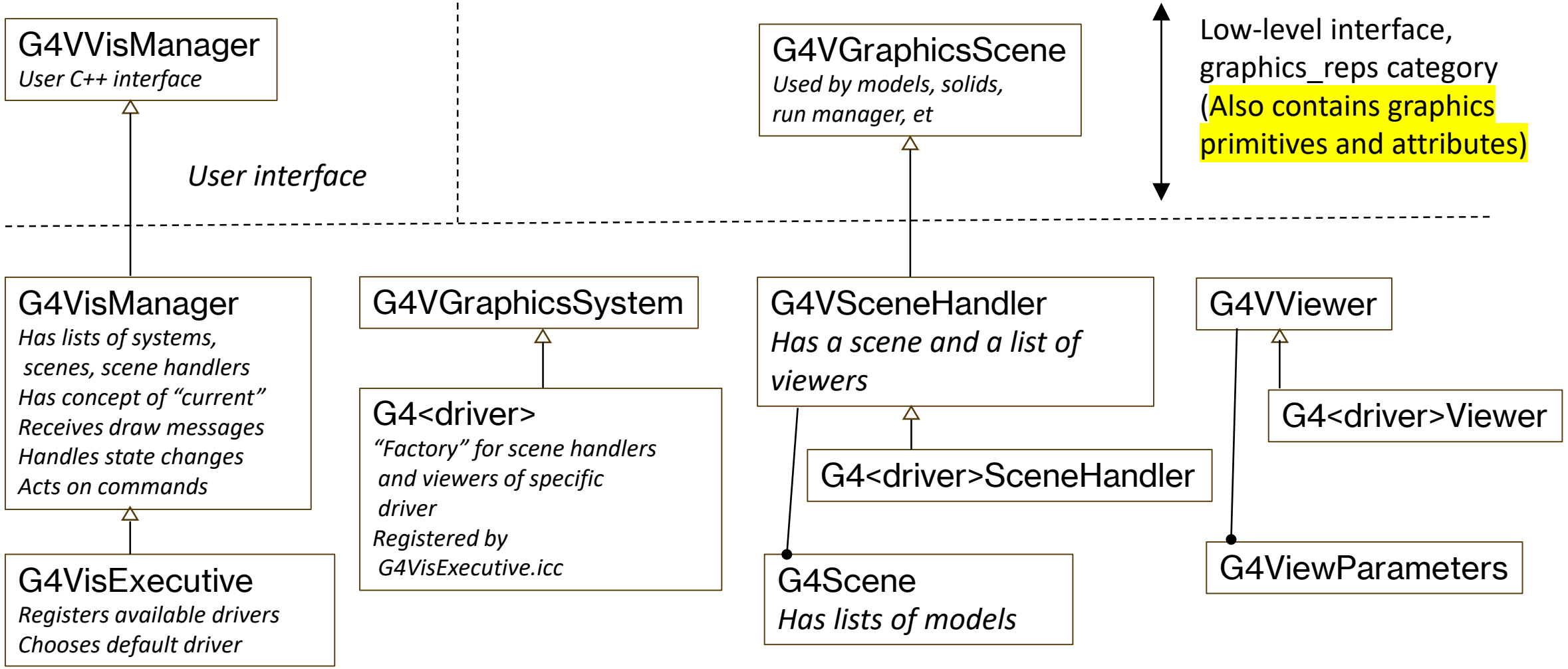
# Geant4 Visualisation: Concepts and Commands

Class Diagram
Scenes, models and all that

Commands

# Class diagram



# Scenes, models and all that

- A scene (**G4Scene**) consists of models representing Geant4 entities
  - The model converts the entity into graphics primitives
    - polyhedron, polyline, circle,...
  - 3 types: run-duration (e.g., detector), end-of-event, or end-of-run
  - At least one model must have an extent (bounding box)
    - The scene has an extent that is the accumulation of the extents of the models
    - The centre of the scene is the standard target point
  - There are some explicit models (see list)
    - **G4CallbackModel** is a generic way of writing a model
  - Models are added with **/vis/scene/add/** commands

```
G4ArrowModel.hh
G4AxesModel.hh
G4CallbackModel.hh
G4DigiModel.hh
G4ElectricFieldModel.hh
G4GPSModel.hh
G4HitsModel.hh
G4LogicalVolumeModel.hh
G4MagneticFieldModel.hh
G4NullModel.hh
G4PSHitsModel.hh
G4PhysicalVolumeModel.hh
G4PlotterModel.hh
G4TextModel.hh
G4TrajectoriesModel.hh
G4VFieldModel.hh
G4VModel.hh
G4VTrajectoryModel.hh
```

- The model knows how to describe itself to the scene handler
  - **G4PhysicalVolumeModel** sends representations (**G4Polyhedron**) of touchables (leaves of the geometry hierarchy tree) to chosen depth
  - The scene handler turns the primitives into graphics-system-specific information for the viewer
  - Smart scene handlers accumulate a graphical database for GPU rendering
- You can instantiate multiple scene handlers of the same or different type of driver, and multiple viewers of each driver
  - **/vis/viewer/list** to see, change current viewer with **/vis/viewer/select**
- All visualisable Geant4 objects have vis attributes (**G4VisAttributes**) (next slides)
  - E.g., **G4LogicalVolume**, **G4Polyhedron**, **G4Circle**,...
- Each viewer has its own view parameters (**G4ViewParamaters**) and standard view (next slides)

```
scene
  add
    arrow
    arrow2D
    axes
    date
    digis
    electricField
    eventID
    extent
    frame
    gps
    hits
    line
    line2D
    localAxes
    logicalVolume
    logo
    logo2D
    magneticField
    plotter
    psHits
    scale
    text
    text2D
    trajectories
    userAction
    volume
```

# Kernel visit

- Most drivers build a graphical database
  - Exploits GPU
  - Rotate, zoom, etc., is very fast
- Some operations require a rebuild
  - Each viewer decides…and then initiates a “kernel visit”
  - G4 objects are re-visited and the models re-interpret them
  - Force a rebuild with **/vis/viewer/rebuild**

  - Viewers without a graphical database initiate a kernel visit for every operation
    - E.g., OGLI, most file-writing drivers
    - It does not consume memory (it used to be a problem)
    - Rotation, etc., is slow

# Vis attributes

- Every visualisable Geant4 object, including **G4LogicalVolume**, has a vis attributes pointer (**G4VisAttributes***)

```
void SetVisibility          (G4bool = true);
void SetDaughtersInvisible  (G4bool = true);
void SetColour              (const G4Colour&);
void SetColor               (const G4Color&);
void SetColour              (G4double red, G4double green, G4double blue,
                             G4double alpha = 1.);
void SetColor               (G4double red, G4double green, G4double blue,
                             G4double alpha = 1.);
void SetLineStyle           (LineStyle);
void SetLineWidth           (G4double);
void SetForceWireframe      (G4bool = true);
void SetForceSolid          (G4bool = true);
void SetForceCloud          (G4bool = true);
void SetForceNumberOfCloudPoints (G4int nPoints);
// nPoints <= 0 means under control of viewer
void SetForceAuxEdgeVisible (G4bool = true);
void SetForceLineSegmentsPerCircle (G4int nSegments);
// Allows choice of circle approximation.  A circle of 360 degrees
// will be composed of nSegments line segments.  If your solid has
// curves of D degrees that you need to divide into N segments,
// specify nSegments = N * 360 / D.
void SetStartTime           (G4double);
void SetEndTime             (G4double);
void SetAttValues           (const std::vector<G4AttValue>*);
void SetAttDefs             (const std::map<G4String,G4AttDef>*);
```

# G4Atts (G4AttDef and G4AttValue)

- Trajectories and some models also have **G4Atts**
  - You can also add your on to **G4VisAttributes**
- **G4Atts** are string-based objects
  - **G4AttDef** defines ID, type, description,…
  - **G4AttValue** contains an ID and value as a string, interpreted by referring to the corresponding **G4AttDef**
  - The idea is that you have one set of **G4AttDef** objects, with multiple economical **G4AttValues**
    - An object typically has methods **GetAttDefs** and **GetAttValues**
      - The user must take care to delete the **G4AttValues**
  - Based on the **HepRep** concept by Joseph Perl – see http://heprep.freehep.org
- In principle can be used to add accessible values to any type of object *without introducing dependencies*
- They a "zero weight" – they do not add to the object itself but are created on demand in user space by (virtual) methods of the object

# View parameters and standard view

- Each viewer has its own view parameters (**G4ViewParameters**)
- The standard view is based on the scene's extent and the view parameters
- On instantiation, a viewer points to the standard target point and its field of view covers the scene
  - Thus, by default, your detector is always centre of the field of view
  - See **G4ViewParameters.hh** for an extensive description of these concepts
- View parameters are set only by commands – there is no user interface.
  - Except some that are overridden by vis attributes
    - E.g., **G4VisAttributes::SetForceWireframe()**, etc

# COMMAND-BASED VISUALIZATION

- Over 200 vis commands, with extensive guidance
  - Plus many more driver specific, modeling and filtering commands
  - The command guidance *is* our documentation
  - Use “ls”, “help” or the GUI help tree to see commands and their guidance
  - To echo or monitor: **/control/verbose 2**
- Common commands
- Scene editing
- G4Atts (**G4AttDef** and **G4AttValue**)
- Trajectory modeling and filtering
- Event keeping and reviewing
- Viewer control
- Saving, replaying and interpolating views
- Making movies
- Touchables
- Viewing meshes
- Geometry overlaps and other useful things
- Plotting
- Driver-specific commands

# Common commands

- To echo commands: **/control/verbose 2**
- Adjust verbosity: **/vis/verbose [verbosity]**
  - E.g., **/vis/verbose confirmations**
  - The default is **warnings**
- **/vis/disable**
  - Good to turn off trajectory storing as well: **/tracking/storeTrajectory 0**
- Useful commands: **/vis/list**, **/vis/viewer/list**, **/vis/scene/list**
- **/vis/open [<driver>]**
  - There is a default driver (OGL at present)
    - The default can be changed by environment or **~/.g4session**
  - **/vis/open** is actually **/vis/sceneHandler/create** + **/vis/viewer/create**
  - You can open multiple drivers of the same or different type, and multiple viewers of each driver
- **/vis/drawVolume [<physical-volume-name>]**
  - Default: world (top) physical volume
  - Actually **/vis/scene/create** + **/vis/scene/add/volume** + **/vis/sceneHandler/attach**
- **/vis/scene/add/trajectories [rich] [smooth]**
  - Adjust presentation with **/vis/modeling/…**, selection with **/vis/filtering/…**
- **/vis/scene/add/axes**, et (see **examples/basic/B1/vis.mac**)

Simple graded message scheme - digit or string (1st character defines):
0) quiet, // Nothing is printed.
1) startup, // Startup and endup messages are printed...
2) errors, // ...and errors...
3) warnings, // ...and warnings...
4) confirmations, // ...and confirming messages...
5) parameters, // ...and parameters of scenes and views...
6) all // ...and everything available.

# Scene editing

- Adding specific volumes (the extents are accumulated so the standard view changes each time you add a volume so that everything is within the field of view)
  - **/vis/scene/create**
  - **/vis/scene/add/volume A**
  - **/vis/scene/add/volume B**
  - …
  - **/vis/sceneHandler/attach**
- Adding trajectories (also hits or digis, if **Draw** methods implemented)
  - **/vis/scene/add/trajectories [rich] [smooth]**
- Event display and keeping behaviour (similarly for run)
  - **/vis/scene/endOfEventAction <refresh|accumulate> [number-of-events-to-be-kept]**
- Activate and de-activate models with **/vis/scene/activateModel**

# Changing a vis attribute with /vis/geometry

- E.g., **/vis/geometry/set/colour <logical-vol-name> <colour>**
- Changes the vis attributes in the *actual* logical volume
  - So changes for all views
  - And for all touchables that have that logical volume
  - Restore original vis attributes with **/vis/geometry/restore**

- Note:
  - To change the vis attributes of a specific touchable for a specific viewer, use **/vis/set/touchable** and **/vis/touchable/set**
    - These are really viewer commands – see **Viewer Control**

# Trajectory modeling and filtering

- /vis/scene/add/trajectories [rich] [smooth]
  - Default (no parameters): basic **G4Atts**
  - **smooth**: includes field interpolation points
    - A **G4Step** has only the start and end points, which can miss several turns of the spiral a charged particle experiences in a field. The default trajectory can look jagged – specify **smooth** to get a nice curved trajectory
  - **rich**: extended **G4Atts** (next slide)
    - Includes **smooth**
  - The **G4Atts** are printed at startup
  - They are available for modeling and filtering

```
G4TrajectoriesModel:
  Event ID (EventID): G4int
  Run ID (RunID): G4int
G4Trajectory:
  Charge (Ch): unit: e+ (G4double)
  Track ID (ID): G4int
  Initial kinetic energy (IKE):
    G4BestUnit (G4double)
  Initial momentum magnitude (IMag):
    G4BestUnit (G4double)
  Initial momentum (IMom):
    G4BestUnit (G4ThreeVector)
  No. of points (NTP): G4int
  PDG Encoding (PDG): G4int
  Parent ID (PID): G4int
  Particle Name (PN): G4String
G4TrajectoryPoint:
  Position (Pos):
    G4BestUnit (G4ThreeVector)
```

# Rich and smooth trajectories

```
G4TrajectoriesModel:
  Event ID (EventID): G4int
  Run ID (RunID): G4int
G4SmoothTrajectory:
  Charge (Ch): unit: e+ (G4double)
  Track ID (ID): G4int
  Initial kinetic energy (IKE): G4BestUnit (G4double)
  Initial momentum magnitude (IMag): G4BestUnit (G4double)
  Initial momentum (IMom): G4BestUnit (G4ThreeVector)
  No. of points (NTP): G4int
  PDG Encoding (PDG): G4int
  Parent ID (PID): G4int
  Particle Name (PN): G4String
G4SmoothTrajectoryPoint:
  Auxiliary Point Position (Aux): G4BestUnit (G4ThreeVector)
  Step Position (Pos): G4BestUnit (G4ThreeVector)
```

```
G4TrajectoriesModel:
  Event ID (EventID): G4int
  Run ID (RunID): G4int
G4RichTrajectory:
  Creator Model ID (CMID): G4int
  Creator Model Name (CMN): G4String
  Creator Process Name (CPN): G4String
  Creator Process Type Name (CPTN): G4String
  Charge (Ch): unit: e+ (G4double)
  Ending Process Name (EPN): G4String
  Ending Process Type Name (EPTN): G4String
  Final kinetic energy (FKE): G4BestUnit (G4double)
  Final Next Volume Path (FNVPath): G4String
  Final Volume Path (FVPath): G4String
  Track ID (ID): G4int
  Initial kinetic energy (IKE): G4BestUnit (G4double)
  Initial momentum magnitude (IMag): G4BestUnit (G4double)
  Initial momentum (IMom): G4BestUnit (G4ThreeVector)
  Initial Next Volume Path (INVPath): G4String
  Initial Volume Path (IVPath): G4String
  No. of points (NTP): G4int
  PDG Encoding (PDG): G4int
  Parent ID (PID): G4int
  Particle Name (PN): G4String
G4RichTrajectoryPoint:
  Auxiliary Point Position (Aux): G4BestUnit (G4ThreeVector)
  Process Defined Step (PDS): G4String
  Process Type Defined Step (PTDS): G4String
  Position (Pos): G4BestUnit (G4ThreeVector)
  Post-step-point status (PostStatus): G4String
  Post-step-point global time (PostT): G4BestUnit (G4double)
  Post-step Volume Path (PostVPath): G4String
  Post-step-point weight (PostW): G4double
  Pre-step-point status (PreStatus): G4String
  Pre-step-point global time (PreT): G4BestUnit (G4double)
  Pre-step Volume Path (PreVPath): G4String
  Pre-step-point weight (PreW): G4double
  Remaining Energy (RE): G4BestUnit (G4double)
  Total Energy Deposit (TED): G4BestUnit (G4double)
```

# Trajectory modeling and filtering (contd)

- The following model and filter factories are registered
- Typically, you create (instantiate) a model (see **B1/vis.mac**)
  - If you do not do this, the vis manager instantiates a **`drawByCharge`** model
- This creates a corresponding set of commands for customising properties
  - Use **help** and guidance to see them
  - See next slide

```
Registered model factories:
  generic
  drawByAttribute
  drawByCharge
  drawByOriginVolume
  drawByParticleID
  drawByEncounteredVolume

Registered filter factories:
  attributeFilter
  chargeFilter
  originVolumeFilter
  particleFilter
  encounteredVolumeFilter
```

```
G4VisManager: Using G4TrajectoryDrawByCharge as
 fallback trajectory model.
```

# More on trajectory models

- Trajectory models and filters are well described in the Book for Application Developers, but in-app guidance is somewhat cryptic
- Creating a model creates its specific commands
  - The commands are created “on the fly” for each model or filter

```
/vis/modeling/trajectories/create/drawByCharge
/vis/modeling/trajectories/drawByCharge-0/set 0 pink
```

charge

```
/vis/scene/add/trajectories rich
/vis/modeling/trajectories/create/drawByEncounteredVolume
/vis/modeling/trajectories/drawByEncounteredVolume-0/set Shape1 cyan
```

physical volume name

  - (Parameters as appropriate to the model)

# More on trajectory models

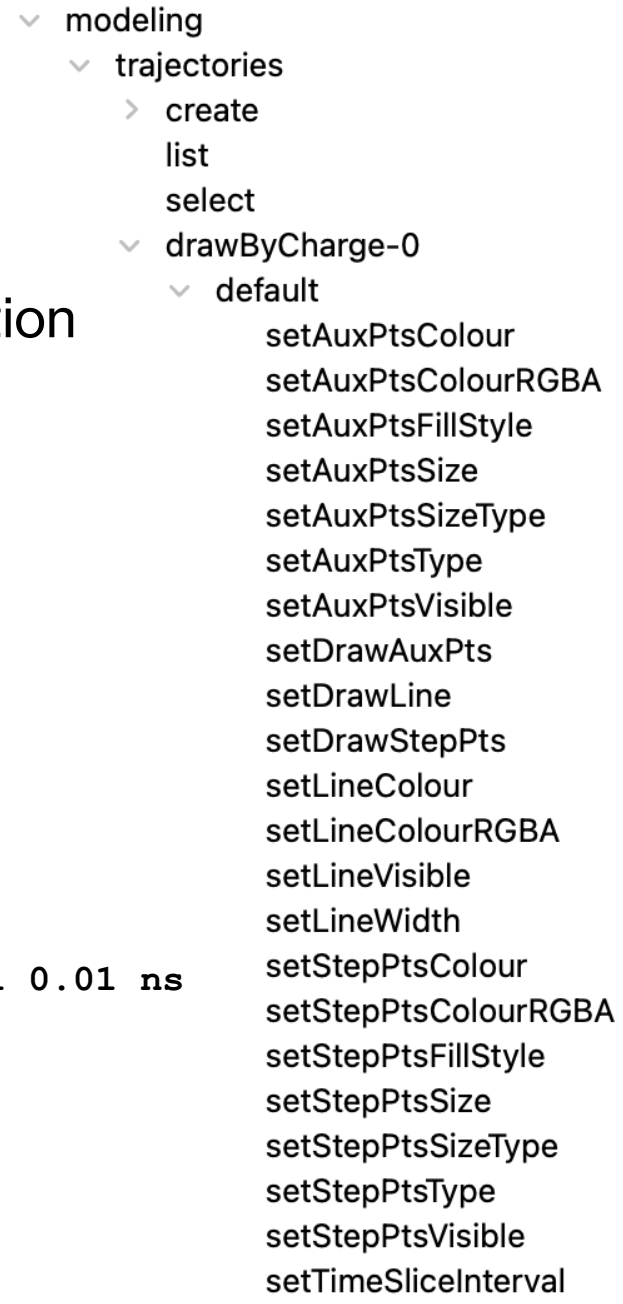
- Also, every model has a set of “defaults”
  - This creates a corresponding set of commands for customisation
    - Use **ls** or **help** and guidance to see them

```
/vis/modeling/trajectories/drawByCharge-0/default/setDrawStepPts true
/vis/modeling/trajectories/drawByCharge-0/default/setStepPtsSize 2
```

  - This is another way of setting colour
  - Of particular interest is **setTimeSliceInterval**
    - Needs rich trajectories

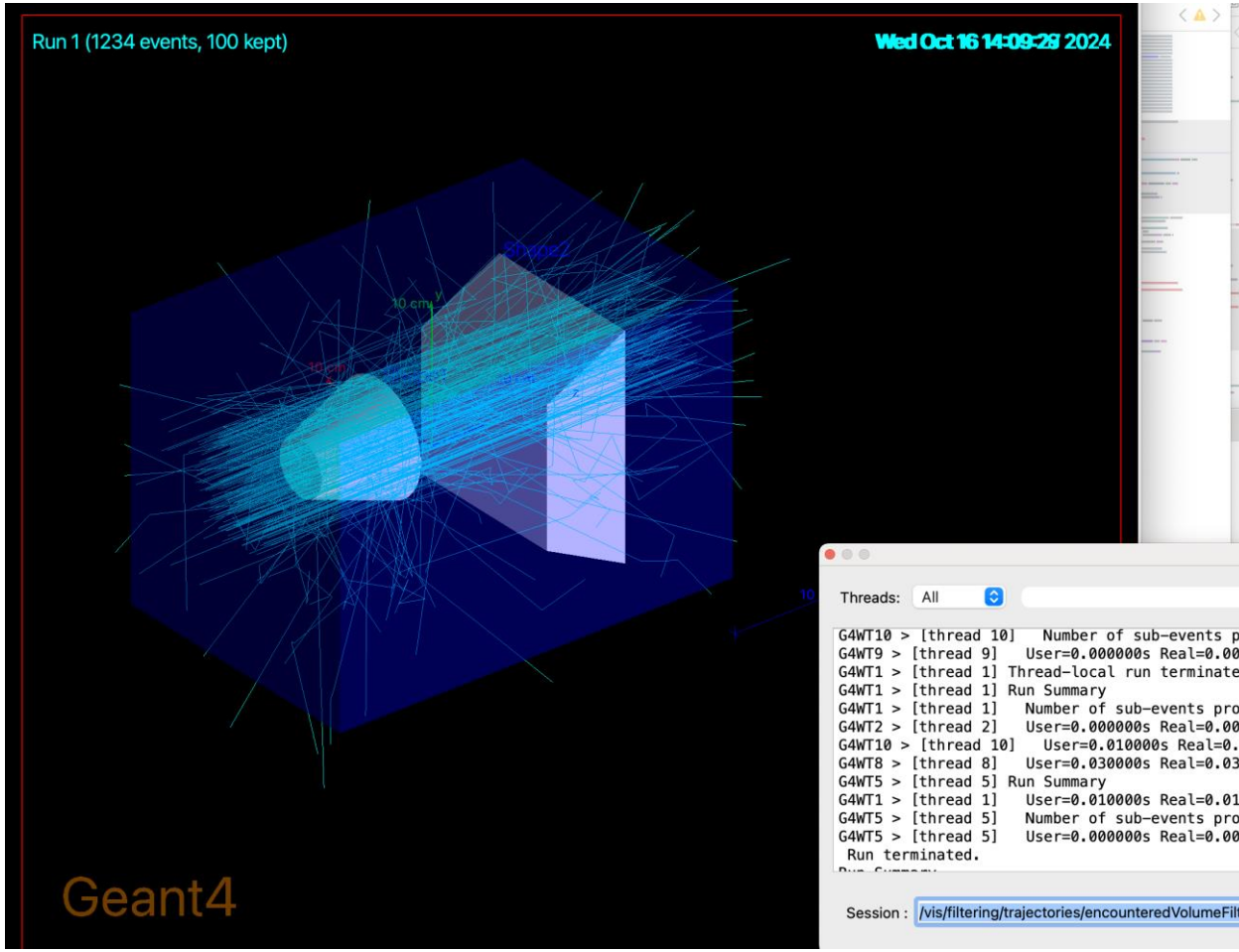
```
/vis/scene/add/trajectories rich
/vis/modeling/trajectories/drawByCharge-0/default/setTimeSliceInterval 0.01 ns
```

    - For its use for animations and movies, see later



# Filters

- Similarly, create a filter
- Filters can be chained




Note: You can pop out the session I/O window in the Qt GUI

```
# To draw only gammas:
/vis/filtering/trajectories/create/particleFilter
/vis/filtering/trajectories/particleFilter-0/add gamma

# To draw all except gammas
/vis/filtering/trajectories/particleFilter-0/invert true

# To draw only trajectories that encounter Shape1
/vis/filtering/trajectories/create/encounteredVolumeFilter
/vis/filtering/trajectories/encounteredVolumeFilter-0/add Shape1
```

# Drawing and filtering by attribute

- **drawByAttribute**

```
/vis/modeling/trajectories/create/drawByAttribute
/vis/modeling/trajectories/drawByAttribute-0/setAttribute CPN
/vis/modeling/trajectories/drawByAttribute-0/addValue brem_key eBrem
/vis/modeling/trajectories/drawByAttribute-0/brem_key/setLineColour magenta

/vis/modeling/trajectories/create/drawByAttribute
/vis/modeling/trajectories/drawByAttribute-1/setAttribute IMag
/vis/modeling/trajectories/drawByAttribute-1/addInterval interval1 0.0 keV 2.5MeV
/vis/modeling/trajectories/drawByAttribute-1/addInterval interval2 2.5 MeV 5 MeV
/vis/modeling/trajectories/drawByAttribute-1/interval1/setLineColourRGBA 0.8 0 0.8 1
/vis/modeling/trajectories/drawByAttribute-1/interval2/setLineColour pink
```

Creator Process Name – see list of G4Atts

Bremsstrahlung will be drawn in magenta

Initial momentum magnitude

- **attributeFilter**

```
/vis/filtering/trajectories/create/particleFilter
/vis/filtering/trajectories/particleFilter-0/add gamma
/vis/filtering/trajectories/particleFilter-0/invert true

/vis/filtering/trajectories/create/attributeFilter/
/vis/filtering/trajectories/attributeFilter-0/setAttribute Imag
/vis/filtering/trajectories/attributeFilter-0/addInterval 2.5 MeV 1000 MeV
```

Filter out gammas

Draw only tracks with initial momentum magnitude between 2.5 and 5 MeV

# Event keeping

- By default, the vis manager keeps 100 events
  - This default number can be changed with **/vis/scene/endOfEventAction**
  - They can be viewed at end of run
    - One by one: **/vis/reviewKeptEvents**
    - All: **/vis/viewer rebuild**
- If you want to keep your own events
  - For example, select a rare event in end-of-event action
  - **/vis/drawOnlyToBeKeptEvents**
    - This turns off event keeping by the vis manager
      - You can do this manually with **/vis/scene/endOfEventAction accumulate 0**
- Actually, the events are kept by the run manager
  - They are deleted at the start of the next run

```
auto evMan = G4EventManager::GetEventManager();
if (<selection-success>) {
  evMan->KeepTheCurrentEvent();
}
```

```
// From Geant4 11.3
if (<selection-success>) {
  event->KeepTheEvent();
}
```

# Viewer control

- **/vis/viewer/zoom**, **/vis/viewer/set/viewpointThetaPhi**, etc
  - Or, of course, interactively, with interactive systems
- **/vis/viewer/set/[wireframe|solid|cloud]**
  - Can be overridden by **G4VisAttributes::SetForceSolid**, et, or **VAMs** (later slides)
- Most graphical viewers are auto-refresh, but file-writing viewers need **/vis/viewer/refresh**
- The viewer is refreshed, either automatically or by hand, and it may decide to rebuild its graphical database (if any) from the Geant4 kernel (a “kernel visit”)
  - E.g., a change from wireframe to solid triggers a rebuild
  - A simple change of viewpoint does not normally require a kernel visit
    - So moving and rotating are very fast for viewers with a graphical database
- **/vis/viewer/rebuild** if in doubt
- It’s a good idea to disable auto-refresh while establishing the scene and the scene – see **examples/basic/B1/vis.mac**

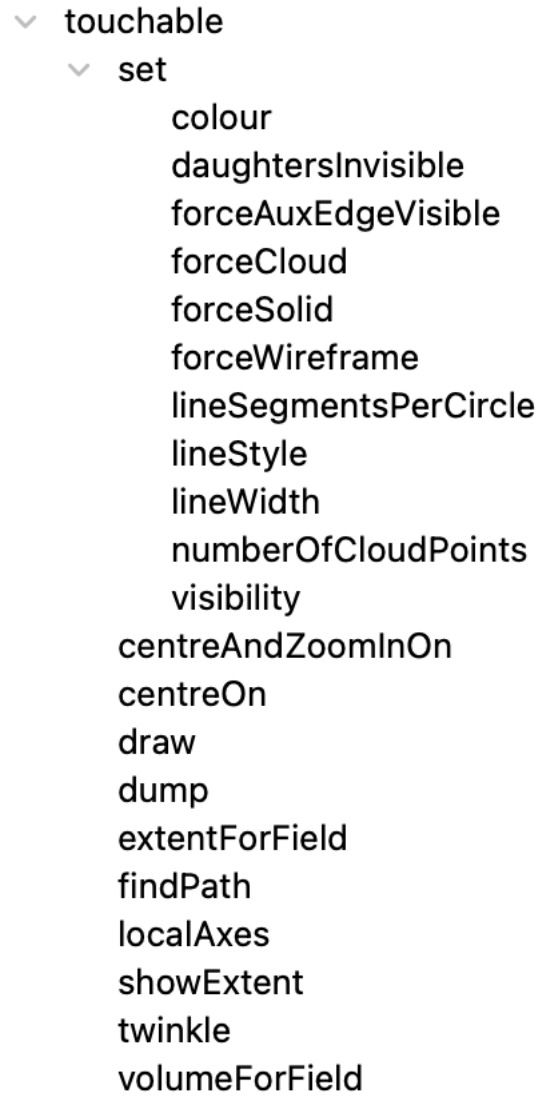
# Viewer commands

- viewer
  - default
    - hiddenEdge
    - style
  - set
  - addCutawayPlane
  - centreAndZoomInOn
  - centreOn
  - changeCutawayPlane
  - clear
  - clearCutawayPlanes
  - clearTransients
  - clearVisAttributesModifiers
  - clone
  - colourByDensity
  - copyViewFrom
  - create
  - dolly
  - dollyTo
  - flush
  - interpolate
  - list
  - pan
  - panTo
  - rebuild
  - refresh
  - reset
  - resetCameraParameters
  - save
  - scale
  - scaleTo
  - select
  - update
  - zoom
  - zoomTo

- set
  - timeWindow
    - displayHeadTime
    - displayLightFront
    - endTime
    - fadeFactor
    - startTime
  - all
  - autoRefresh
  - auxiliaryEdge
  - background
  - culling
  - cutawayMode
  - defaultColour
  - defaultTextColour
  - edge
  - explodeFactor
  - globalLineWidthScale
  - globalMarkerScale
  - hiddenEdge
  - hiddenMarker
  - lightsMove
  - lightsThetaPhi
  - lightsVector
  - lineSegmentsPerCircle
  - lineWidth
  - numberOfCloudPoints
  - picking
  - projection
  - rotationStyle
  - sectionPlane
  - specialMeshRendering
  - specialMeshRenderingOption
  - specialMeshVolumes
  - style
  - targetPoint
  - upThetaPhi
  - upVector
  - viewpointThetaPhi
  - viewpointVector

# Touchables

- **Touchables** are the leaves of the geometry hierarchy tree
- To print the tree: **/vis/drawTree**
  - Various verbosity options: **/vis/ASCIITree/verbose**
- To find the path to a touchable
  - **/vis/touchable/findPath <physical-vol-name>**
- Then, with information supplied, **/vis/set/touchable**
- Then **/vis/touchable/set/colour** (other options!!)
  - Also **/vis/touchable/twinkle** (and other fancy options!!)
- This is all achieved using **Vis Attribute Modifiers (VAMs)** – see **How do we do that**?
  - **VAMs** belong to the viewer, and may be copied with **/vis/viewer/clone** or **/vis/open** + **/vis/viewer/copyViewFrom**
    - So they can be different for different views of the same scene



# /vis/drawTree

- Very useful
- PV and LV by default

```
#  Now printing with verbosity 1
#  Format is: PV:n / LV (SD,RO)
#  Abbreviations: PV = Physical Volume,     LV = Logical Volume,
#                 SD = Sensitive Detector,  RO = Read Out Geometry.
  "World":0 / "World"
    "Envelope":0 / "Envelope"
      "Shape1":0 / "Shape1"
      "Shape2":0 / "Shape2"
G4ASCIITreeSceneHandler::EndModeling
```

- More information with verbosity 5

```
  "World":0 / "World" / "World"(G4Box), 20.736 L  , 1.20479 mg/cm3 (G4_AIR), 8.736 L  , 10.525 g
    "Envelope":0 / "Envelope" / "Envelope"(G4Box), 12 L  , 1 g/cm3  (G4_WATER), 10.8881 L  , 10.8881 kg
      "Shape1":0 / "Shape1" / "Shape1"(G4Cons), 1.75929 dL , 1.127 g/cm3  (G4_A-150_TISSUE), 1.75929 dL , 198.272 g
      "Shape2":0 / "Shape2" / "Shape2"(G4Trd), 9.36 dL , 1.85 g/cm3  (G4_BONE_COMPACT_ICRU), 9.36 dL , 1.7316 kg
Calculating mass(es)...
Overall volume of "World":0, is 20.736 L   and the daughter-included mass to unlimited depth is 12.8285 kg
```

# /vis/drawTree (contd)

```
#  Set verbosity with "/vis/ASCIITree/verbose <verbosity>":
#    <  10: notifies but does not print details of repeated volumes.
#    >= 10: prints all physical volumes (touchables).
#  The level of detail is given by verbosity%10:
#    >=  0: physical volume name.
#    >=  1: logical volume name (and names of sensitive detector and
#           readout geometry, if any).
#    >=  2: solid name and type.
#    >=  3: volume and density.
#    >=  5: daughter-subtracted volume and mass.
#    >=  6: physical volume dump.
#    >=  7: polyhedron dump.
#  and in the summary at the end of printing:
#    >=  4: daughter-included mass of top physical volume(s) in scene
#           to depth specified.
#  Note: by default, culling is switched off so all volumes are seen.
#  Note: the mass calculation takes into account daughters, which can be
#        time consuming.  If you want the mass of a particular subtree try:
#    /vis/drawTree <subtree-physical-volume-name>
#  Or if you want more control, for example:
#    /vis/open ATree
#    /vis/ASCIITree/verbose 14
#    /vis/scene/create
#    /vis/scene/add/volume <subtree-physical-volume-name> ! <depth>
#    /vis/sceneHandler/attach
#    /vis/viewer/flush
#  Note: dumping the physical volumes produces a lot of output. It is advisable
#        to select the volume of interest, as for a sub-tree above.
```

Printed first time used

# Saving, replaying and interpolating views

- **/vis/viewer/save [filename.g4view]**
  - By default, saves to **g4_00.g4view**, **g4_01.g4view**,...
- Then, to get the view again
  - **/control/execute g4_00.g4view**
- Or interpolate to get an animation
  - **/vis/viewer/interpolate**
  - One can make a movie (next slides)
- With time-slicing, you can watch the particles move through time

# Time-slicing

See examples/extended/visualization/movies

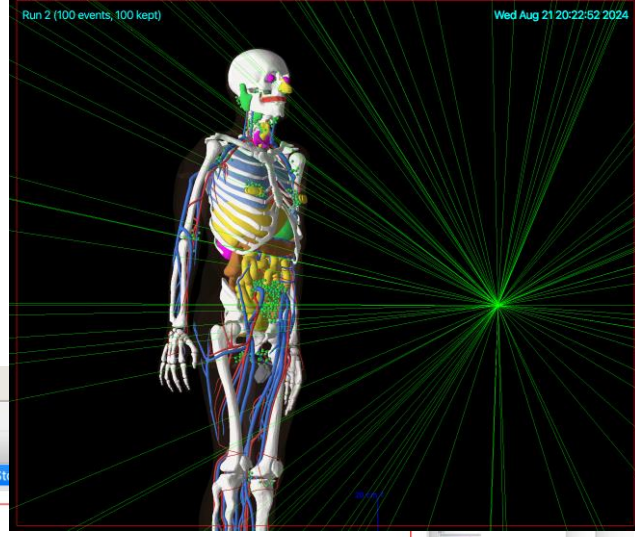
```
/vis/scene/add/trajectories rich
/vis/modeling/trajectories/drawByCharge-0/default/setTimeSliceInterval 0.01 ns
# Optionally add features (see guidance on /vis/viewer/set/timeWindow/)
/vis/viewer/set/timeWindow/displayLightFront true 0 0 -20 cm -0.01 ns
/vis/viewer/set/timeWindow/displayHeadTime true
/vis/viewer/set/timeWindow/fadeFactor 1
/run/beamOn
# Then set a time window and save
/vis/viewer/set/timeWindow/startTime 0 ns .1 ns
/vis/viewer/save
# Then zoom, pan etc to a view of interest
# Then set the next time window and save
/vis/viewer/set/timeWindow/startTime .5 ns .1 ns
/vis/viewer/save
# Then zoom, pan etc to a view of interest
# Then set the next time window and save
/vis/viewer/set/timeWindow/startTime 1 ns .1 ns
/vis/viewer/save
# Then another view, the next time window, and a save...
# ...repeat a few more times
# Then try
/vis/viewer/interpolate
```

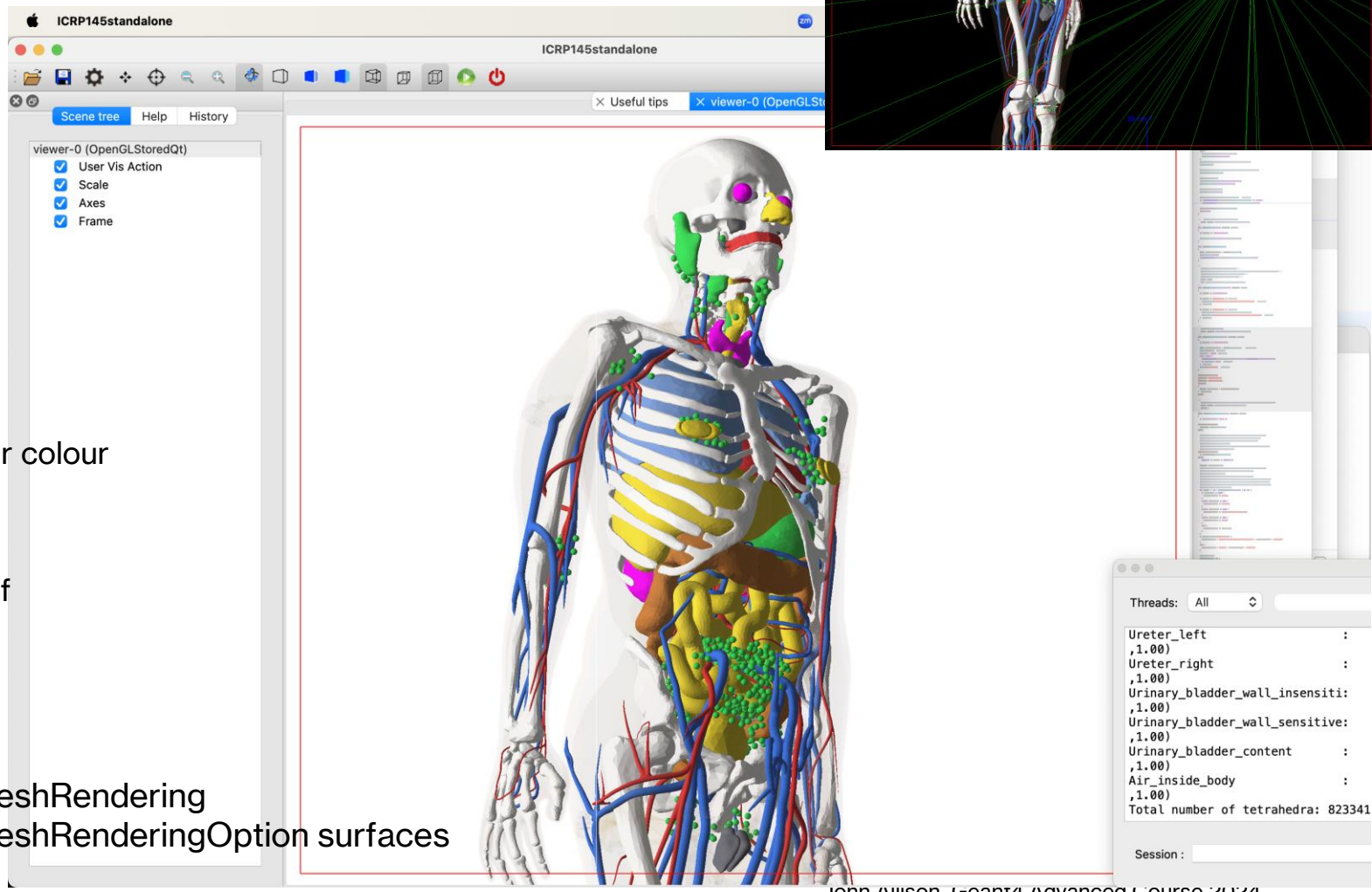
# Making movies

- On MacOS, QuickTime has a “Screen recording” feature
  - Just run your app
    - To record sound: on the small control panel: Options/Microphone and select
    - To stop recording, there’s a button on the top bar
- Alternatively, export images using **/vis/viewer/interpolate** (OpenGL only)
  - **/vis/viewer/interpolate ! ! ! ! export**
    - This produces lots of files
  - You can change export format with (for example)
    - **/vis/ogl/set/exportFormat jpg**
  - Then import them into your favourite movie maker – see Making a movie

# Meshes (G4VNestedParameterisation)

- Typically used for representing regions of variable material
  - E.g., human phantom, **examples/advanced/ ICRP145_HumanPhantoms**
- Only rectangular and tetrahedral meshes at present
  - Clever algorithm to remove shared surfaces (E. Tchernaiev)
  - We simply take the volumes of the G4VNestedParameterisation and eliminate shared surfaces
    - This is done separately for each material or colour
  - The resulting G4TessellatedSolid(s) are then drawn as normal
    - Typically, a 10 times reduction in number of facets and a corresponding speed-up
- Dots or surfaces available

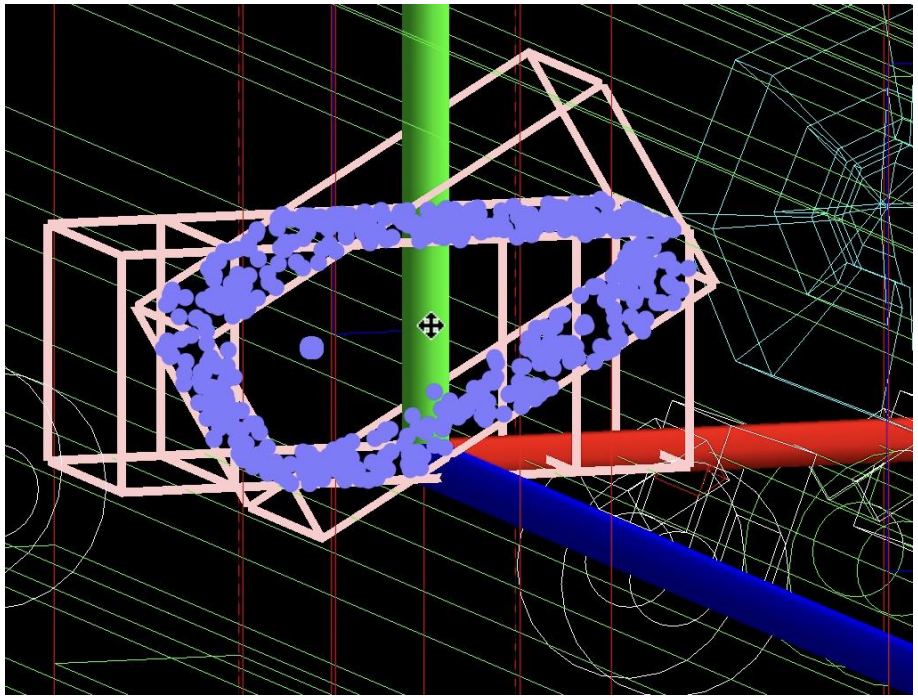



```
# Draw geometry:
/vis/viewer/set/specialMeshRendering
/vis/viewer/set/specialMeshRenderingOption surfaces
/vis/drawVolume
```

# Geometry overlaps and other useful things

- /vis/drawLogicalVolume <log-vol-name>
  - Boolean components
  - Geometry voxels
  - Local axes
  - Geometry overlaps
  - Demo

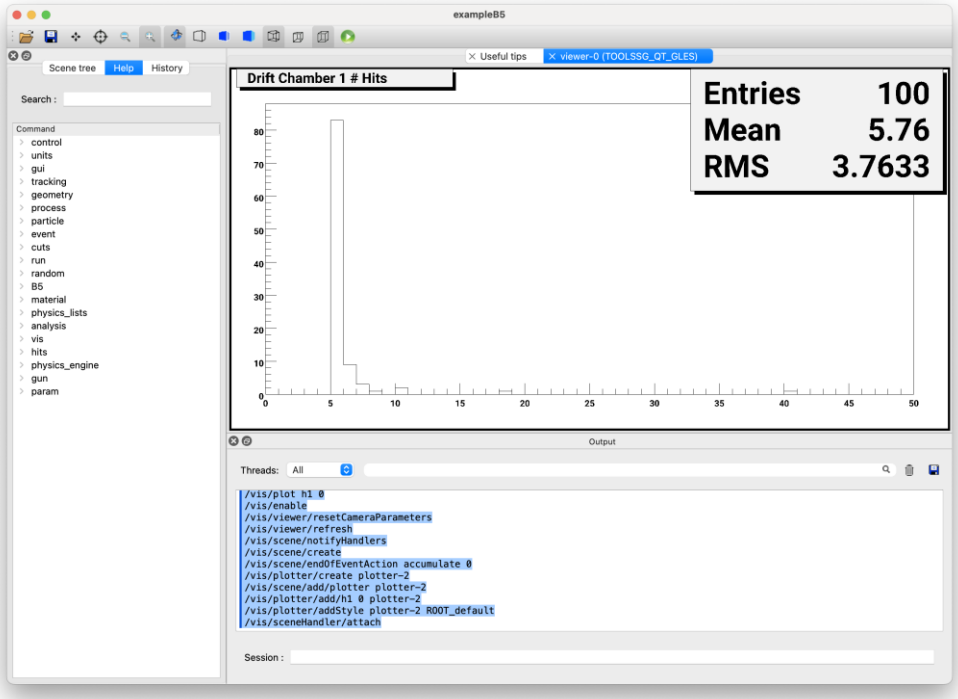


```
Command /vis/drawLogicalVolume
Guidance :
Draws logical volume with additional components.
Synonymous with "/vis/specify".
Creates a scene consisting of this logical volume and asks the
  current viewer to draw it. The scene becomes current.
Adds a logical volume to the current scene,
Shows boolean components (if any), voxels (if any), readout geometry
  (if any), local axes and overlaps (if any), under control of the
  appropriate flag.
  Note: voxels are not constructed until start of run -
 "/run/beamOn".  (For voxels without a run, "/run/beamOn 0".)
```

# Plotting

- If you have histograms registered with the Analysis Manager you can plot them at end of run
  - Only simple 1D and 2D with boxes for now
- At the end of run, the Vis Manager prints
- Only with ToolsSG (TSG) viewer
  - For best results, build with GEANT4_USE_FREETYPE=ON
    - Choice of styles – see guidance
    - From Geant4 11.3 (December 2024)
- Try basic example B5
  - /run/beam 100
  - /vis/plot h1 0

```
100 events have 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 histograms that can be viewed with visualization:
  2 h1 histograms(s)
  2 h2 histograms(s)
  List them with "/analysis/list".
  View them immediately with "/vis/plot" or "/vis/reviewPlots".
```



# Driver specific options and commands

- Ogl/export
- TSG export, offscreen
- OI beam history
- VTK cutting, clipping and export
  - **/vis/vtk/export FORMAT FILENAME**
  - **/vis/vtk/set/clipper 1**
  - **/vis/vtk/set/cutter 1**

# End of Concepts and Commands