

G4/vis/ToolsSG/Offscreen MinGW

G4 Rennes 2022 workshop

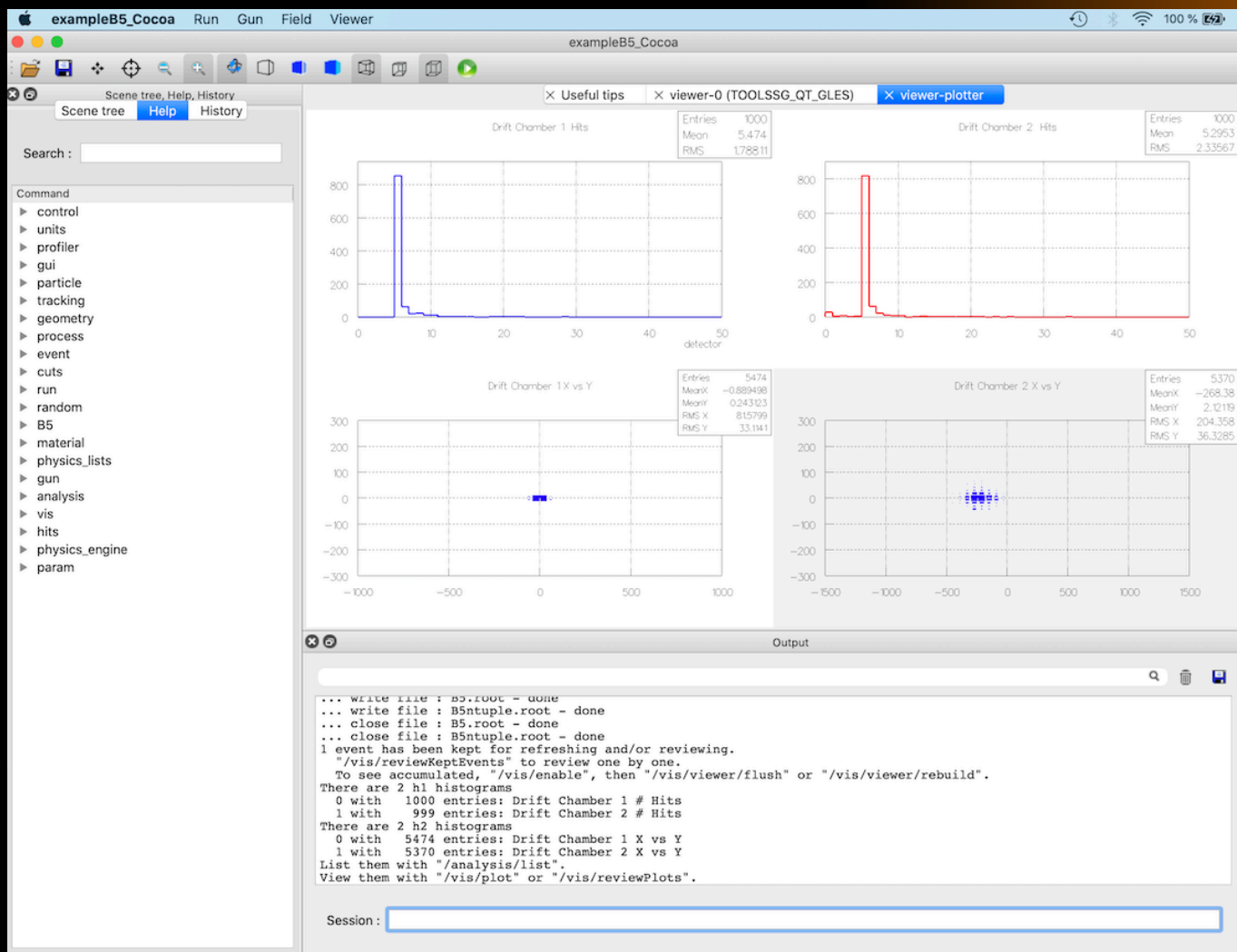


- A vis driver introduced in 2021 based on a **scene graph logic** developed at LAL (now IJCLab) since 2010, and for which the code is now in **g4tools**.
- Good part of the code is on C++ standard libs.
- Some rather light code in **toolx** to bind to various renderer and Windowing systems, today: OpenGL, X11, Xt, Windows, Qt.
- Few code in visualization/ToolsSG: only the “glue” to the general/generic G4/vis system.
- Try to keep some free “academic way” to do visualisation.

vis/ToolsSG plotting

- tools/sg contains a “plotter node” (then based on the tools/sg scene graph logic).
- Already used in G4/analysis for “batch plotting”.
- In G4/vis, had been introduced the **G4Plotter model** logic to be able to bind and activate this g4tools plotting from the **G4/vis system** and then also from the **G4 command system**.
- The **G4/vis/plotting** knows the **histos** in **G4/analysis**.
- (Not so easy to find the right way to connect all these!).
- See **examples/basic/B5** for an example.

vis/ToolsSG plotting (2)



vis/ToolsSG plotting (3)

- Today only 2D “flat” histo plotting, but more could come soon... (tools::sg::plotter can do 3D “lego” plotting and plots also functions).
- Today fully available with the ToolsSG driver, but within G4/vis, the logic is so (in particular the way to bind the G4/analysis histos) that other vis drivers “plotting able” (VTK?) may bring their plotting within the same architecture.

vis/ToolsSG rendering

- The **screen** rendering is today based on OpenGL (gluing code within **toolx/sg/GL_action**), but it is **virtualised at the level of the scene graphs**, so we can introduce other ones.
- For example Apple/Metal (when/if Apple strongly remove its OpenGL from macOS).
- Or an **offscreen** renderer...

TSG_OFFSCREEN

- Then a new “sub driver”: TSG_OFFSCREEN (beside TSG_[QT,X11,XT,WINDOWS]_GLES).
- To produce views (.ps, .png, .jpeg) straight from pure “batch”, without having to tie to some Windowing system.
- Purely on C++ standard libs (then no X11, Windows, Qt, OpenGL around). It uses tools/sg/gl2ps_action but also the tools/sg/zb_action, a zbuffer renderer already in g4tools.
- It uses tools/gl2ps, but also new tools/fpng, toojpeg to write at the png and jpeg file formats.
- Fully thread safe (including gl2ps and the png, jpeg writers).
- g4> /vis/open TSG_OFFSCREEN
- Code is here, I hope to submit a MR soon...

MinGW



MinGW/general ideas

- Following a recent MR about a port of Geant4 with MinGW, I had a look to **MinGW-w64**...
- On Windows it permits to build **Windows apps** by using **gcc** and **clang** (a MinGW flavour of them).
- From a CYGWIN prompt it is rather easy to install and operate.
- **BUT**, we can also install MinGW on a Linux or Mac to cross compile a Windows app from here!

MinGW on my apps

- I had a try with my apps and it worked nicely! With both MinGW-gcc and MinGW-clang on my Windows-11 (installed through CYGWIN) and MinGW-gcc on my Mac (installed with MacPorts).
- It is ok too with MinGW-gcc on my g4exa, g4view apps working on a g4-10.03.p01.
- With MinGW-clang, the compiler just crash on one of G4 file... (G4EmDNAChemistry.cc).
- Not so much `#ifdef __MINGW32__` to have that.

MinGW & G4 ui & vis ?

- From master, with some `#ifdef __MINGW32__`, I have been able, from my Mac, and with MinGW-gcc, to build examples/basic/B1 and B5 (with plotting) with:
 - G4UIWin32
 - OpenGLWin32
 - TOOLSSG_WINDOWS_GLEScopy the .exe and run it on my Windows-11!
- (For the moment, the build had been done with my “bush” build scripts, not cmake).

MinGW & G4 ui & vis & Qt ?

- From CYGWIN, we can install a MinGW-Qt5!
- I have been able to build (with bush), on the PC and MinGW-gcc, `examples/basic/B1` and `B5` (with plotting) with:
 - `G4UIQt`
 - `OpenGLQt`
 - `TOOLSSG_QT_GLES`and run them on my Windows-11!
- On Mac, MacPorts does not come with a MinGW-Qt5...

MinGW & G4 ui & vis ?

- Then at least MinGW-gcc is at hand with the Win32 UI & vis!
- And Qt, only with MinGW-gcc on the PC for the moment for me...
- **MinGW-clang?** On master, for the moment, it fails to compile PTL for me. (Not clear to me how to fix things. May be bugs in the compiler...).
- **MinGW-Qt5** on Mac?
- **G4/cmake** and **MinGW**? I think Ben & Gabriele are on it...
- For a developer, MinGW could be convenient, at least to check a Windows build from a Mac or a Linux...