

# **2022 - Resume of //** **Vis session**

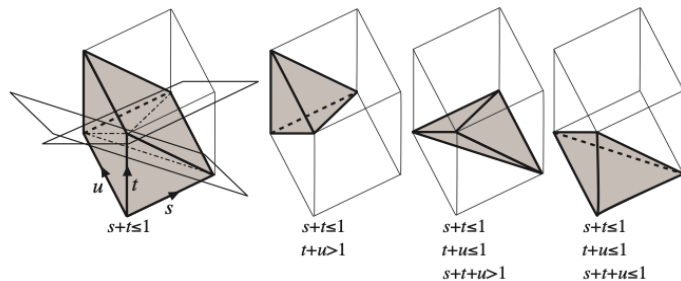
**Geant4 workshop - Rennes 2022**  
**Laurent Garnier - Vis group**

# John Alisson - Special Mesh rendering

- Discussion around G4Tetrahedron

## Generating Random Point in Tetrahedron

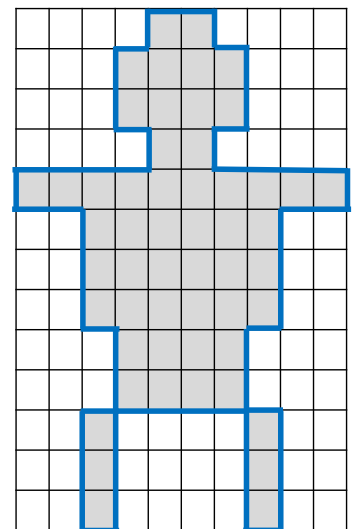
- Steps to generate a random point inside a tetrahedron defined by four vertices ( $v_0, v_1, v_2, v_3$ ):
  - Generation of three random values:  $s, t, u$ . The point corresponding to these values is inside of a cube with the sides equal to 1
  - The cube can be subdivided in six equal tetrahedra. Applying a transformation that places the point into the tetrahedron at the origin
  - Calculation of the position in the original tetrahedron:  $p = v_0(1 - s - t - u) + v_1s + v_2t + v_3u$



- Detailed explanation of the algorithm:  
[C.Rocchini and P.Cignoni, Generating Random Points in a Tetrahedron, Journal of Graphics Tools, Volume 5, Issue 4 \(2000\)](#)

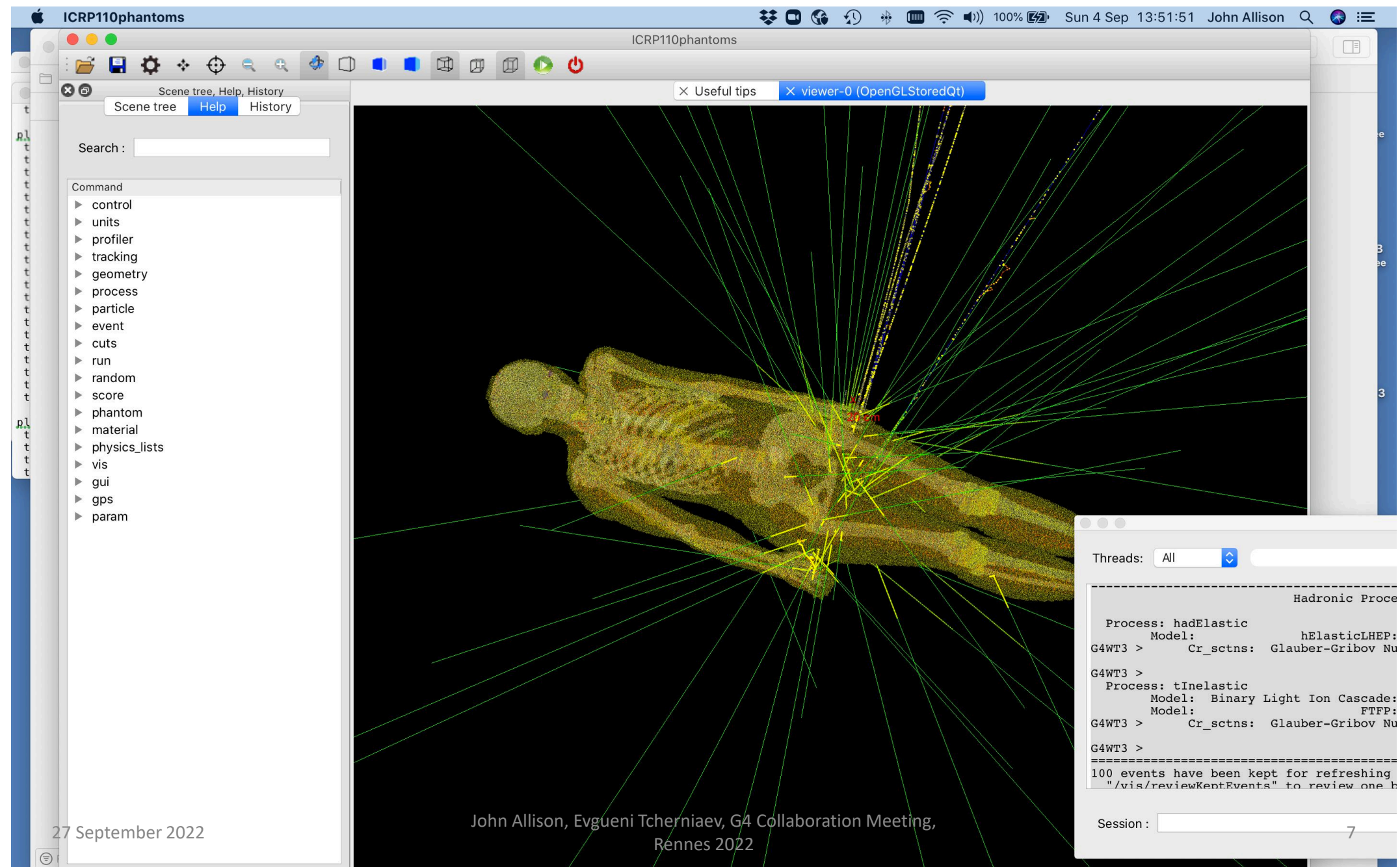
## Constructing Polyhedron from rectangular mesh

- G4PolyhedronBoxMesh(sx, sy, sz, positions)
  - sx, sy, sz – voxel dimensions
  - positions – array of voxels centers
- The construction of a polyhedron is fast, it takes  $O(N)$  time, and is done in two steps:
  - Step one: Construction of a 3D grid surrounding the mesh, the grid cells corresponding to the mesh voxels are marked (grey cells on the sketch image)
  - Step two: Only cell faces that do not have marked neighboring cells are included into the resulting polyhedron (blue edges on the sketch)



# John Allison - Special Mesh rendering

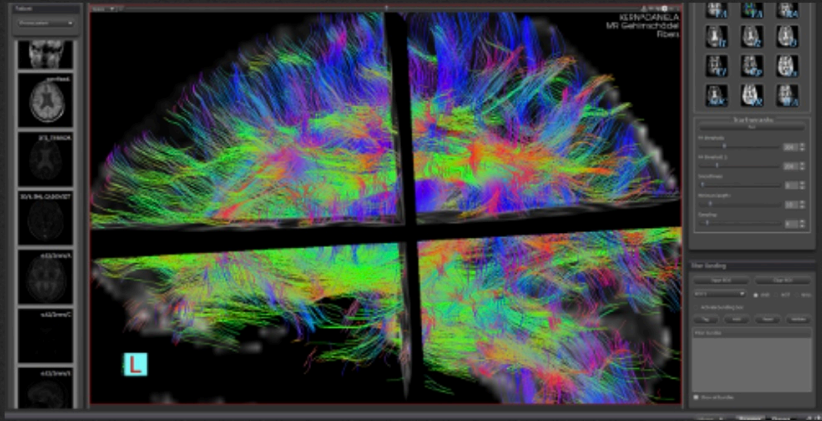
- Discussion around Point cloud generation



# John Alisson - Special Mesh rendering

- Discussion around how to visualize DICOM formats in Geant4
- Coffee discussion with an INRIA member who made MedInria

**“medInria:  
Whole New Glasses  
for your Medical Images”**



Stay posted with medInria's latest news by [subscribing to the announces mailing list](#).  
You can also follow us on Twitter to get the latest news: [@med\\_inria](#).

**Download v3.2.0**

Also check out the [doc](#) and the [forum](#). If you wish to see behind the scenes or start developing your own plugins, get the [source code](#)!

**medInria** is a **multi-platform** medical image processing and **visualization** software. It is **free and open-source**. Through an intuitive user interface, medInria offers from standard to **cutting-edge** processing functionalities for your medical images such as 2D/3D/4D image visualization, image **registration**, diffusion MR processing and **tractography**.

# John Alisson - Output streams

## Mapping to output streams

- It would be nice to eliminate the if statements

```
    if (verbosity >=
G4VisManager::confirmations)
{
    G4cout << some
output...
by something much simpler, e.g:
    G4visConfirm << some
output...
```
- `G4visConfirm` is an object that is aware of the vis verbosity and uses `G4cout`, `G4warn`, etc., as appropriate.
- The colours are suggestions for HTML-aware sessions, e.g., `G4UIQt`
- Genuine errors will use `G4Exception`

verbosity >=	Object	Replace if verbosity >= "column 1" by
Quiet	N/A	N/A
Startup	G4visStartup	G4cout
Errors	G4visError	G4warn
Warnings	G4visWarn	G4warn
Confirmations	G4visConfirm	G4cout
Parameters	G4visParam	G4cout
All	G4visAll	G4cout



# Guy Barrand - ToolsSG

Offscreen renderer thanks to gl2ps.

/vis/open TSG\_OFFSCREEN

*G4/vis/ToolsSG/Offscreen  
MinGW*

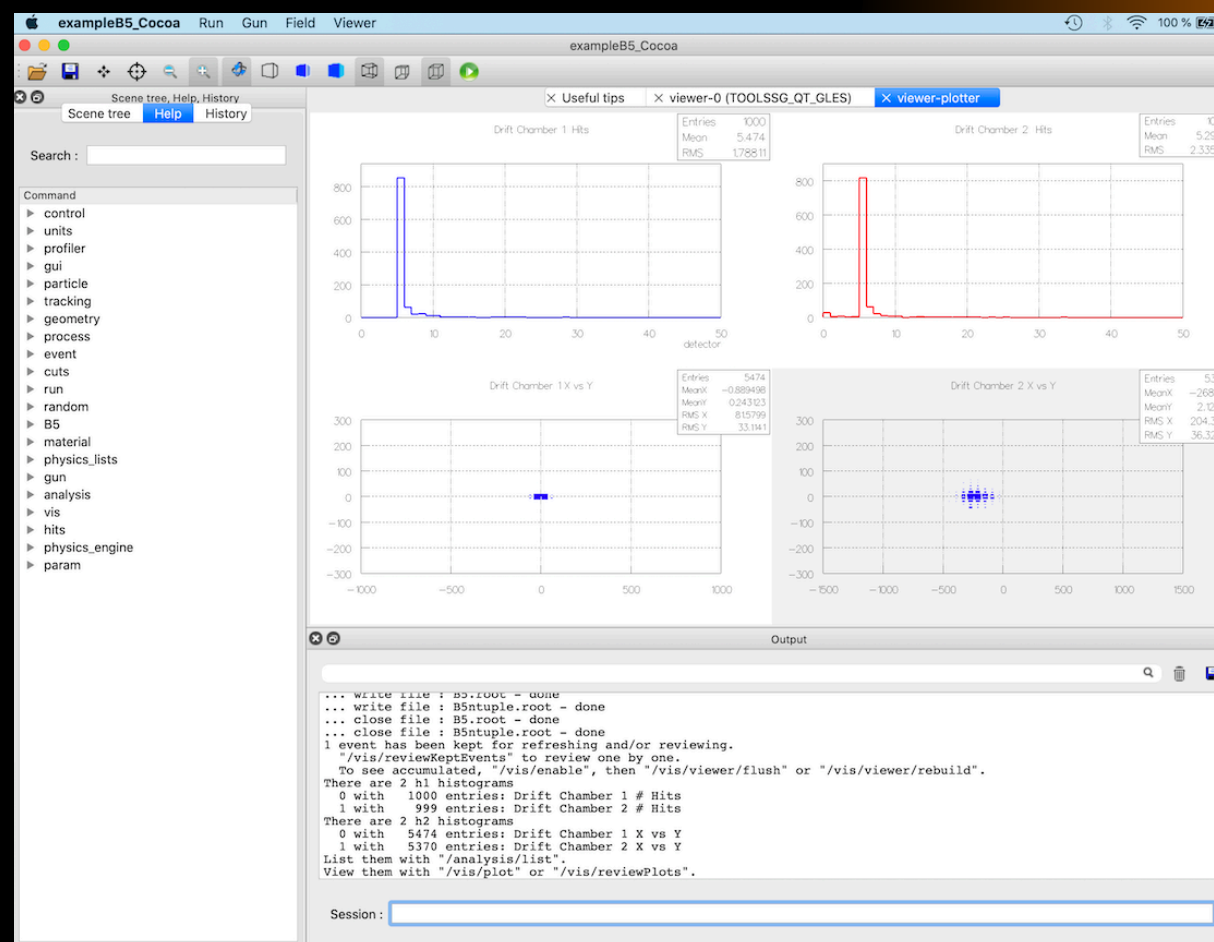
G4 Rennes 2022 workshop



# Guy Barrand - ToolsSG

## Plotting with ToolsSG

*vis/ToolsSG plotting (2)*



# Guy Barrand - MinGW

## *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...



**WSL** comes with Windows 10 and 11 and enables installation and operation of Linux (various distributions) without the need for a virtual machine or dual boot system.

Installation is easy: `wsl --install -d Ubuntu-20.04`

[set wsl version to 2 and install kernel update]

`bash` [opens a terminal window]

## My Experience (using with Windows 10):

- Building and installing Geant4 is identical to doing it on a native Linux system.
- Ubuntu is quite stripped down and **many** packages had to be installed to support Geant4.
- Existing documents for “Geant4 on WSL” were helpful but occasionally wrong.

## What about visualization? **Short answer:** everything works!

- WSL2 strategy: The application connects to an X server with the GLX extensions. The server is a Windows application and hence uses the Windows 3d graphics drivers.
- Best X server: GWSL. Didn't work for me: Xming (never); VcXsrv (eventually broke).
- Built and tested all the interactive drivers: OGLQt, Open Inventor, Vtk, and ToolsSG, without any problems, even running all of them at once. Very fast OpenGL rendering.
- **WSLg** (Windows 11 only) offers seamless graphics without the need for an X server.

## Is this important for Geant4?

- It could be an alternative way to support Geant4 on Windows. However, it would have to be operable within the existing workflows and supporting applications used by Windows Geant4 users.
- It allows Geant4 visualization access to high-performance 3d graphics hardware and drivers that are commonly used on gaming and professional Windows systems.

# Other topics

## **VTK driver**

- A MR is coming soon, we hope to have a VTK driver for the next release !

## **Qt**

- Qt6 is on the way, should be ready for next release !



