Geant4 Updates Visualization, GUI and Analysis

Joseph Perl - SLAC

For the Geant4 Visualization/Visualisation Working Group

15th Geant4 Space Users Workshop

Pasadena

December 2023

All slides are taken from presentations by others at the annual Geant4 Collaboration Workshops from 2019 to 2023.

G4 vis is an object-oriented multi-driver system. The core vis system converts the G4 scene (geometry, trajectories, etc.) into G4 graphics primitives. A driver renders the primitives and brings additional features, such as clipping, volume rendering, file export, etc.

You can choose your driver, even have different drivers in the same interactive session, and multiple views of the G4 scene.

Graphics is an ever-changing scene. It’s hard to keep up. We want to exploit modern features to keep Geant4 looking smart and performant. In particular we are looking for people to take on the work of continuing development of the Vtk driver (under Stewart’s supervision). We could do with someone with experience of Qt C++.

Come and join—even if part time—this fun and challenging working group.

New or improved and retired vis drivers

- New in Geant4 11.0 and further developed for 11.1 and 11.2
  - Qt3D (John Allison): limited functionality but nice
  - ToolsSG (TSG) (Guy Barrand): working nicely
    - Most features of the OpenGL drivers
    - Also supports plotting
    - Full-screen driver, TOOLSSG_OFFSCREEN—always built, default in batch mode
  - Open Inventor Qt (OIQt) (Fred Jones): Also very nice, requires users
    - Includes “bookmarking” and “navigation”
  - Vtk (Stewart Boogert, Laurie Nevay): Improved multi-featured version on the way
    - Interactive cutting and clipping
    - Export to GLTF (modern 3D object transfer protocol), interface to other packages
    - Export to web – scene can be rendered and manipulated in a webpage, “fantastic for manuals and documentation”
    - Off-screen rendering
- Retired (removed) in Geant4 11.1
  - HepRep/Wired (HepRepFile/HepRApp is retained)
  - VRML1 (VRML2 is retained)
  - The “network” drivers (those that communicate with their browser via BSD sockets”
    - VRML2 (VRMK2FILE is retained)
    - DAWN (DAWNFILE is retained)

What about Qt6 OpenGL driver ?

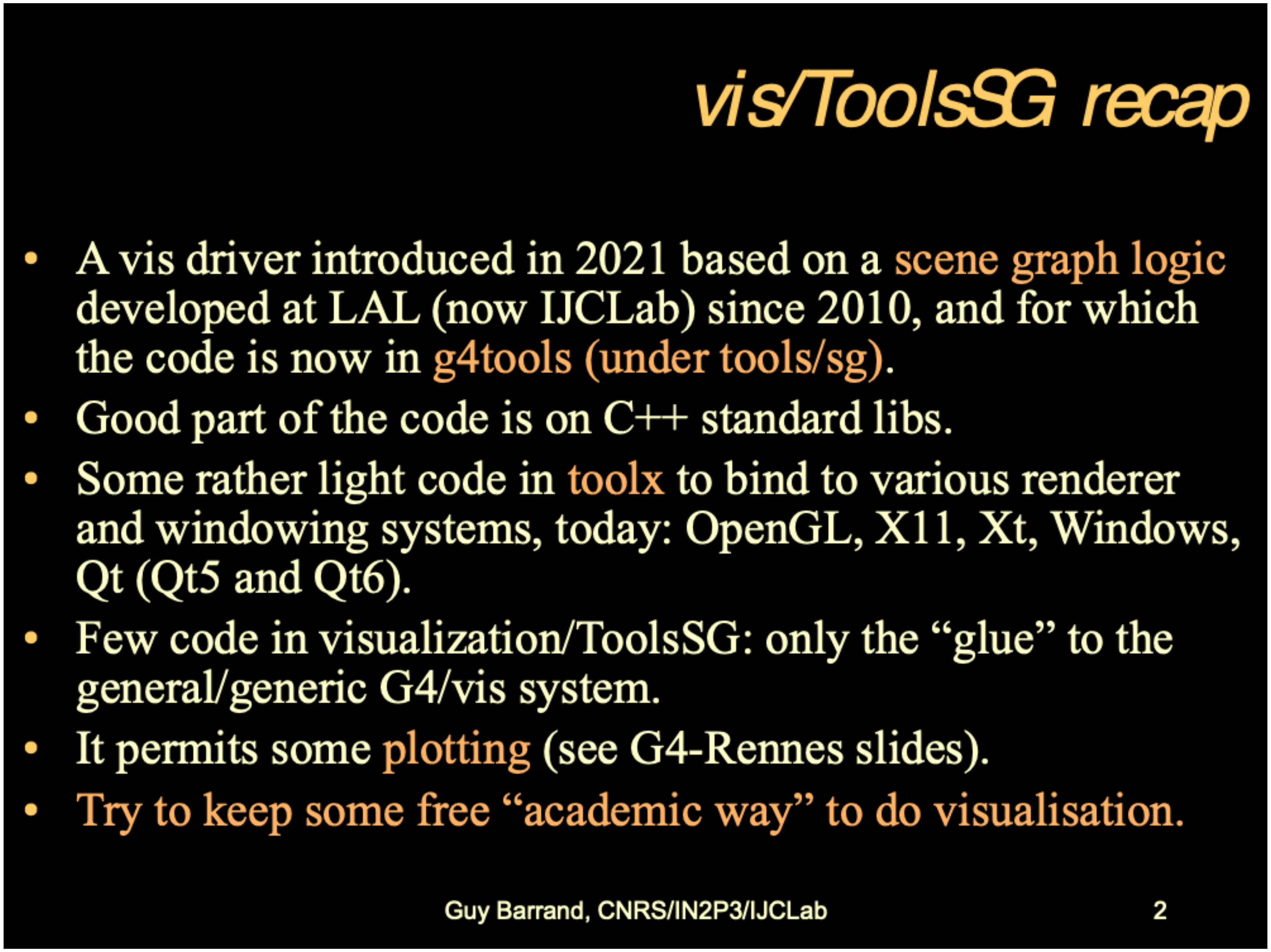
- Difficult to migrate and maintain OpenGL across ALL viewers
  - Qt6 OpenGL is object oriented whether Qt5 is C code
  - Viewers have to continue working even X rendering viewers (no Qt inside)
- Lot of migration work has been done
- More work to do about OpenGLContext and Multithreading

- At the moment we do not provide OGLQt with Qt.
- We asking ToolsSG to stand in for OpenGL

G4/vis/ToolsSG/Offscreen MinGW

G4 Rennes 2022 workshop

Guy Barrand, CNRS/IN2P3/IJCLab

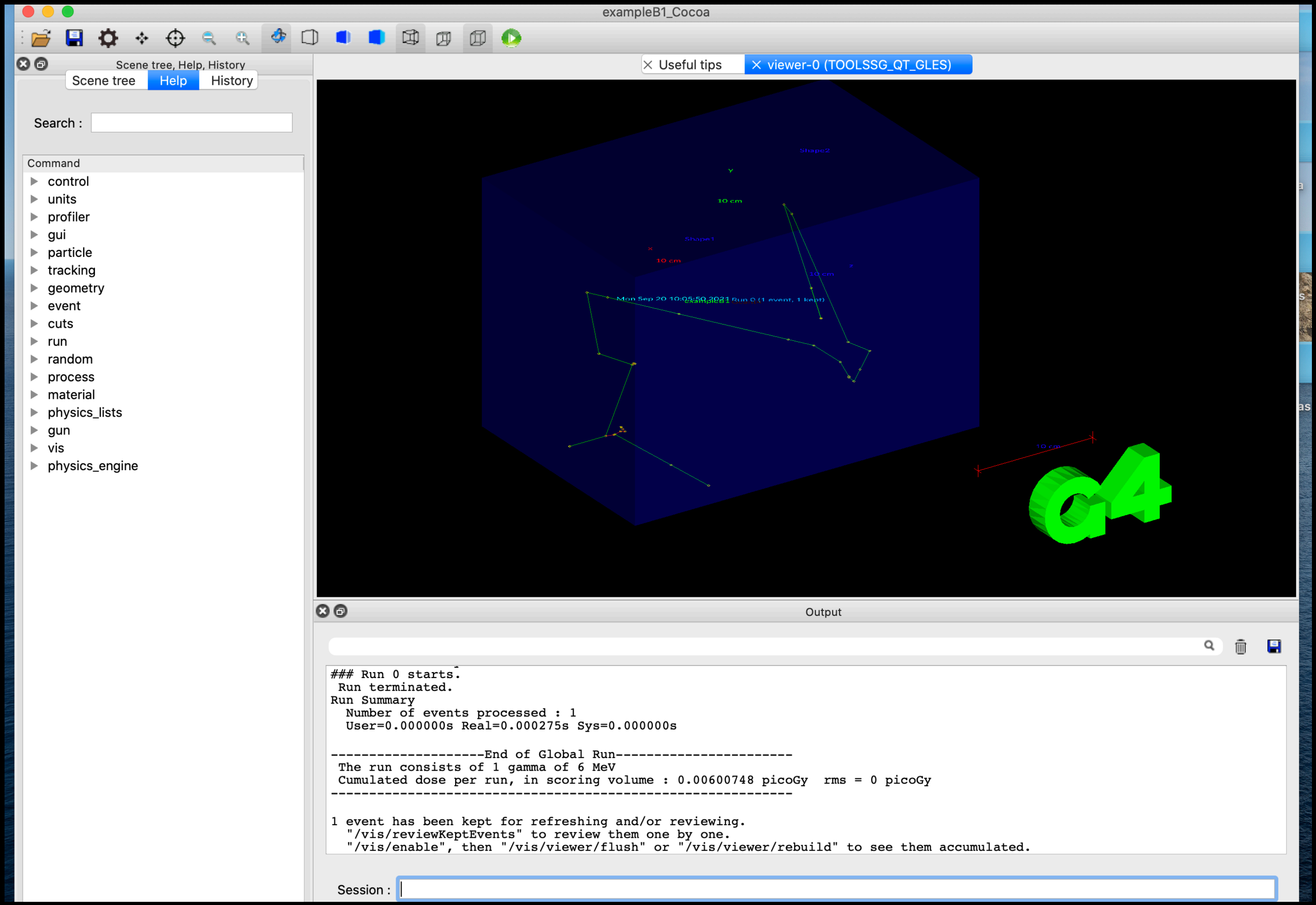


vis/ToolsSG recap

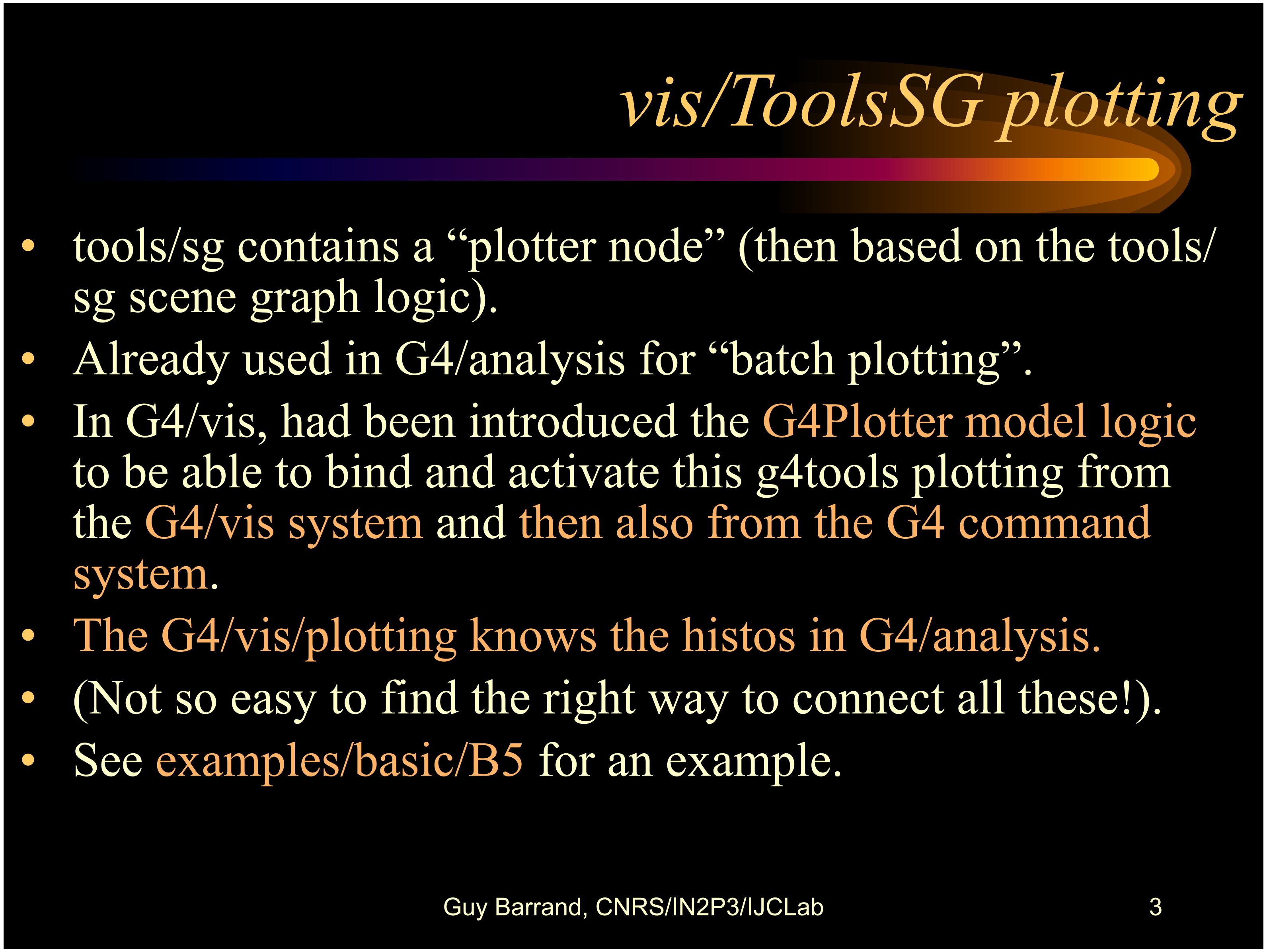
- A vis driver introduced in 2021 based on a scene graph logic developed at LAL (now IJCLab) since 2010, and for which the code is now in g4tools (under tools/sg).
- Good part of the code is on C++ standard libs.
- Some rather light code in toolx to bind to various renderer and windowing systems, today: OpenGL, X11, Xt, Windows, Qt (Qt5 and Qt6).
- Few code in visualization/ToolsSG: only the "glue" to the general/generic G4/vis system.
- It permits some plotting (see G4-Rennes slides).
- Try to keep some free "academic way" to do visualisation.

Guy Barrand, CNRS/IN2P3/IJCLab 2

ToolsSG/Qt/OpenGL driver on B1



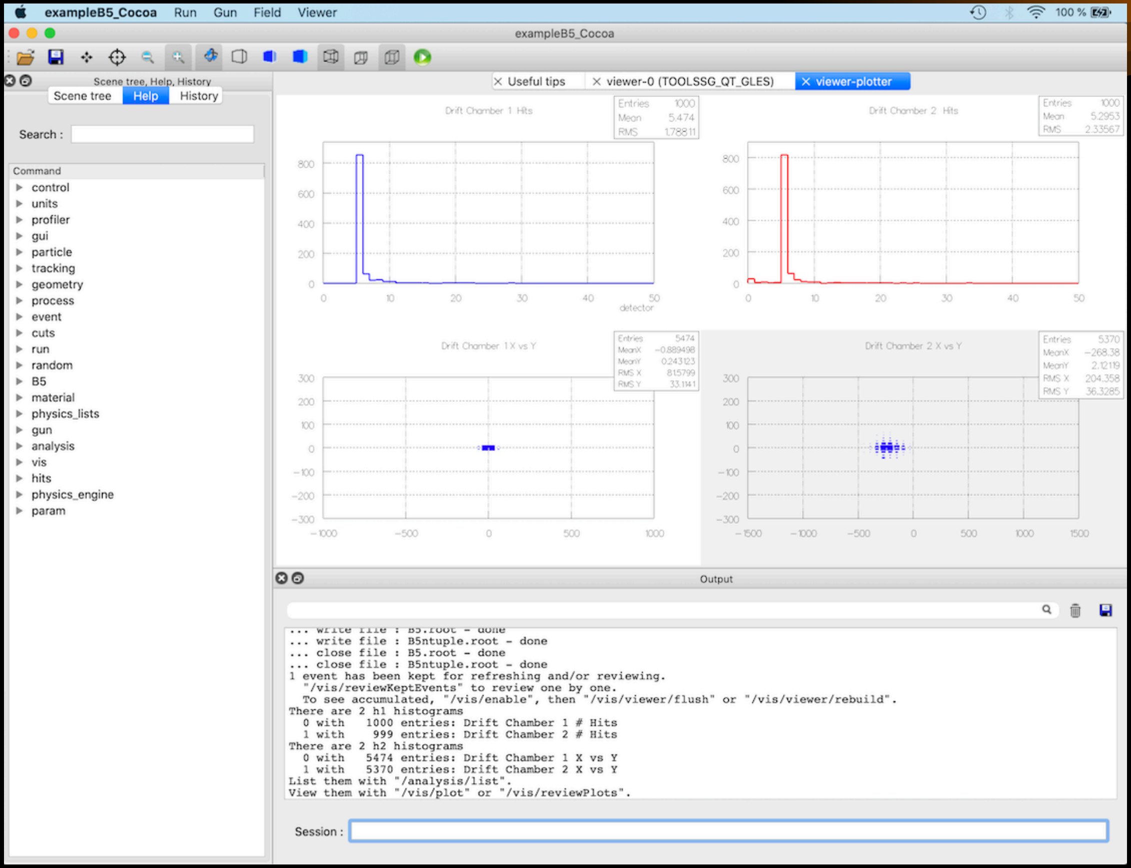
```
### Run 0 starts.
 Run terminated.
Run Summary
  Number of events processed : 1
  User=0.000000s Real=0.000275s Sys=0.000000s

--------------------End of Global Run-----------------------
 The run consists of 1 gamma of 6 MeV
 Cumulated dose per run, in scoring volume : 0.00600748 picoGy  rms = 0 picoGy
------------------------------------------------------------

1 event has been kept for refreshing and/or reviewing.
  "/vis/reviewKeptEvents" to review them one by one.
  "/vis/enable", then "/vis/viewer/flush" or "/vis/viewer/rebuild" to see them accumulated.
```



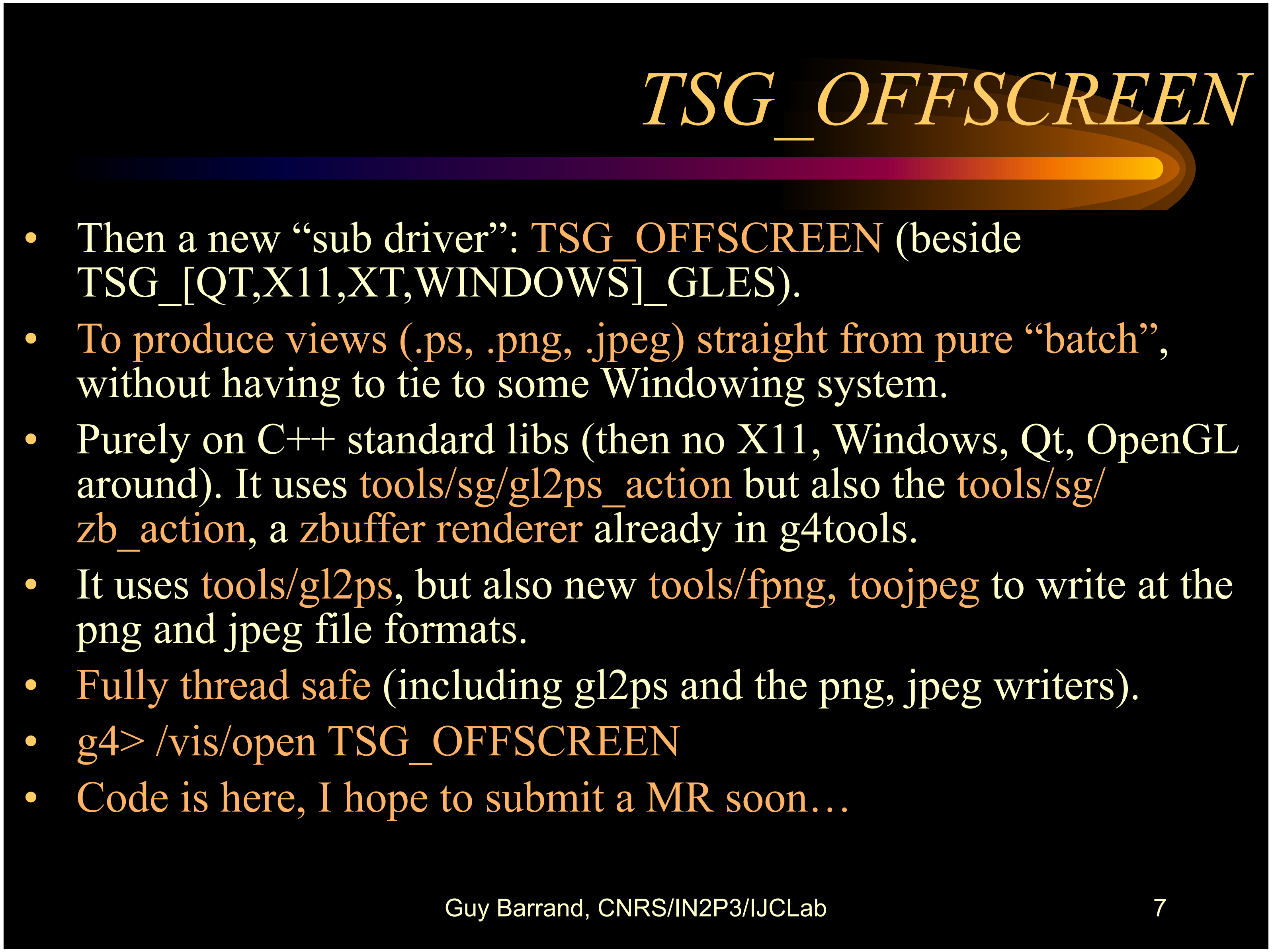
vis/ToolsSG plotting

- 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)



```
... write file : B5.root - done
... write file : B5ntuple.root - done
... close file : B5.root - done
... close file : B5ntuple.root - done
1 event has been kept for refreshing and/or reviewing.
  "/vis/reviewKeptEvents" to review one by one.
  To see accumulated, "/vis/enable", then "/vis/viewer/flush" or "/vis/viewer/rebuild".
There are 2 h1 histograms
  0 with   1000 entries: Drift Chamber 1 # Hits
  1 with    999 entries: Drift Chamber 2 # Hits
There are 2 h2 histograms
  0 with   5474 entries: Drift Chamber 1 X vs Y
  1 with   5370 entries: Drift Chamber 2 X vs Y
List them with "/analysis/list".
View them with "/vis/plot" or "/vis/reviewPlots".
```



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…

TRIUMF

Open Inventor Qt Viewer

Frederick Jones, TRIUMF

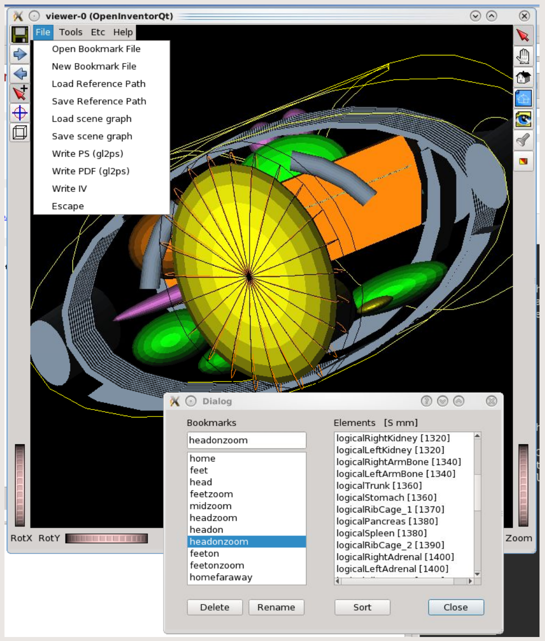
The Qt-based viewer (OIQt) was completed and released in Geant4 11.0.

How to use:

- Install Qt5 (most people will already have this to support the Qt UI).
- Install Coin3d libraries Coin 4.0 and SoQt 1.6 (or newer versions) using your package manager OR by downloading the sources for Coin and SoQt from https://coin3d.github.io/ and following the build instructions for Linux or MacOS.
- Build with `cmake –DGEANT4_USE_OPENINVENTOR_QT ...`
- To open the viewer: `/vis/open OI` either in terminal or UIQt session
- MacOS note: this vis driver is intended to use the native OpenGL on MacOS. There is no requirement for XQuartz or any X11 or Mesa libraries.

Some unique capabilities

- Full 3d manipulation and interactivity via mouse/trackpad, viewer buttons, menus, and a bookmark/navigation panel.
- Bookmarking: similarly to a web browser, any 3d view can be "bookmarked" and revisited later with a single mouse-click on an editable list which is persistent between sessions. Useful for geometry development/debugging and interactive slide shows.
- Navigation along a reference path defined by a trajectory or by a 3d polyline, with precise camera movements and rotations.
- Others, including fly-through animation, seek function, mouse-over readouts of volumes and trajectories, operates within the Qt UI or can be detached from it as a resizable window.



Is anybody using it? Waiting for user comments and questions!

Why VTK?

- Best in class scientific rendering
- KitWare world leading opensource scientific computing company (cf. Cmake)
- Trusted in medical applications, high end engineering, non-HEP scientific visualisation
- Large user community
- Stable and **documented** API in C++
- 1000s of C++ examples to base visualization code on
- Started in December 1993 (pre-dates Geant4)
- **Works exceptionally well for Geant4!**

Development supported by

Los Alamos NATIONAL LABORATORY EST. 1943

NATIONAL LIBRARY OF MEDICINE

ASC™

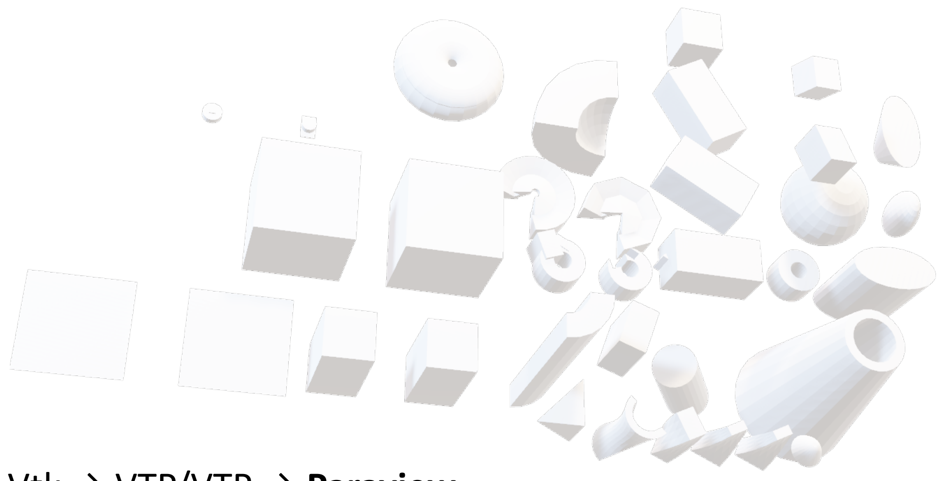
Sandia National Laboratories

ARL U.S. Army Research Laboratory

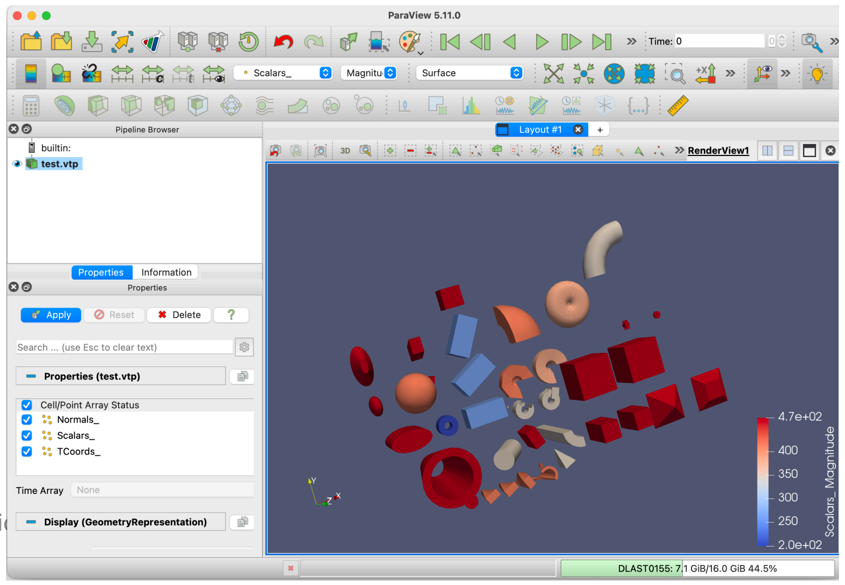
Vtk exports

- Export to almost all possible 3D file formats
  - VTU/VTP (vtk.js web model)
  - OBJ (convert to GLTF/USD web mode)
  - VRML
  - PLY
- Export screen grabs
  - PS
  - JPEG
  - TIFF
  - PNG
  - BMP
- Offscreen rendering and 3D conversion etc

Vtk → OBJ/GLTF → **Powerpoint**



Vtk → VTP/VTP → **Paraview**



Vtk possibilities (HEP outreach)

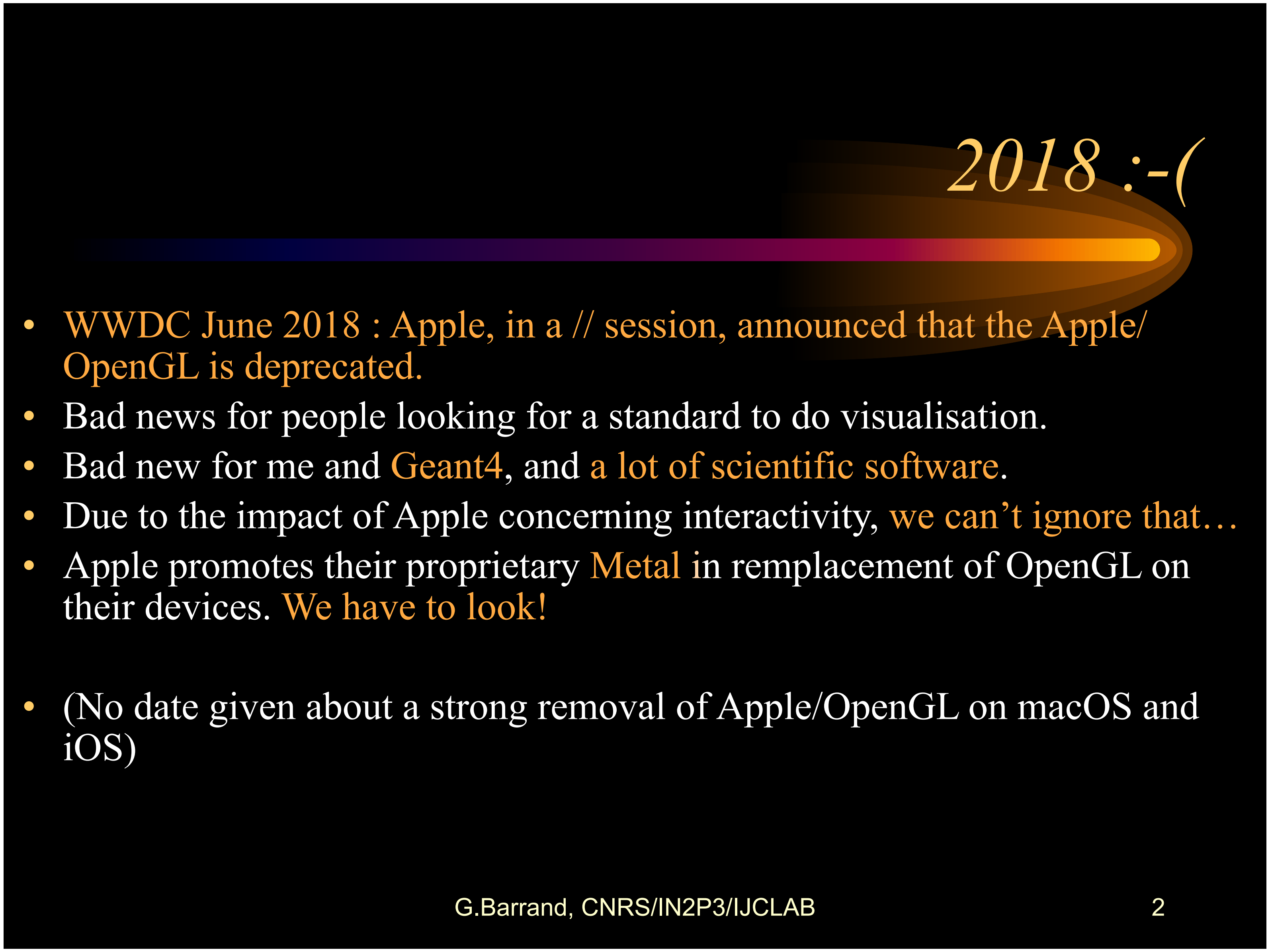
- Complex lighting
- Shadows
- Physically based rendering
- Ray tracing Ospray
- OpenXR simple to enable (VR/AR for free)
- Vtk for iOS and Android and Javascript so mobile version possible.

- **Each of these (non-critical improvements) each will take ~40 hours work to implement**
- Boogert happy to support keen post-doc or PhD student wanted to develop an HEP outreach project

Apple/Metal at the Orsay forge

Geant4 collaboration meeting Sep 2020

G.Barrand, CNRS/IN2P3/IJCLAB 1



2018 :-(

- WWDC June 2018 : Apple, in a // session, announced that the Apple/OpenGL is deprecated.
- Bad news for people looking for a standard to do visualisation.
- Bad new for me and Geant4, and a lot of scientific software.
- Due to the impact of Apple concerning interactivity, we can’t ignore that…
- Apple promotes their proprietary Metal in remplacement of OpenGL on their devices. We have to look!

- (No date given about a strong removal of Apple/OpenGL on macOS and iOS)

The X11 way on Macs

- Anyway now X11, OpenMotif, an X11 server, OpenGL/Mesa, are available in a consistent way (for exemple through MacPorts) on macOS, then we can always run on Macs this way without any GUI and graphics Apple software.
- (Avoid to mix with XQuartz libs !).
- (Would Qt/X11 with Mesa/GL be running on these ?)

- (In fact the same is true on Windows by using CYGWIN).

VISUALISATION DEVELOPMENTS IN GEANT4 10.5 AND FOR 10.6

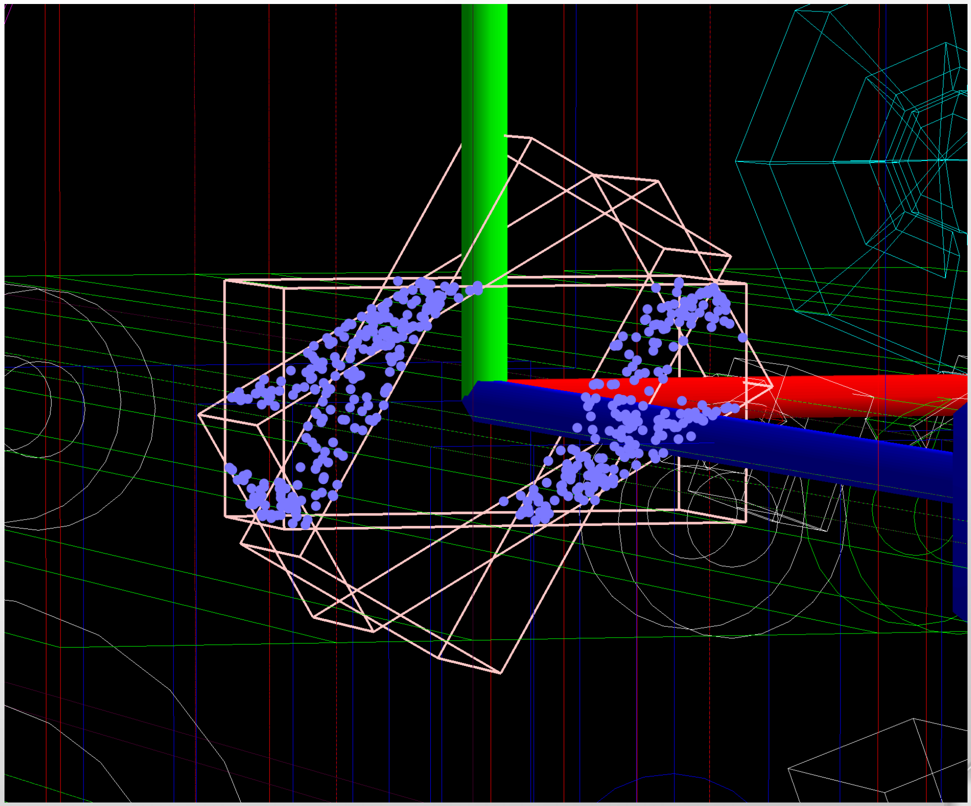
JOHN ALLISON

GEANT4 COLLABORATION MEETING JEFFERSON LAB

23-27 SEPTEMBER 2019

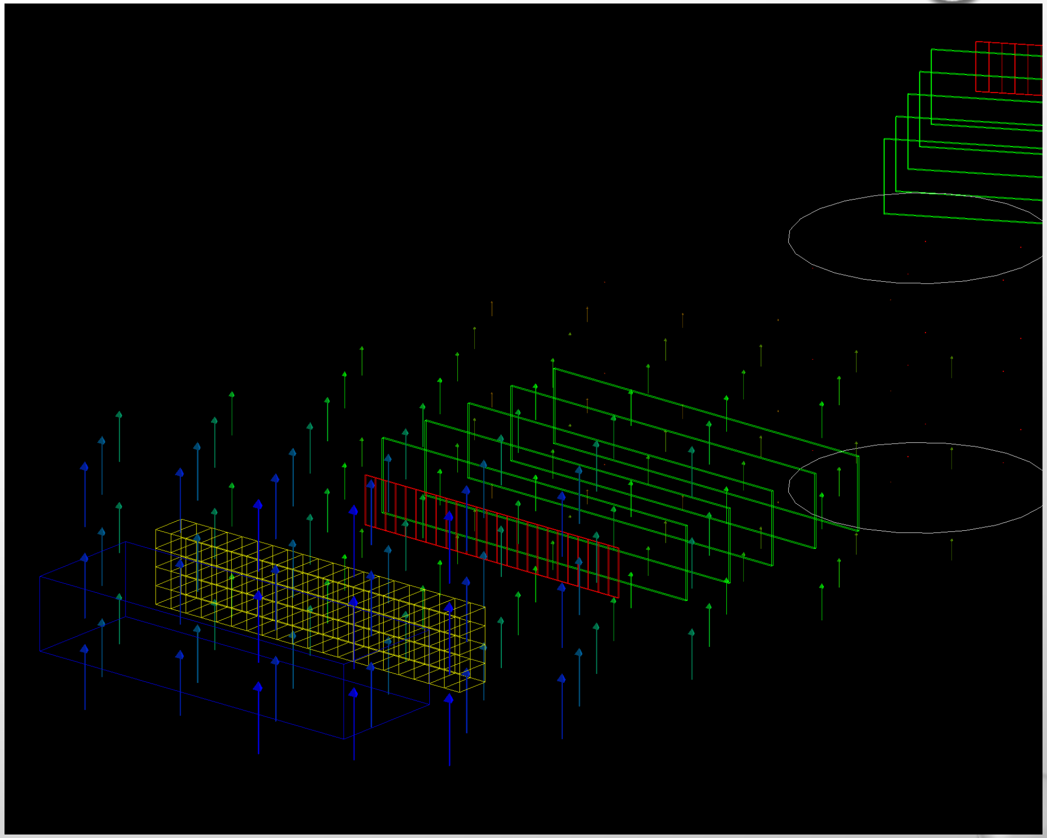
VISUALISATION OF OVERLAPS

- `/vis/drawLogicalVolume`
  - Uses `/vis/scene/add/logicalVolume` and opens a new scene.
  - See guidance.
  - By default prints and draws overlaps.



VISUALISATION OF FIELDS

- /vis/scene/add/magneticField
- /vis/scene/add/electricField (MIKE KELSEY)
- /vis/set/extentForField AND /vis/set/volumeForField
  - With these commands you can limit the extent over which the field is drawn. This would help, for example, if drawing over the whole scene produced so many arrows or lines that it clutters the scene.
  - See guidance for further explanation.

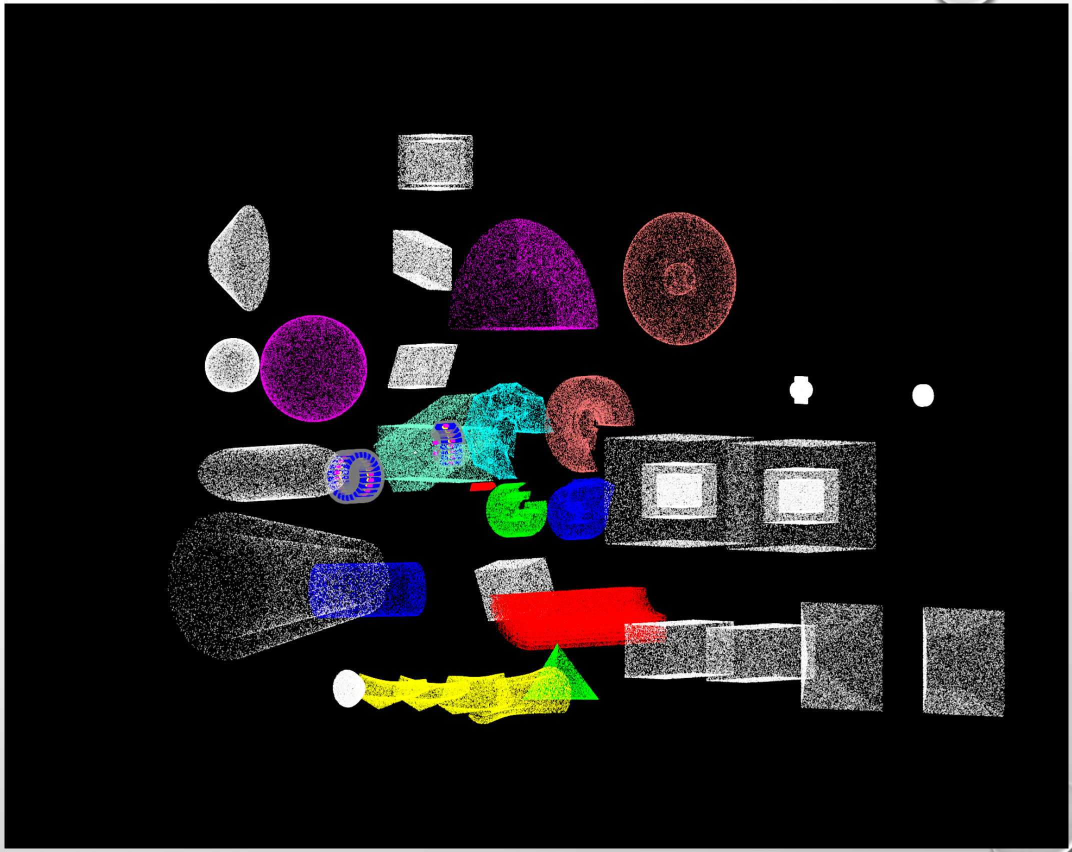


CENTRING

- /vis/viewer/centreOn AND /vis/viewer/centreAndZoomInOn
  - This allows one to centre the view (and zoom in) on a volume.
  - Reset with /vis/viewer/reset.
  - See movie—next slide.

CLOUD DRAWING STYLE

- /VIS/VIEWER/SET/STYLE CLOUD
  - Cloud drawing uses solid->GetPointOnSurface, i.e., it uses kernel algorithms and by-passes polyhedral representations.
  - The solid is represented by a polymarker of dots.
  - The default number of points is 1000.
    - This can be changed with `/vis/viewer/set/numberOfCloudPoints`
  - If polyhedral representation fails, for example during Boolean processing, drawing falls back to cloud.
  - Can be CPU demanding.



Special Mesh Rendering

John Allison and Evgueni Tcherniaev

What is a “G4Mesh”?

- In many applications, especially medical applications, the material world is represented in part by a `G4PVParameterised` (including `G4VNestedParameterisation`).
- Each element has a different location (and size?) and may be programmed to have a different material and colour.
- On request
  ```
  /vis/viewer/set/specialMeshRendering
  ```
  the vis manager asks `G4PhysicalVolumeModel` to look out for candidates. If a volume’s descendents (up to 3 deep) is a `G4PVParameterised`, the volume is declared a “container” and wrapped in a special class, `G4Mesh`, and passed to the scene handler for ”special rendering”.
- `G4Mesh` anticipates 5 types (see inset) but only 3—rectangular(2 types) and tetrahedron—are programmed at present.
- The user may ask for specific meshes:
  ```
  /vis/viewer/set/specialMeshVolumes <container-name>
  ```
- There are two options:
  ```
  /vis/viewer/set/specialMeshRenderingOption [dots|surfaces]
  ```
- The above commands must come before
  ```
  /vis/drawVolume
  ```

```
class G4Mesh {
 public:
  enum MeshType {
    invalid
    , rectangle
    , nested3DRectangular
    , cylinder
    , sphere
    , tetrahedron
  };
```

Special mesh rendering

- When a mesh is asked for and is found, instead of descending the geometry tree, `G4PhysicalVolumeModel` passes the whole mesh to the vis driver.
  - The driver may (optionally) implement AddCompound(const G4Mesh&)
  - Otherwise the base class default is invoked, drawing just the "container".
- The following drivers invoke "standard special mesh rendering":
  - OpenGL, ToolsSG, Qt3D and OpenInventor

```
void G4OpenGLSceneHandler::AddCompound(const G4Mesh& mesh) {
  StandardSpecialMeshRendering(mesh);
}
```

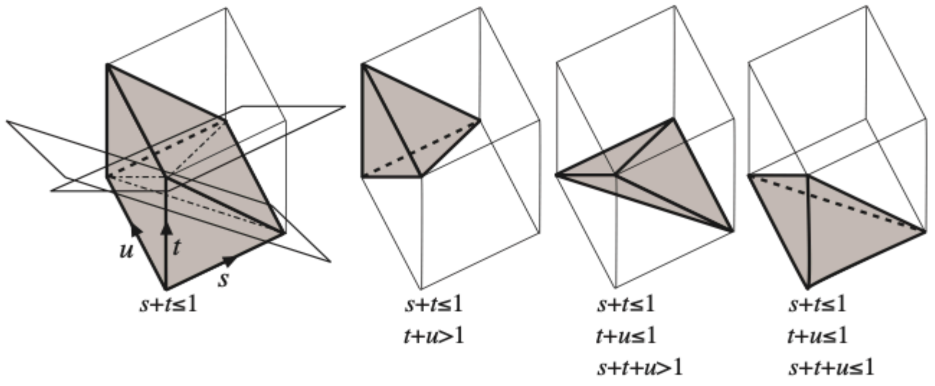
Standard special mesh rendering

- 4 programmed possibilities:
  - `Draw[3DRect|Tet]MeshAs[Dots|Surfaces]`
- Each involves a descent into the geometry sub-tree of the mesh, picking up the coordinates of the individual elements of the mesh, with their material and colour.
  - For "dots", each element gets a random point within it and a `std::multimap` is filled by material (with its colour).
    - Exploits fast rendering of points (`G4Polymarker::dots`) in OpenGL.
  - For "surfaces", each element is accumulated and used to construct a `G4Polyhedron` with `G4PolyhedronBoxMesh` or `G4PolyhedronTetMesh` as appropriate.
    - This is where the magic happens. Shared surfaces are removed by a very clever fast algorithm (Evgueni Tcherniaev) leaving a tractable `G4Polyhedron` of the outer faces.

```
void G4VSceneHandler::StandardSpecialMeshRendering(const G4Mesh& mesh)
// Standard way of special mesh rendering.
// MySceneHandler::AddCompound(const G4Mesh& mesh) may use this if
// appropriate or implement its own special mesh rendereing.
{
  G4bool implemented = false;
  switch (mesh.GetMeshType()) {
    case G4Mesh::rectangle: [[fallthrough]];
    case G4Mesh::nested3DRectangular:
      switch (fpViewer->GetViewParameters().GetSpecialMeshRenderingOption()) {
        case G4ViewParameters::meshAsDots:
          Draw3DRectMeshAsDots(mesh);  // Rectangular 3-deep mesh as dots
          implemented = true;
          break;
        case G4ViewParameters::meshAsSurfaces:
          Draw3DRectMeshAsSurfaces(mesh);  // Rectangular 3-deep mesh as surfaces
          implemented = true;
          break;
      }
      break;
    case G4Mesh::tetrahedron:
      switch (fpViewer->GetViewParameters().GetSpecialMeshRenderingOption()) {
        case G4ViewParameters::meshAsDots:
          DrawTetMeshAsDots(mesh);  // Tetrahedron mesh as dots
          implemented = true;
          break;
        case G4ViewParameters::meshAsSurfaces:
          DrawTetMeshAsSurfaces(mesh);  // Tetrahedron mesh as surfaces
          implemented = true;
          break;
      }
      break;
    case G4Mesh::cylinder: [[fallthrough]];
    case G4Mesh::sphere: [[fallthrough]];
    case G4Mesh::invalid: break;
  }
  if (!implemented) {
    G4VSceneHandler::AddCompound(mesh);  // Base class function - just print warning
  }
  return;
}
```

Generating Random Point in Tetrahedron

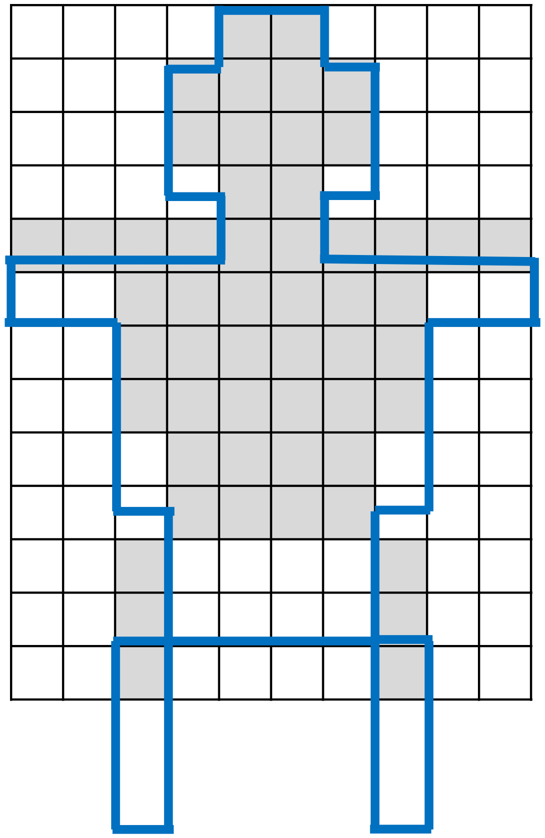
- Steps to generate a random point inside a tetrahedron defined by four vertices (v_0,v_1,v_2,v_3):
  - Generation of three random values: *s*, *t*, *u*. The point corresponding to these values is inside of a cube with the sides equal to 1
  - The cube can be subdivided in six equal tetrahedra. Appling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



- 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)]

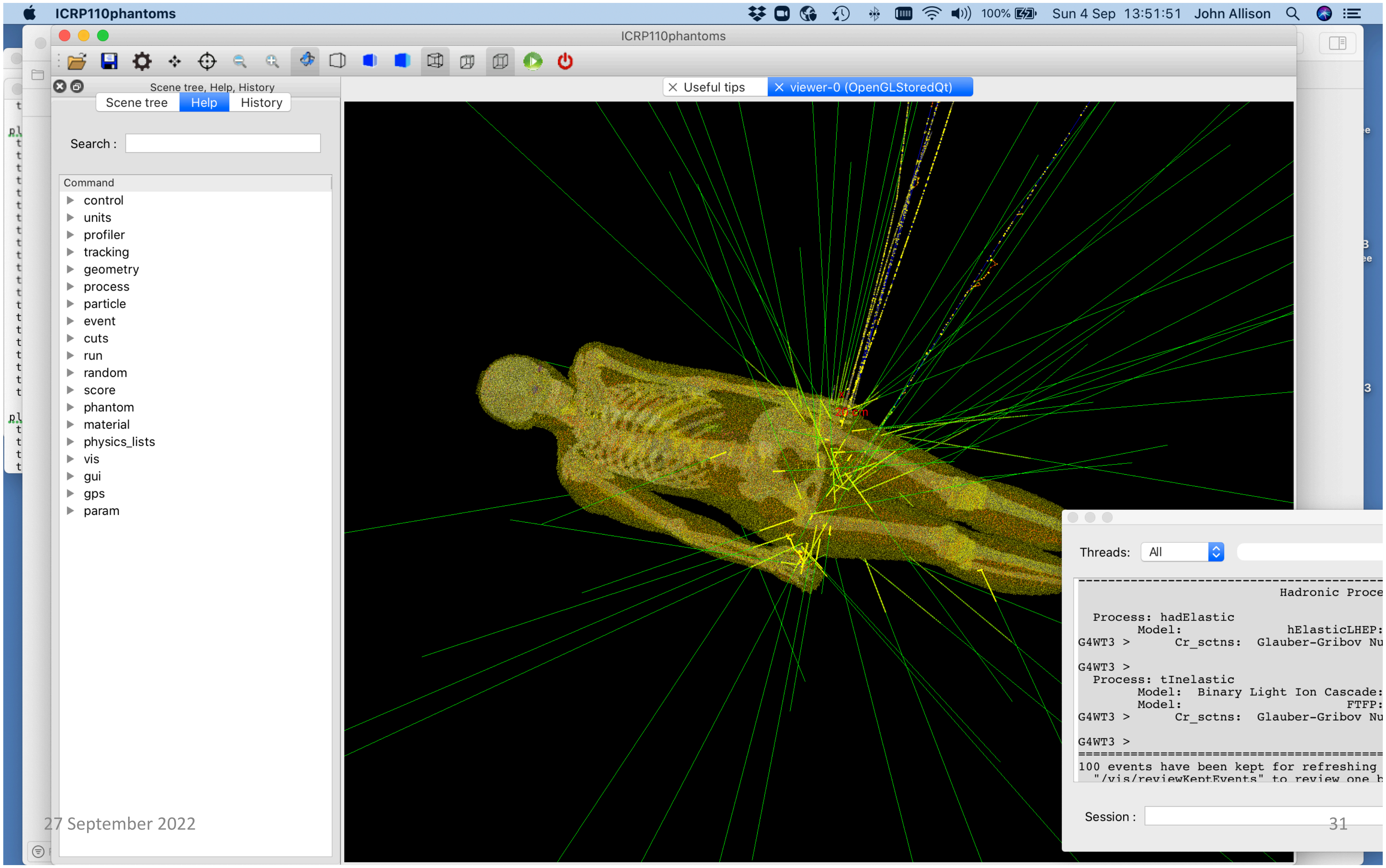
Constructing Polyhedron from rectangular mesh

- 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)



ICRP110 Performance

- ICRP110 is an existing Geant4 advanced example:
  - From ICRP, 2009. Adult Reference Computational Phantoms. ICRP Publication 110. Ann. ICRP 39 (2).
  - Contains a `G4VNestedParameterisation` of 299x137x348 (14,255,124) boxes:
    - 3,885,291 of which are “visible” representing 52 organs
    - one dot per box are sorted into 52 `G4Polymarker` objects of different materials (and colour)
  - These 3,885,291 dots are rendered at about 5 fps on MacBook Pro (Retina, Mid 2012), 2.7 GHz Quad-Core Intel Core i7, 16 GB



```
-----------------------------------------------
                                 Hadronic Proce
 
  Process: hadElastic
        Model:                 hElasticLHEP:
G4WT3 >       Cr_sctns:   Glauber-Gribov Nu

G4WT3 >
  Process: tInelastic
        Model:   Binary Light Ion Cascade:
        Model:                      FTFP:
G4WT3 >       Cr_sctns:   Glauber-Gribov Nu

G4WT3 >
===============================================
100 events have been kept for refreshing
  "/vis/reviewKeptEvents" to review one b
```

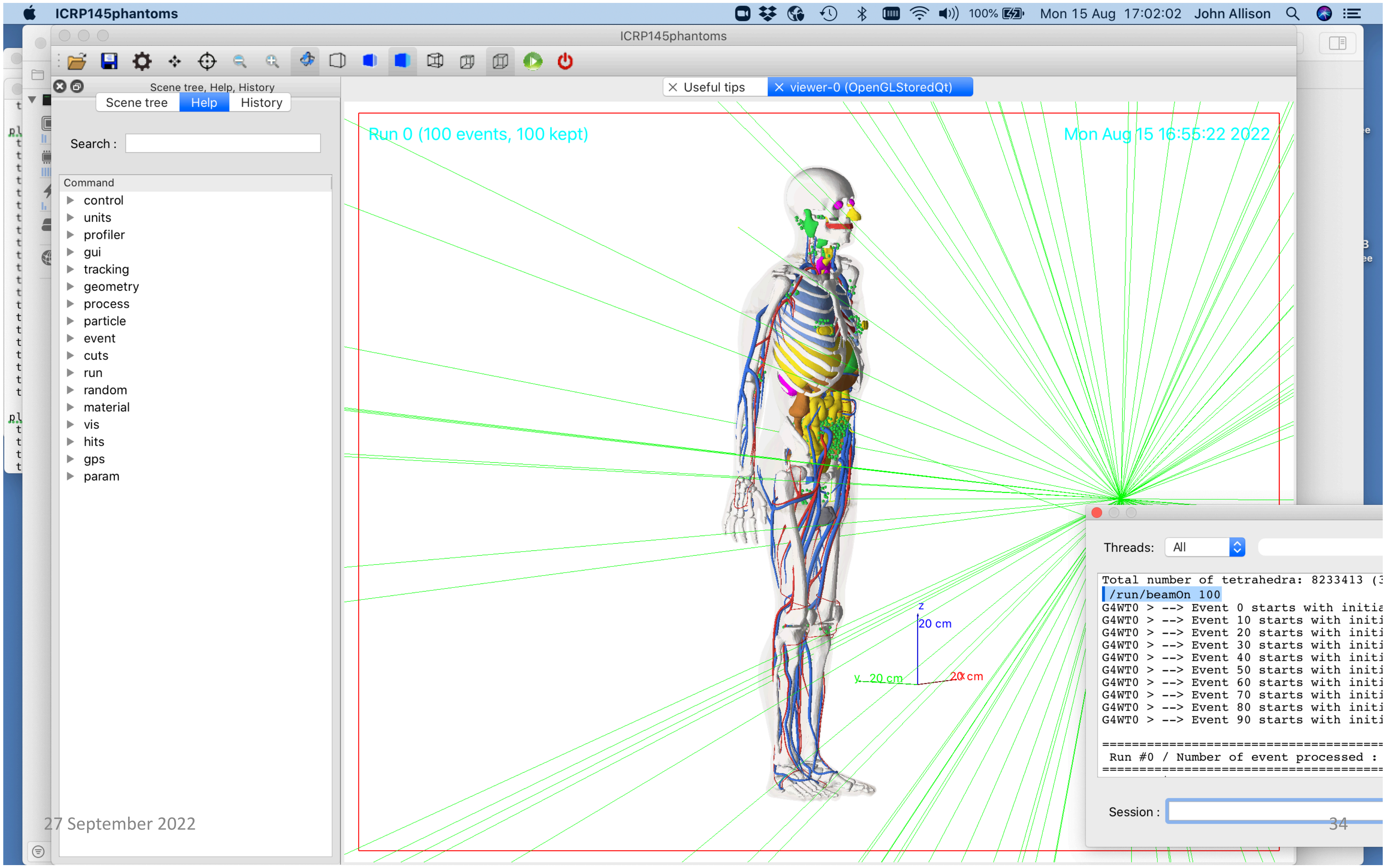
Constructing Polyhedron from tetrahedron mesh

- G4PolyhedronTetMesh(vertices)
  vertices – tetrahedra, four vertices per tetrahedron
- Elimination of internal (shared) faces is done in two steps using a technique similar to “hash map”. Objects are sorted into lists, the index of the list for an object is calculated by applying a hash function to this object
  - Step one: Identification of coincident vertices. Below is a C++ code to generate a hash value for a vertex:

    ```
    auto index = std::hash(v.x());
    index ^= std::hash(v.y());
    index ^= std::hash(v.z());
    index %= n_of_lists;
    ```
  - Step two: Identification of internal/external faces and selecting the external ones. The faces are sorted into lists using smallest index among the three vertex indices that define the face

ICRP145 Performance

- ICRP145 is a new advanced example, presented in the Examples Session:
  - From ICRP, 2020. Adult mesh-type reference computational phantoms. ICRP Publication 145. Ann. ICRP 49(3)
    C.H. Kim, Y.S. Yeom, N. Petoussi-Henss, M. Zankl, W.E. Bolch, C. Lee, C. Choi, T.T. Nguyen, K. Eckerman, H.S. Kim, M.C. Han, R. Qiu, B.S. Chung, H. Han, B. Shin
  - Contains a one-level `G4PVParameterised` of 8,233,413 `G4Tets` (32,933,652 faces):
    - The 8,233,413 `G4Tets` represent 187 organs
    - By eliminating internal shared faces, tetrahedra are converted to 187 `G4Polyhedron` objects by material (and colour) with only 4,807,770 faces (14% of original number of faces)
    - Timings:
      - Geometry construction: 20 s
      - Physics tables: 90 s
      - Closing geometry: 20 s
      - Creating graphical database: 20 s
      - Rendering: 2 fps on MacBook Pro (Retina, Mid 2012), 2.7 GHz Quad-Core Intel Core i7, 16 GB



```
Total number of tetrahedra: 8233413 (3
/run/beamOn 100
G4WT0 > --> Event 0 starts with initia
G4WT0 > --> Event 10 starts with initi
G4WT0 > --> Event 20 starts with initi
G4WT0 > --> Event 30 starts with initi
G4WT0 > --> Event 40 starts with initi
G4WT0 > --> Event 50 starts with initi
G4WT0 > --> Event 60 starts with initi
G4WT0 > --> Event 70 starts with initi
G4WT0 > --> Event 80 starts with initi
G4WT0 > --> Event 90 starts with initi

======================================
 Run #0 / Number of event processed :
======================================
```

“Twinkling”

- When you centre or zoom in on a volume:

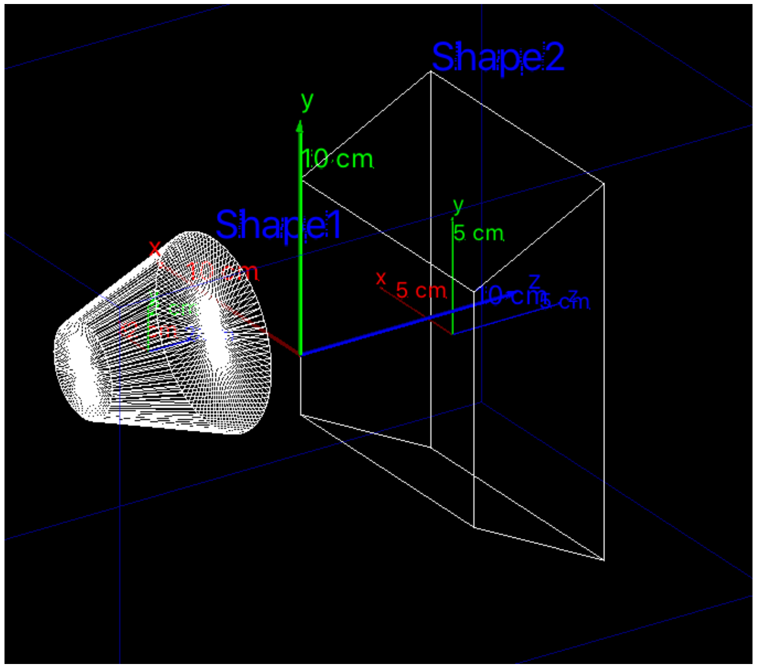
  ```
  /vis/viewer/centreOn and centreAndZoomInOn
  ```

  ```
  /vis/viewer/centreOn and centreAndZoomInOn
  ```

  the volume “twinkles” (varies on brightness) for a short time to attract your attention
- Restriction: the feature is switched off for complex geometries, i.e., those for which a rebuild of the graphics database takes more than $0.1\ s$. This because each frame with different brightness of the volume requires a rebuild.

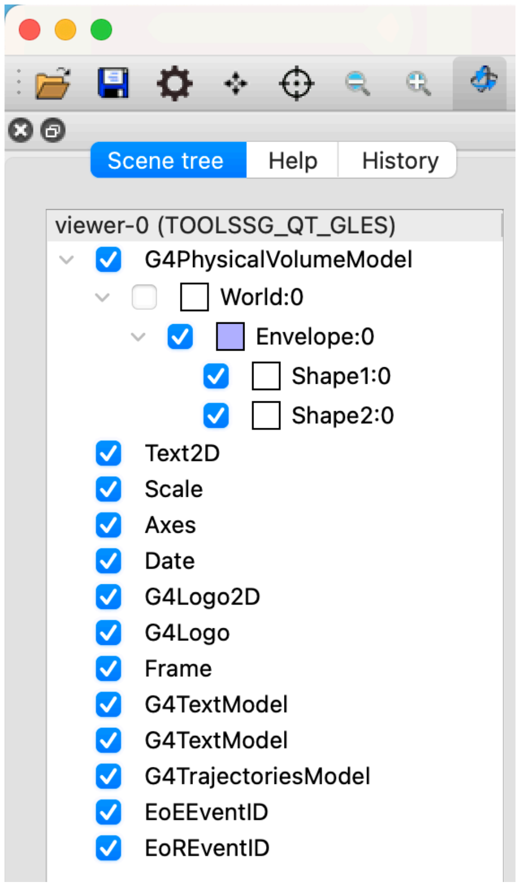
Local axes

- /vis/scene/add/localAxes
- /vis/touchable/localAxes



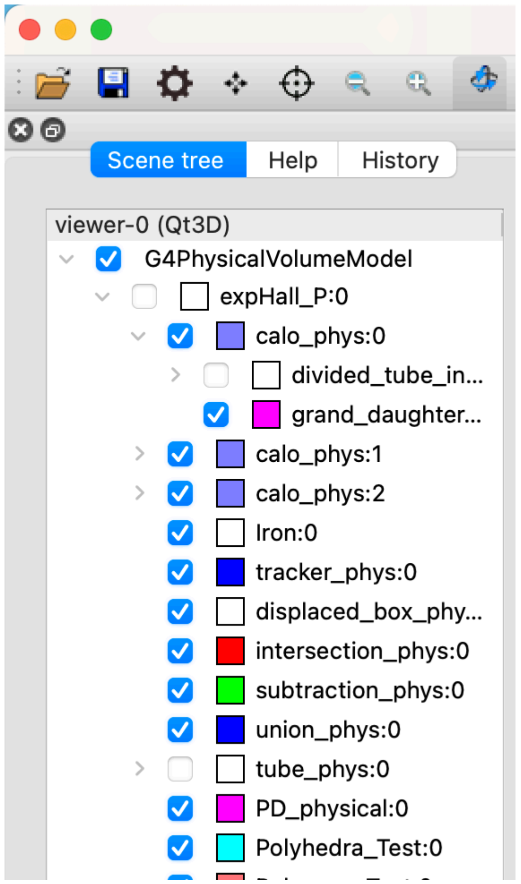
A new scene tree

- Available to *any* viewer when using the Qt GUI
  - Including non-Qt viewers, off-screen and file-writing viewers
  - A unique scene tree is held by each viewer and made available to the Qt GUI
  - The Qt GUI renders it to an interactive tree and notifies the interactions to the viewer
- Features
  - Hover to get dump of touchable
  - Click on blue check box to make invisible/visible
  - Click on chevron to hide/expose list of daughters in the scene tree
  - Double click on small square box to change colour
  - Right-click to get menu of actions



viewer-0 (TOOLSSG_QT_GLES)

- ✓ G4PhysicalVolumeModel
  - World:0
    - ✓ Envelope:0
      - ✓ Shape1:0
      - ✓ Shape2:0
- ✓ Text2D
- ✓ Scale
- ✓ Axes
- ✓ Date
- ✓ G4Logo2D
- ✓ G4Logo
- ✓ Frame
- ✓ G4TextModel
- ✓ G4TextModel
- ✓ G4TrajectoriesModel
- ✓ EoEEventID
- ✓ EoREventID

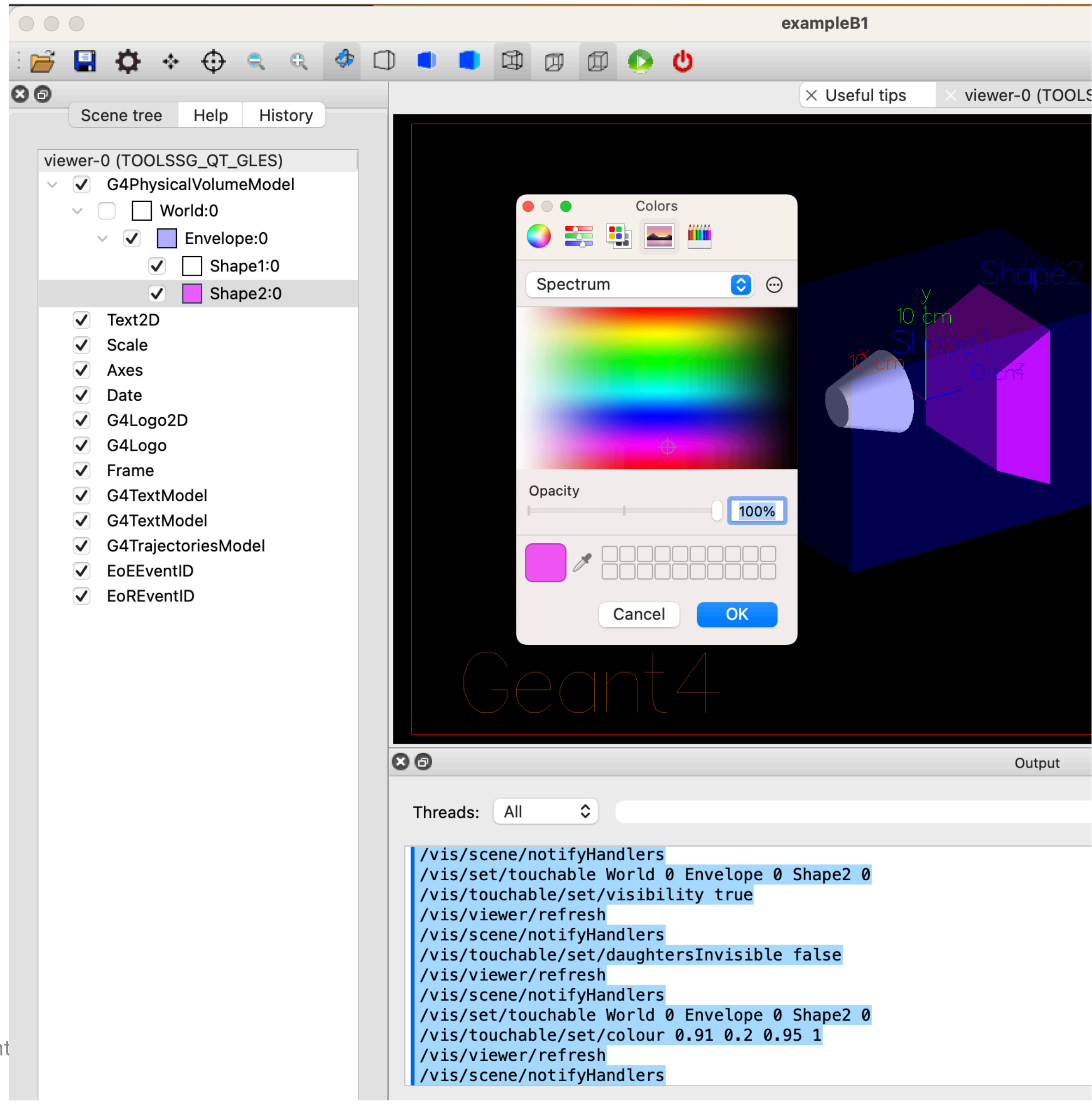


viewer-0 (Qt3D)

- ✓ G4PhysicalVolumeModel
  - expHall_P:0
    - ✓ calo_phys:0
      - divided_tube_in...
      - ✓ grand_daughter...
    - ✓ calo_phys:1
    - ✓ calo_phys:2
    - ✓ Iron:0
    - ✓ tracker_phys:0
    - ✓ displaced_box_phy...
    - ✓ intersection_phys:0
    - ✓ subtraction_phys:0
    - ✓ union_phys:0
    - tube_phys:0
    - ✓ PD_physical:0
    - ✓ Polyhedra_Test:0

New scene tree (contd)

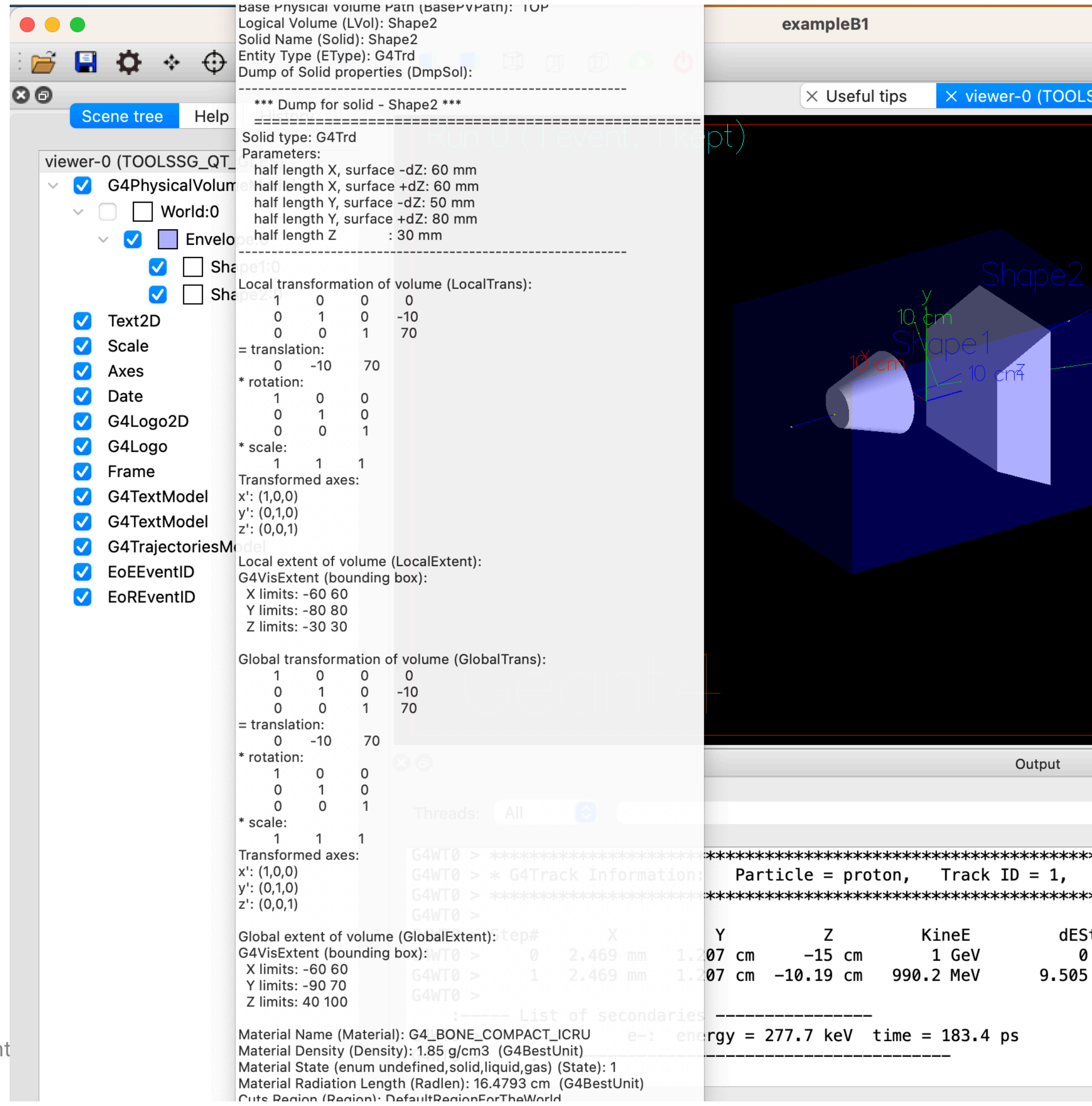
Double click on small square box to change colour



```
/vis/scene/notifyHandlers
/vis/set/touchable World 0 Envelope 0 Shape2 0
/vis/touchable/set/visibility true
/vis/viewer/refresh
/vis/scene/notifyHandlers
/vis/touchable/set/daughtersInvisible false
/vis/viewer/refresh
/vis/scene/notifyHandlers
/vis/set/touchable World 0 Envelope 0 Shape2 0
/vis/touchable/set/colour 0.91 0.2 0.95 1
/vis/viewer/refresh
/vis/scene/notifyHandlers
```

New scene tree

Simply hover to get dump of touchable

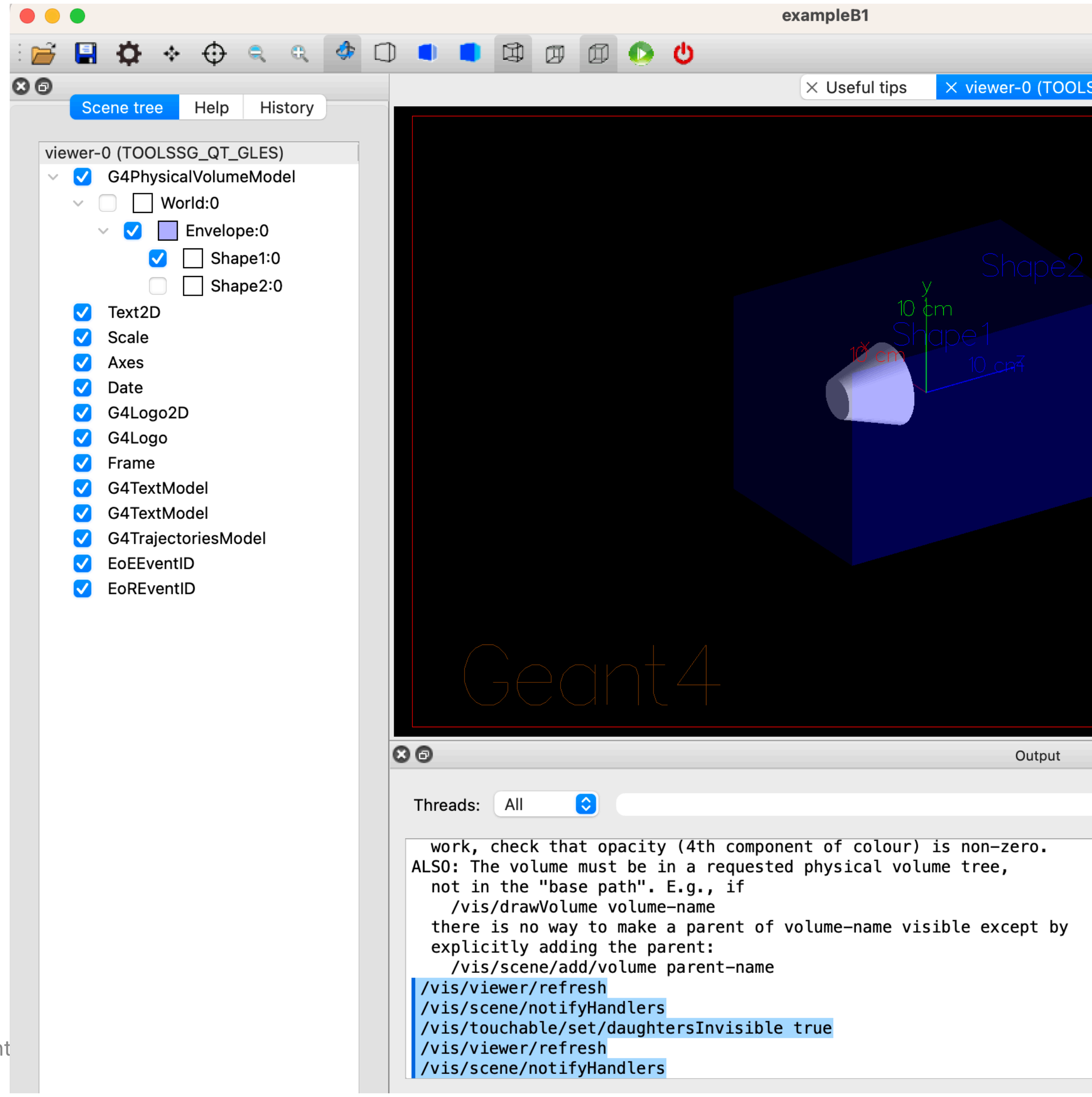


```
Base Physical Volume Path (BasePVPath):  TOP
Logical Volume (LVol): Shape2
Solid Name (Solid): Shape2
Entity Type (EType): G4Trd
Dump of Solid properties (DmpSol):
-----------------------------------------------------------
   *** Dump for solid - Shape2 ***
   ===================================================
 Solid type: G4Trd
 Parameters:
   half length X, surface -dZ: 60 mm
   half length X, surface +dZ: 60 mm
   half length Y, surface -dZ: 50 mm
   half length Y, surface +dZ: 80 mm
   half length Z            : 30 mm
-----------------------------------------------------------

Local transformation of volume (LocalTrans):
      1        0        0        0
      0        1        0      -10
      0        0        1       70
= translation:
      0      -10       70
* rotation:
      1        0        0
      0        1        0
      0        0        1
* scale:
      1        1        1
Transformed axes:
x': (1,0,0)
y': (0,1,0)
z': (0,0,1)

Local extent of volume (LocalExtent):
G4VisExtent (bounding box):
  X limits: -60 60
  Y limits: -80 80
  Z limits: -30 30

Global transformation of volume (GlobalTrans):
      1        0        0        0
      0        1        0      -10
      0        0        1       70
= translation:
      0      -10       70
* rotation:
      1        0        0
      0        1        0
      0        0        1
* scale:
      1        1        1
Transformed axes:
x': (1,0,0)
y': (0,1,0)
z': (0,0,1)

Global extent of volume (GlobalExtent):
G4VisExtent (bounding box):
  X limits: -60 60
  Y limits: -90 70
  Z limits: 40 100

Material Name (Material): G4_BONE_COMPACT_ICRU
Material Density (Density): 1.85 g/cm3  (G4BestUnit)
Material State (enum undefined,solid,liquid,gas) (State): 1
Material Radiation Length (Radlen): 16.4793 cm  (G4BestUnit)
Cuts Region (Region): DefaultRegionForTheWorld
```

```
**********************************************
  Particle = proton,   Track ID = 1,
**********************************************

   Y           Z          KineE        dES
07 cm      -15 cm        1 GeV           0
07 cm   -10.19 cm    990.2 MeV       9.505

 ---------------
 rgy = 277.7 keV   time = 183.4 ps
 ------------------------
```

New scene tree (contd)

Click on blue check box
to make invisible/visible



exampleB1

Scene tree | Help | History

Useful tips | viewer-0 (TOOLS

viewer-0 (TOOLSSG_QT_GLES)
- ✓ G4PhysicalVolumeModel
  - World:0
    - ✓ Envelope:0
      - ✓ Shape1:0
      - Shape2:0
- ✓ Text2D
- ✓ Scale
- ✓ Axes
- ✓ Date
- ✓ G4Logo2D
- ✓ G4Logo
- ✓ Frame
- ✓ G4TextModel
- ✓ G4TextModel
- ✓ G4TrajectoriesModel
- ✓ EoEEventID
- ✓ EoREventID

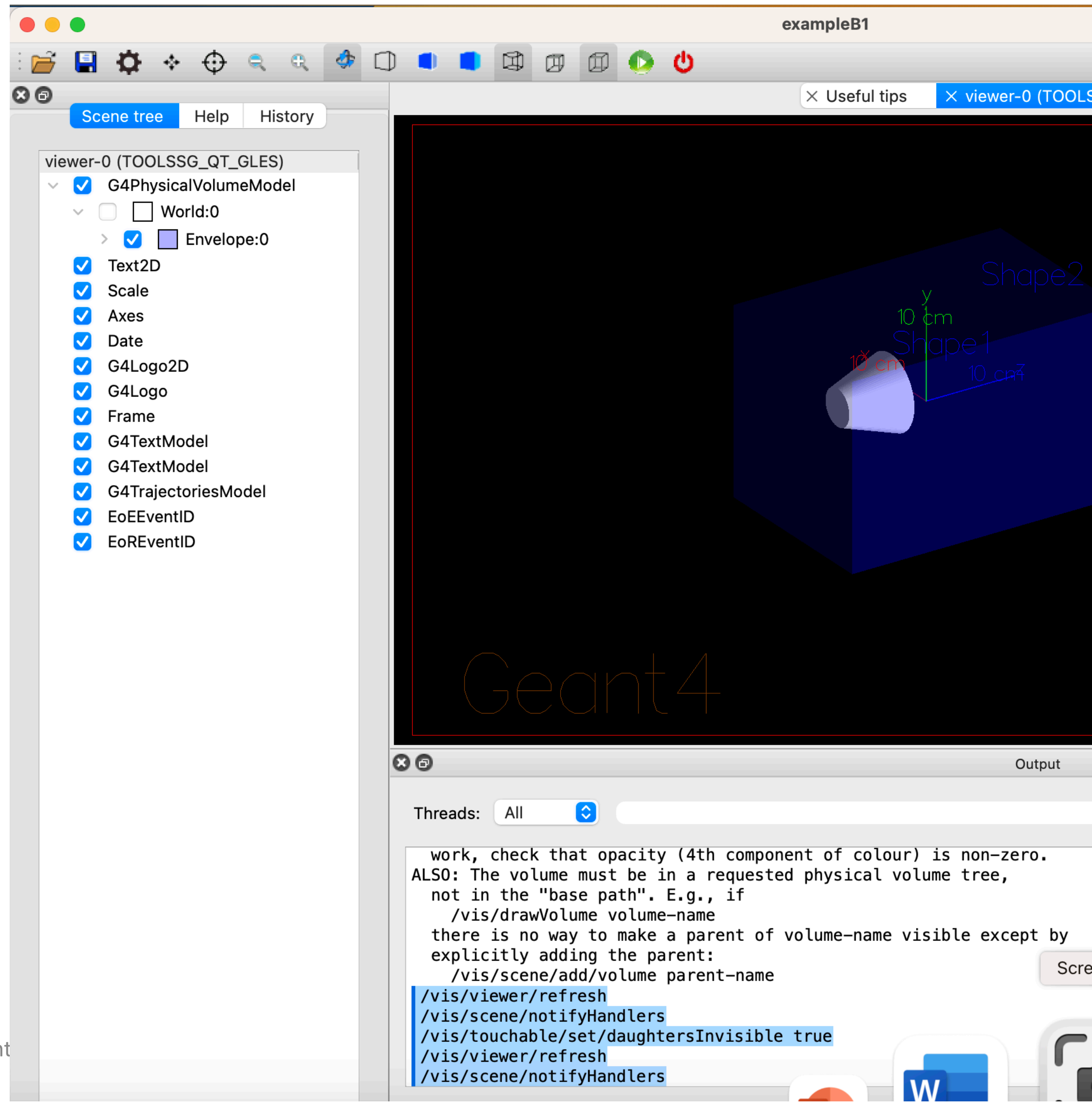
Output

Threads: All

```
  work, check that opacity (4th component of colour) is non-zero.
ALSO: The volume must be in a requested physical volume tree,
  not in the "base path". E.g., if
    /vis/drawVolume volume-name
  there is no way to make a parent of volume-name visible except by
  explicitly adding the parent:
    /vis/scene/add/volume parent-name
/vis/viewer/refresh
/vis/scene/notifyHandlers
/vis/touchable/set/daughtersInvisible true
/vis/viewer/refresh
/vis/scene/notifyHandlers
```

New scene tree (contd)

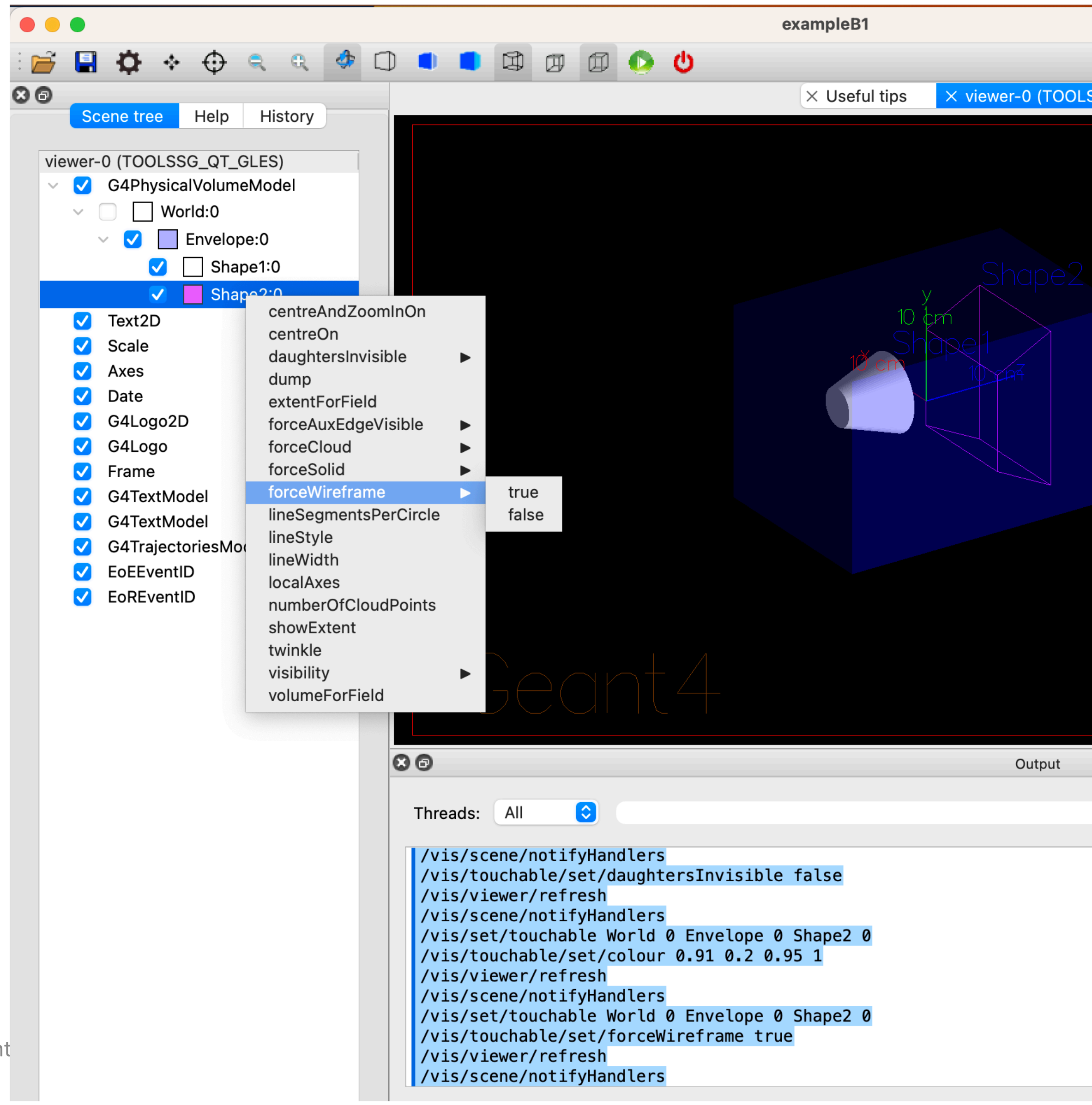
Click on chevron to hide/expose list of daughters in the scene tree



```
 work, check that opacity (4th component of colour) is non-zero.
ALSO: The volume must be in a requested physical volume tree,
 not in the "base path". E.g., if
   /vis/drawVolume volume-name
 there is no way to make a parent of volume-name visible except by
 explicitly adding the parent:
   /vis/scene/add/volume parent-name
/vis/viewer/refresh
/vis/scene/notifyHandlers
/vis/touchable/set/daughtersInvisible true
/vis/viewer/refresh
/vis/scene/notifyHandlers
```

New scene tree (contd)

Right-click to get menu of actions

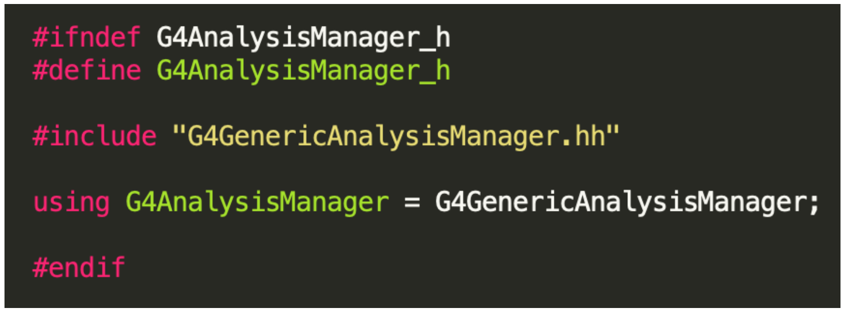


```
/vis/scene/notifyHandlers
/vis/touchable/set/daughtersInvisible false
/vis/viewer/refresh
/vis/scene/notifyHandlers
/vis/set/touchable World 0 Envelope 0 Shape2 0
/vis/touchable/set/colour 0.91 0.2 0.95 1
/vis/viewer/refresh
/vis/scene/notifyHandlers
/vis/set/touchable World 0 Envelope 0 Shape2 0
/vis/touchable/set/forceWireframe true
/vis/viewer/refresh
/vis/scene/notifyHandlers
```

Migration to Generic Analysis Manager in 11.0

- G4GenericAnalysisManager – first introduced in 10.7 and became the default in 11.0
  - It allows to select the output file type at run time, supports multiple files, including multiple output types for histograms and profiles
  - Still limitation of one output type for n-tuple management
- Users can still use the output specific manager, eg. G4RootAnalysisManager etc.
  - They need to include it directly
  - No deprecation warning is issued

- A new header file G4AnalysisManager.hh provided in 11.0 and used in almost all examples



```
#ifndef G4AnalysisManager_h
#define G4AnalysisManager_h

#include "G4GenericAnalysisManager.hh"

using G4AnalysisManager = G4GenericAnalysisManager;

#endif
```

- The output specific headers `g4csv.hh, g4hdf5.hh, g4root.hh, g4xml.hh and g4analysis.hh were removed`