EsbRootView

A portable event display for ESSnuSB

(In 8 minutes!)

vCHEP 2021

Guy Barrand, CNRS/IN2P3/IJCLAB

ESSnuSB in (very, very) short

- ESS (European Spallation Source) : first a linear proton accelerator at Lund, Sweden. Second, through splintering of atoms by protons (=spallation), a pion source. Third, due to pion decay, a neutrino beam. First operation of the linac foreseen for 2023.
- ESSnuSB : R&D for a near detector to monitor the neutrino beam, and a far detector of 540 kt of water, positioned at the “second maximum of neutrino oscillation” (540 km from Lund in the Garpenberg mine (?)).
- Physics goal : measure CP violation at 2nd neutrino oscillation maximum.
- See two videos : “ESSnuSB looking for the answer” and "ESSnuSB Design Study Project”, on ESSnuSB main page or on YouTube.

softinex

- Around 2010, I bought the first iPad and I reconsidered the way to do graphics and GUI in order to be able to handle not only "classical desktops/laptops", but also the new wave of smartphones and tablets.
- Six months of work => a very portable C++ (98) set of classes, based on a "scene graph logic" (strongly inspired by OpenIventor), being able to render with GL-ES on most devices around : macOS/Cocoa, Linux/X11, Windows/WIN32 but also iOS and Android. (GL-ES being common to all of them), and also now in web browsers by using WebAssembly and WebGL.
- Code put under the name "softinex" (= software done with the inlib/exlib classes). Web portal : https://gbarrand.github.io

Apps : g4view, ioda, pmx

- Contact with the HEP physics by doing demonstrator apps.
- Geant4: it (10.3.x) appeared to be highly portable! Also on not yet supported platforms as Android, iOS and WebAssembly. g4exa and g4view apps.
- Contact also with “general purpose analysis tools”: histogramming, ntupling, plotting, IO. Write classes for these, including a very portable and light ROOT format IO reader. We can also read HDF5 files (and fits ones for astronomy). ioda app.
- Plotting done with our “scene graph” logic.
- GUI done also with it. We have a “unified graphics” leading then to the same look and feel on all platforms.
- But not yet a “true contact” with an experiment. (Despite the pmx app showing LHCb geometry and some events).

See also CHEP 2013, 2016, 2018 for presentations

MEMPHYS = > ESSnuSB/WP5 => EsbRootView

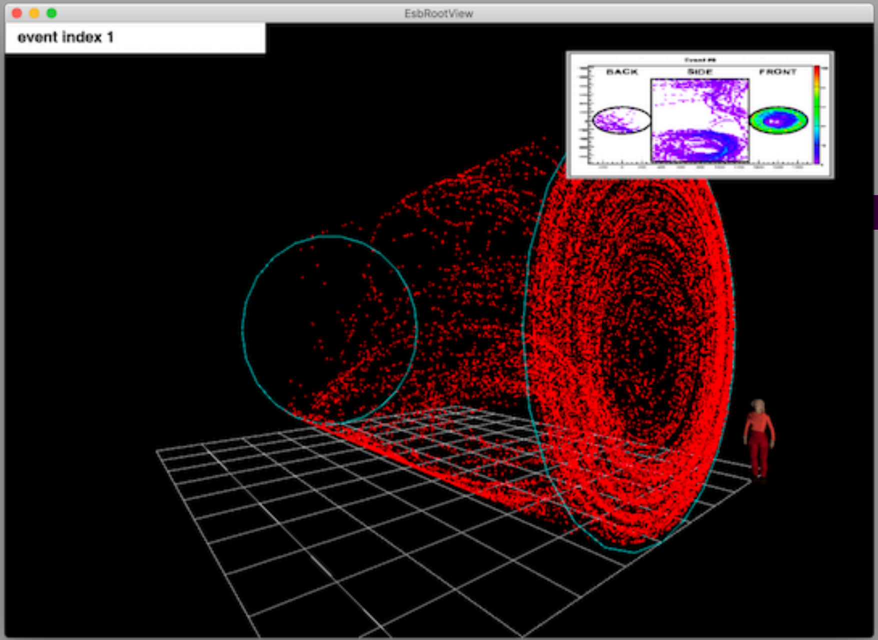
- In 2017, contacted by the ESSnuSB/WP5 team to help on reactivating for them the MEMPHYS G4 simulation and visualisation I worked on with J.E.Campagne at LAL around 2000-2005.
- Discussing with people at Uppsala, despite that they chose a ROOT based framework (EsbRoot, derived from FairRoot), they intended to deposit in ROOT files their event data by using common ROOT containers as TTree, TCloneArray, and TGeo classes for the detectors.
- Since I can read these by using my portable IO classes, we could think to a portable display using my technologies => EsbRootView.

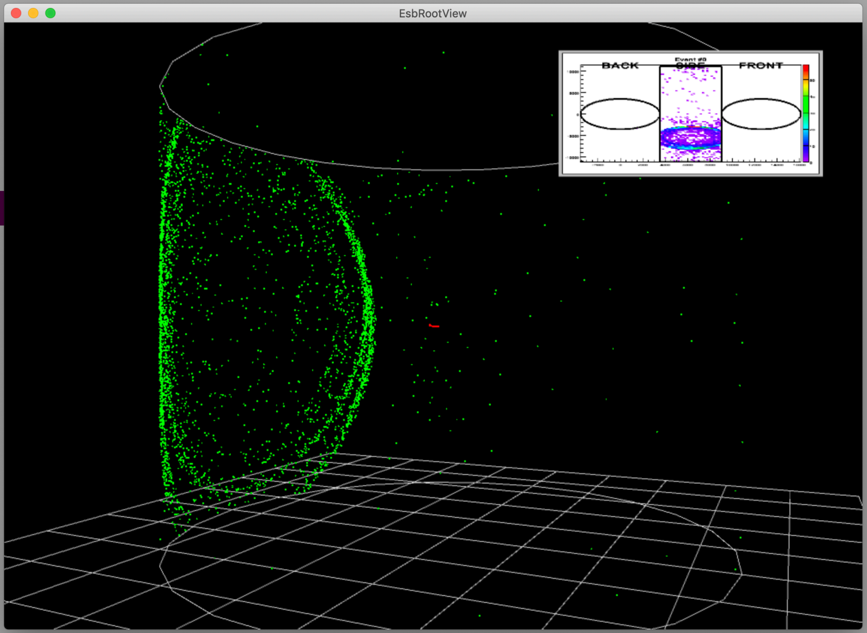
EsbRootView

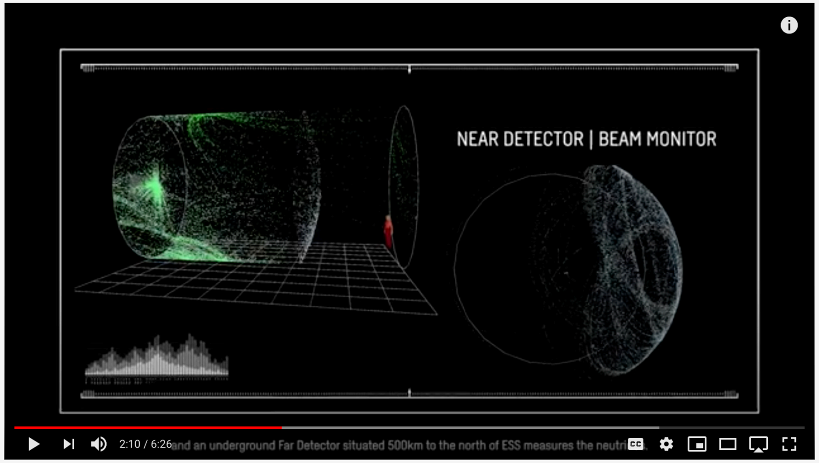
EsbRootView = a C++ highly portable ROOT file reader (event and detector) and renderer (by using GL-ES) knowing ESSnuSB data schemas.

Features (in very, very) short

- Visualisation of detector cylinders.
- Visualisation of MCTracks, WCDetectorPoints by using various representations.
- Strong customisation by using a bash-like scripting system (insh).
- Terminal insh command typing on desktops/laptops.
- Animation: deployment of Cherenkov rings. Nice! Could be seen in the outreach video “ESSnuSB Design Study Project”.
- Plotting (done also with scene graphs).
- Same GUI on all platforms (done also with scene graph).
- detector+event+plotting+GUI done with a “unified graphics”.

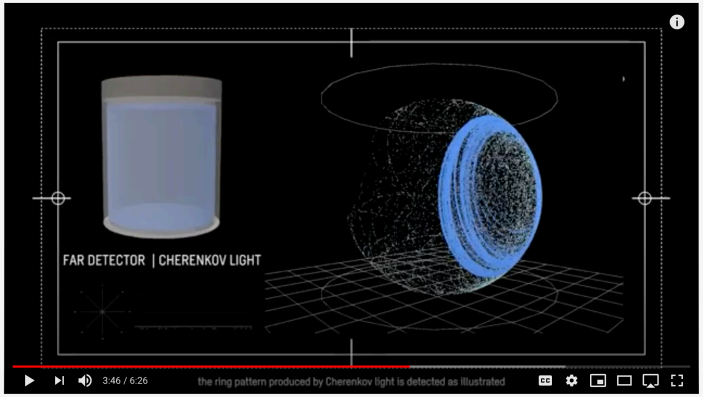






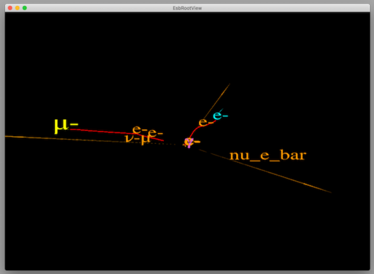
and an underground Far Detector situated 500km to the north of ESS measures the neutri

ESSnuSB Design Study Project



the ring pattern produced by Cherenkov light is detected as illustrated

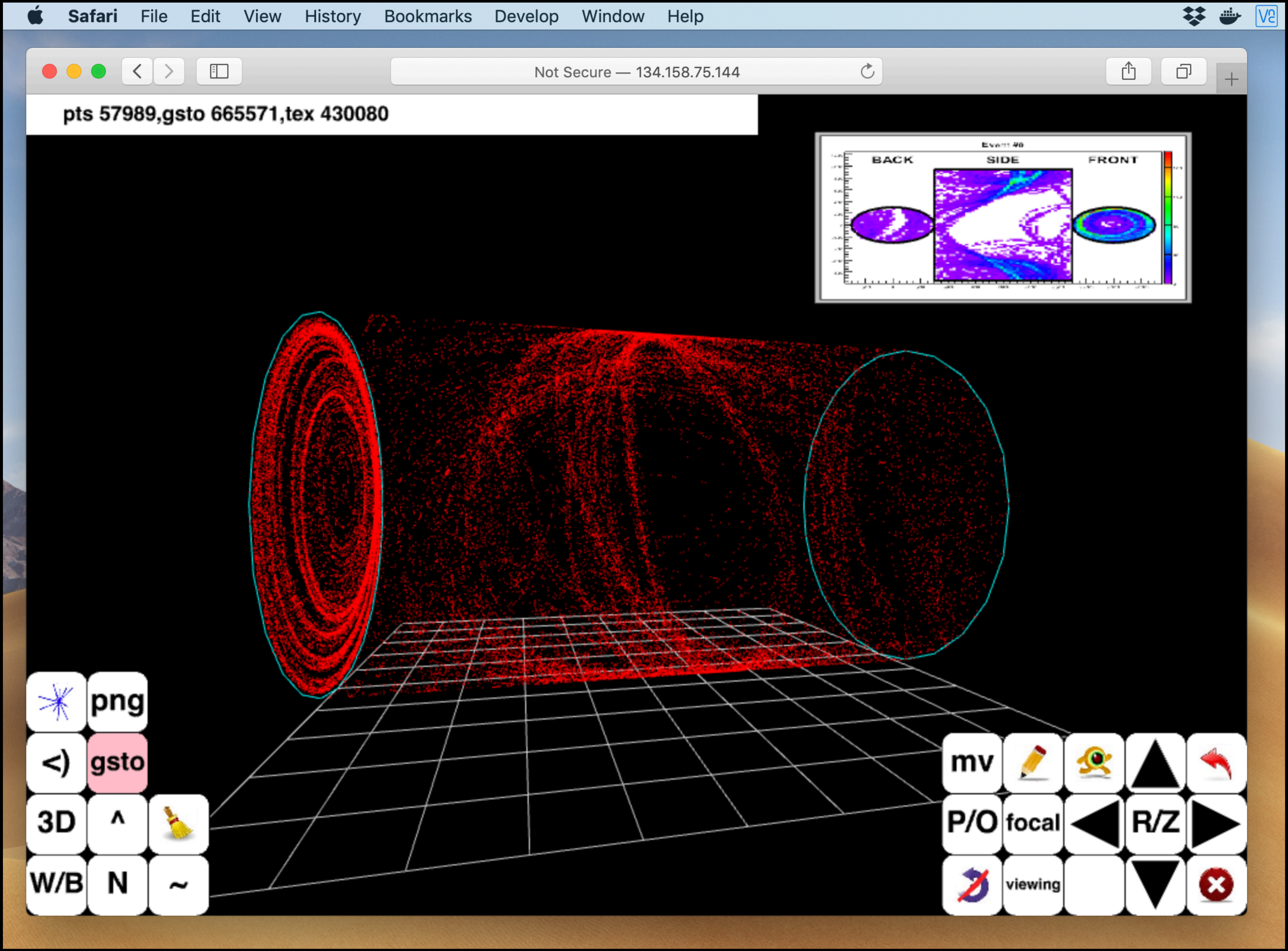
ESSnuSB Design Study Project



Guy Barrand, CNRS/IN2P3/IJCLAB

Also straight on the Web!

- By using WebAssembly.
- Thanks to Budimir Klicek (Zagreb) who made me aware of this.
- WebAssembly is some kind of virtual machine existing in all web browser permitting to execute cross-compiled C++ code and doing graphics with WebGL.
- EsbRootView had been ported here! Nice!
- No need to rewrite everything in javascript!
- Good performances (but quite a little bit less reactive than a native app).
- This offers a straightforward world wide deployment of this C++ display!
- g4exa, g4view, and then Geant4, run here too!



pts 57989,gsto 665571,tex 430080

Conclusions

- 95% of C++ code portable natively on most devices with the bonus of the WebAssembly version. (Remaining 5% is "constructor windowing code").
- With inlib/exlib, we cover a lot of what is needed to do visualisation for HEP, including plotting and analysis tools.
- Part of the code shared with Geant4 (g4tools).
- The situation is highly satisfactory at the engineering level.
- A drawback: it would be great to share the code of the event, detector models, and also the coworking critical IO service with "the batch" (= EsbRoot framework)…

Quick demo…(?)