mind if proved to be wrong. (If not, he is a pure dictator). Is there some scientist now around CERN-ROOT? In HEP software?

Hmmm, some may say that there are only Revealed Gods in CERN-ROOT. Then people that are, by principle, always right on everything. But seeing the index count in JIRA/ROOT, it is hard to believe that. (Incidentally, is there still an average of one bug per 20 lines of code?)

(For the Galaxy Note 7, at some point Samsung suspected some fundamentalists to have attempted to port CERN-ROOT on the exploded devices. But no, this time it was not them).

Some tell me that "now things had changed", really? Does the herd make a difference now between an interpreter and an IO system? Between an IO lib and a graphics lib? How much millions of lines of code (loc) do I have to compile and install now to open a CERN-ROOT file and write/read MY histo in it with the CERN implementation? (I do that with 20 kloc with my things). And to create one push-button? And to draw one line? And to do a plot of a 1D histo? (I do that with 30 kloc on most devices, including off-screen for the batch). Is the TH2 still a TH1? Are things properly namespaced now? For libs, do we have now:

```
lib_ROOT_Xxx.so
```

(instead of libXxx.so). Can I include now with:

```
#include <ROOT/H1D.h>
```

Is there still the fancy g auto management of everything? (The HEP acronym is now known to be for Highly Exotic Programming).

Is the "The ROOT of EVERYTHING" still here? (A case study sentence now in psychoanalysis).

Due to the importance of storing data in science and HEP in particular, it is long that all what is tied to the IO should have been isolated in a standalone unit. Something as a "massive OO storage library for scientific data". A "massive OO storage" being not an OO database. HEP scientists are not bankers (for sure). At the output of an experiment you need a "write very fast once", and then "read very fast many" when doing analyses. The data usage pattern is not the same as in a bank. An "event" is not a salary. But well, here too a wall of bricks. HDF5 is probably going to do it.

CMS/Fireworks uses CERN-ROOT for the GUI and graphics. (Naughty boy). Would it be easy, for example, to replace the GUI by Qt but keeping the ROOT graphics, and removing the libGui from the distribution? Or keeping the ROOT GUI but using the Qt3D, and then removing libGraf, etc..? Is there no more entanglement phenomena between CERN-ROOT libraries, so that we can be sure that coarse graining parts are not related, and that some can be isolated? Or are we still condemned to embark the whole thing whatever? Without few hope then to run something as Fireworks on iOS and Android.

Do you want to really build a foundation on that?

If so, the HSF has to change its logo and have the Tower of Pisa instead. How much did it cost up so far to avoid the Tower of Pisa to collapse? How much CERN-ROOT will cost to HEP?

In general now I put "CERN-" in front of the "ROOT" to remember that CERN, with the E of Europe in its name, let pass that for the LHC. "You take it (as it is) or you leave it". How could we have surrendered in front of that? History already judge, now HEP is out concerning a whole trend of interactive and visualization technologies. What next? (Soon I will have to vote in France, Frexit or not Frexit?). Will the HSF surrender too without compelling a strong refactoring of CERN-ROOT?

I did my weeding long time ago and I am faraway from that now: a relief. (But I still have a secateurs under my pillow and a huge barrel of Roundup in my garage, just in case).

To finish on a positive point: since people are interested in an exchange format, I point out that I can provide very light C++ code to read data in a CERN-ROOT file at the condition that these data are some "more or less fixed things" with a "human understandable streamers" as histos, arrays. I do that with my inlib/root pure header code. (The TTree cost me some time in an asylum but now I am ok. In fact here I met House M.D, he helped a lot, thanks Greg!). I can read TGeos for long (and display them on mobile devices since 2011). Then It would be probably ok for me (my analyst heavily disagrees) to extend my readers to read the TEve primitives if they are "fixed streamer-understandable", and then foster the CERN-ROOT binary format as an exchange format. (Yes, yes, again I have nothing against the file format in itself). At least it would be a binary format defacto much more effective that some XML or JSON ascii format. (Ascii format is ok for small things, but not for big ones).

The binary CERN-ROOT file format as a basement? Hmmm, why not. (Oups, my analyst just resigned) (and the man runs really fast...).

7 George, what else? Coin3D and ATLAS/VP1

And then, when a VP1 installable in two clicks on my Mac? On my iPad? On the MacAppStore? Here I have for long my agora app that can show some geometry from an old atlas.root file and the vertex from a JiveXML file, but... that's all.

About "viewers not tied to a fancy-framework but knowing HEP things", I have such apps running mainly everywhere, especially on iOS and Android, that can read root files (then with TGeo in them), heprep, GDML, VRML and Inventor ascii and binary files. My ioda app can do that for long. I may improve them with root files with TEve things.

For VRML and Inventor/HEPVis files, on iOS and Android I ported Coin3D and use it to get SoNodes in memory, do a traversal with an SoCallbackAction to get points, lines, triangles, and then use GL-ES to render all that. It works! I can visualise an LHCb event in this way by having first produced the .iv with Panoramix on the remote-afternoon-lxplus. It should work with ATLAS/VP1 too, and also for other displays around doing the graphics with Coin3D. But, I repeat that these apps are "iv viewers" and not event displays, we can't have "physics feedback picking". Hmmm, we may arrange something with SoInfo nodes, no? But even with that I would not call them event displays.

In fact we should state some deep truth: a viewer app knowing or not the physics but not straigh on some event and detector models is not an event display, it is a viewer. And the primary target should be the event displays, and then the frameworks. Another file format may be helpfull, but can't be a priority. The priority should be to organize some protest marches toward the 32, some sittings to block the cantines ("no data, no pizza", "no data, no pizza",...), have the anonymous on the back of root.cern.ch, etc... so that the changes be done in the whole plumbing. (Is there some FEMEN in HEP?) (The FEMHEP?)

(Hmmm, no the FEMEN is perhaps not a so good idea, some plumbers may forever add knots of tubes on purpose).

Incidentally, knowing that there are already file formats knowing HEP physics, do we need something more? For the geometry we have GDML and .root/TGeo, for the event heprep, why an extra one? For the pure pleasure to produce some ascii JSON? To enforce a web visualization?

8 JSON?

It is ascii. And why not the binary BSON? That would duplicate the CERN-ROOT file format of the event.

Even if choosing JSON, you have to define (as for XML) what are the objects, and then define the MY_XML or MY_JSON format that will need specific readers to understand the data. Syntax is not semantic. Do you want something human readable or not? If not, then do you want the object definitions automatically done from something else? From

CERN-ROOT? And then enforce (as usual) to the rest of us the class design (?) around graphics found here? Yes, because if doing that, it will enforce to the rest of us to have to understand these .jroot files in our viewers. It would be an overall bad thing if these files be understood only from the CERN-ROOT prompt or only in an application doing its graphics with CERN-ROOT. (I did my weeding, I do not want to be compelled with a gun in the back to have to link again my apps to anything of CERN-ROOT).

Will I have a chance, in my viewers, to read these files and understand their semantic on all my devices without becoming, once more, totally mad?

9 AVRO?

And why not AVRO? Which seems to be the IO buzzword around the big data big buzzword.

10 Visualization and plotting

I strongly disagree that plotting is not under the visualization umbrella. I know that some around believe to have the monopoly about that. ("The PLOTTER for EVERYTHING"). They are wrong. It would be a very bad point for the HSF that plotting be discussed only within some analysis group in which only CERN-ROOT, and then only CERN-ROOT plotting, would be discussed. (Again, how much lines of code to compile and install now to do a single plot of a 1D histo with CERN-ROOT?)

Qt people had said that they have plotting. If some around tried it, it could be fine to give feedback (and code examples)!

Obviously there is all the activity around plotting in the sneaky python world. Can I embed straight the graphics of matplotlib in a Qt or gtk window, or a GL-ES environment? (Straight meaning without passing through the cut and paste of a jpeg or png image). Can I have a matplotlib plot in my event? etc...

There is a lot of things to look these days dealing with plotting and visualization.

11 Frameworks and the Higgs

The 4th of July 2012, honestly, I had a huge doubt. Was the bump a systematic from the software or not? Five years later people look confident that it is not.

12 Mobile devices

It is just amazing that in the last CHEP papers and in the slides of the first March HSF visualization workshop, the words iOS and Android were just not existing. Then ten to seven years after their appearance in the computing world. Other communities are much more reactive, why?

Some time ago, I did a presentation "Feedback on some iOS and Android usage in HEP and Astronomy" at the JoSy days in Paris dedicated to "Mobile technologies: feedback and prospective". I have now an English version of the slides that can be found at:

`http://softinex.lal.in2p3.fr`

under the section Presentations (bottom left):

`Feedback on some usage of iOS and Android in HEP and Astronomy.`

It may interest some. At least to understand why I pass now good part of my time in an other area of science where, at least, I can read a data file on my devices. It should be at:

`http://softinex.lal.in2p3.fr/download/com/G_Barrand_JoSy_Oct_2016_English.pdf`

13 Conclusions

It is astounding to see how humankind can complicate its life to draw a line representing a track. HEP is clearly a champion here.

Again and again: what is the point of a framework that does not permit me to read data on my device? People interested in HEP and visualization should concentrate on that first and arrange, in particular through the HSF, that some "portable minimum" be always provided on any experiment by their core software team for doing visualization in an effective and proper way. Even if nobody at end use the display to scan any event, the simple fact to have such a display done in an effective way will induce better code everywhere, and then to be much more effective in doing an analysis in batch. The HSF should help to enforce that effective visualization be always in the starting requirements of an experiment. (It is in other areas of science).

Fostering now an activity around an exchange file format may be undsertood by plumbers as a fact that event display makers are happy with the today situation and surrender in front of their forever knotty productions to manage data. Somewhere it is going to be a "carpet hiding exchange file format", a poor solution to bypass the poor situation around frameworks

Then, for you, what should be the primary focus/target of a visualization group in a foundation for software in HEP? Be effective or do forever carpet hiding?

To all, it is spring, then don't you think it could be a good time to do a spring cleaning in your HEP house, do weeding in your HEP garden, stop to be blind as a bat and then have an eagle eye in your physics.

(If you think to be already crystal clear with your basement and want to see a bit of detector and a bunch of tracks on your phone or tablet, let me know, I would be glad to try my window cleaner classes here).

14 Afterword

Then Panoramix returned back in his village, faraway from the center of the empire with roots a bit nauseous, preparing in his cauldron much more tasty soups, improving its magic potion recipe, weeding ever and ever his garden and whatching at night the starry sky. In some dark corner of his hut, a souvenir from the Mentions of the Gods: a battered helmet.